

## University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

UNIVERSITY OF SOUTHAMPTON  
FACULTY OF PHYSICAL AND APPLIED SCIENCES  
School of Electronics and Computer Science

# Enabling Technologies for Distributed Body Sensor Networks

by  
Dirk de Jager

Thesis for the degree of Doctor of Philosophy

June 2012



UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES  
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Dirk de Jager

Low Power Wireless Sensor Networks, Preventative Healthcare and Pervasive Systems are set to provide long-term continuous monitoring, diagnosis and care for patients in the next few years. Distributed forms of these networks are investigated from a holistic point of view. Individual components of these systems including: sensors, software and hardware implementations are investigated and analysed. Novel sensors are developed for low power capturing of Body Sensor Network (BSN) information to enable long term use. Software frameworks are designed to enable these technologies to run on low power nodes as well as enabling them to perform evaluation of their data before transmission into the network. An architecture is designed to enable task distribution to intensive processing from low power nodes. Two forms of distributed BSNs are also developed: a horizontal network and a vertical network. It is shown that using these two types of networks enables information and task distribution allowing low power sensing nodes to evaluate information before transmission. These systems have the opportunity to revolutionise expensive acute episodic care systems of today, but are not currently being implemented or investigated to the extent that they could. The technological barriers to entry are addressed in this thesis with the investigation and evaluation of distributed body sensor networks. It is shown that horizontal networks can distribute information efficiently, while vertical networks can distribute processing efficiently.





# Contents

<b>Front matter</b>	<b>v</b>
List of Figures . . . . .	ix
List of Tables . . . . .	xiii
Nomenclature . . . . .	xv
Abbreviations . . . . .	xvii
Declaration . . . . .	xix
Publications . . . . .	xxi
Acknowledgements . . . . .	xxiii
 <b>1 Introduction</b>	 <b>1</b>
1.1 Problem Statement . . . . .	2
1.2 Research Aims . . . . .	5
1.3 Prototype Horizontal and Vertical BSN Platforms . . . . .	6
1.4 Scope and Contribution . . . . .	6
1.5 Document Structure . . . . .	8
 <b>2 Body Sensor Network Platforms for Healthcare</b>	 <b>11</b>
2.1 The Human++ Research Program . . . . .	16
2.2 The Smart and Aware Pervasive Healthcare Environments Project . . . . .	18
2.3 BioMobius . . . . .	19
2.4 CodeBlue . . . . .	20
2.5 Discussion . . . . .	21
 <b>3 Developing Low Power Sensors and Sensor Interfaces</b>	 <b>25</b>
3.1 Guarded Capacitor Sensor . . . . .	26
3.1.1 Guarded Capacitive Sensor . . . . .	26
3.1.2 Measuring the Capacitance Change . . . . .	28
3.1.3 Prototype Sensor System . . . . .	29
3.1.4 Results . . . . .	30
3.1.5 Conclusions . . . . .	32
3.2 Low Power Camera Interface . . . . .	36
3.2.1 Parallel Camera Interface . . . . .	37
3.2.2 The Low Power MCU Interface . . . . .	37
3.2.3 Prototype . . . . .	40
3.2.4 Results . . . . .	42
3.2.5 Conclusions and Future Work . . . . .	45
3.3 Discussion . . . . .	46

<b>4</b>	<b>Software Frameworks supporting Distributed Sensor Networks</b>	<b>49</b>
4.1	Information Costing and Valuation Framework . . . . .	50
4.1.1	System Overview . . . . .	54
4.1.2	Information Value . . . . .	56
4.1.3	Information Cost . . . . .	62
4.1.4	Simulated Value/Costing using real data . . . . .	67
4.1.5	Discussion . . . . .	89
4.2	Simple Wireless Network Framework (SWNF) . . . . .	91
4.2.1	Simple Wireless Network Framework . . . . .	93
4.2.2	Hardware Implementations . . . . .	100
4.2.3	Framework vs Operating System . . . . .	101
4.2.4	Results . . . . .	101
4.2.5	Conclusions . . . . .	107
4.3	Tiered Vertical Distribution Framework . . . . .	109
4.3.1	System Overview . . . . .	110
4.3.2	Component Description . . . . .	111
4.3.3	Sensing Device . . . . .	112
4.3.4	Network Gateway . . . . .	113
4.3.5	Service Application . . . . .	114
4.3.6	Conclusions . . . . .	114
4.4	Discussion . . . . .	116
<b>5</b>	<b>Prototype Horizontal and Vertical Body Sensor Network Platforms</b>	<b>117</b>
5.1	Distributed Personal Body Sensor Network (miNet) . . . . .	117
5.1.1	System Overview . . . . .	118
5.1.2	Hardware . . . . .	118
5.1.3	Software Framework . . . . .	121
5.1.4	Information Valuation and Costing Model Implementation . . . . .	126
5.1.5	Results . . . . .	128
5.1.6	Conclusions . . . . .	132
5.2	Memory Assistance Device: DejaView . . . . .	133
5.2.1	Conceptual use and Scenario . . . . .	133
5.2.2	DejaView Device . . . . .	136
5.2.3	Android Application . . . . .	138
5.2.4	Internet Service Application . . . . .	139
5.2.5	Results . . . . .	139
5.2.6	Conclusions . . . . .	141
5.3	Discussion . . . . .	143
<b>6</b>	<b>Conclusions and Future Work</b>	<b>145</b>
6.1	Developing Low Power Sensors and Sensor Interfaces . . . . .	145
6.2	Software Frameworks supporting Distributed Sensor Networks . . . . .	146
6.3	Prototype Horizontal and Vertical Body Sensor Network Platforms . . . . .	146
6.4	Discussion . . . . .	147
6.5	Future Work . . . . .	147
<b>A</b>	<b>SWNF Example Applications</b>	<b>151</b>

---

<b>B Circuit Diagrams</b>	<b>157</b>
<b>C Printed Circuit Board Layouts</b>	<b>171</b>
<b>References</b>	<b>175</b>



# List of Figures

1.1	Centralised BSN . . . . .	1
1.2	Envisaged Body Sensor Network Overview . . . . .	4
1.3	Horizontal Distribution . . . . .	5
1.4	Vertical Distribution . . . . .	6
1.5	Mindmap of Thesis Outline . . . . .	9
2.1	Example of time-limited motion artifacts on SPO2 signal . . . . .	12
2.2	Body Sensor Network Overview . . . . .	15
2.3	Star topology BSN . . . . .	16
2.4	Thermoelectric watch, wireless radio, and COTS finger SpO <sub>2</sub> meter . . . .	17
2.5	Intelligent Plaster with radio and antenna showing . . . . .	18
2.6	CodeBlue Architecture . . . . .	20
3.1	Parallel Plate Capacitor . . . . .	27
3.2	Guarded Capacitor Sensor . . . . .	27
3.3	Two states are shown for the sensor during inhalation and exhalation. The effect of the inhalation and exhalation on the sensor are shown . . . .	28
3.4	Relaxation Oscillator . . . . .	28
3.5	Theoretical Plot of Frequency vs Capacitance . . . . .	29
3.6	Capaciflective Sensor . . . . .	29
3.7	Relaxation Oscillator tests using fixed values of capacitance . . . . .	30
3.8	Testing Sensor Plate Configurations . . . . .	31
3.9	Capacitive Sensor Plate Designs. 1-6 follows from left to right . . . . .	31
3.10	Capacitive Sensor Frequency ranges . . . . .	33
3.11	Result of 5 slow breaths taken using Sensor Design 3 with 1.5M Oscillator feedback resistor . . . . .	33
3.12	Result of 9 slow breaths taken using Sensor Design 3 & 6 with 1.5M & 1M feedback resistors. The signals are in anti-phase as some measured areas of the body, such as around the diaphragm between the ribs, force the sensor plate closer to the body during exhalation as opposed to inhalation and thus display an inverted signal. . . . .	34
3.13	Processing of respiratory signal . . . . .	34
3.14	Camera interface clocking signals . . . . .	37
3.15	Overview of Low Power Camera Interface . . . . .	38
3.16	CPLD Code . . . . .	40
3.17	Camera Interface Test Board . . . . .	41
3.18	Sample Images captured using the interface . . . . .	43
3.19	Current draw of different compressed image resolutions for OV5642 . . . .	44

3.20	Comparison of Delays between image capture and number of images for a 1AH LiPo Battery . . . . .	45
3.21	Comparison of the total lifetime for capture using the interface for a 1000mAH LiPo Battery . . . . .	46
4.1	Messages increase polynomially with both increase in nodes as well as increase in hops . . . . .	50
4.2	On node Information Valuation and Costing model overview . . . . .	54
4.3	Temporal valuation and reconstructed signals of 3 different signals . . . .	58
4.4	Intel Lab Research Mote Layout . . . . .	67
4.5	Intel Lab Dataset Details . . . . .	69
4.6	Individual measured signals averaged over all time . . . . .	70
4.7	Overall Bandwidth Occupancy for entire network show in packets per hour	71
4.8	Node Statistics taken from raw data . . . . .	72
4.9	Packet Maps . . . . .	73
4.10	Matrix Map of Node correlation . . . . .	74
4.11	Comparison of best (Nodes 1 and 33) and worst (Nodes 15 and 49) correlated nodes . . . . .	76
4.12	Entropy and RMS Error of Intel Data . . . . .	78
4.13	Temporally Valuable Nodes over total time . . . . .	79
4.14	Spatially Valuable Nodes over total time . . . . .	80
4.15	RMS Errors of Temporally, Spatially and Total signals . . . . .	80
4.16	Information Costs . . . . .	83
4.17	Bandwidth in Minutes . . . . .	86
4.18	Results of Information Valuation/Costing system on all nodes . . . . .	87
4.19	RMS Errors . . . . .	88
4.20	Node 11's humidity signal raw and compressed dataset . . . . .	89
4.21	Overview of the Unified Framework . . . . .	94
4.22	The core SWNF Components . . . . .	94
4.23	Simple Wireless Network Framework Messaging Operation . . . . .	95
4.24	Comparison of Lines of Code . . . . .	103
4.25	Comparison of the Number of Files used in Build . . . . .	104
4.26	Number of Directories . . . . .	104
4.27	Compiled Application Sizes . . . . .	106
4.28	Energy usage comparison for Different Applications showing the hardware implementation differences . . . . .	108
4.29	Movement of data through proposed system . . . . .	110
4.30	Implementation of the Tiered Vertical Distribution Architecture . . . . .	111
4.31	Data Message Transfer . . . . .	112
4.32	Data Capture Process . . . . .	112
4.33	Rule Structure . . . . .	113
4.34	Information Shard . . . . .	114
5.1	Sensor boards developed for the miNet system . . . . .	120
5.2	Software Overview of miNet Node . . . . .	121
5.3	Flow of data captured from miNet sensors . . . . .	122
5.4	Overview of Data to Information transfer in miNet . . . . .	122
5.5	Accelnode Activity Classification sensor model . . . . .	126

5.6	ECG Sensor model . . . . .	126
5.7	Functional Test showing miNet working over three colocated nodes measuring temperature . . . . .	128
5.8	Temporal Information Valuation on Real miNet data . . . . .	130
5.9	Signal vs Received Signal of Information Valuation and Costing system active . . . . .	131
5.10	The DejaView Device Version 2 . . . . .	133
5.11	The DejaView device overview . . . . .	138
5.12	Data captured by the Dejaviw System . . . . .	139
5.13	A Screenshot of the Internet Service Application . . . . .	140
5.14	DejaView Device Sensing Current Draw . . . . .	140
5.15	DejaView Device Capture Image Current Draw . . . . .	141
5.16	Data captured by the Dejaviw System showing the associated signals . . . . .	142
B.1	Low Power Camera Interface Camera, CPLD and Memory . . . . .	158
B.2	Low Power Camera Interface Microcontroller Interface . . . . .	159
B.3	Low Power Camera Interface USB and Power . . . . .	160
B.4	Cardionode Circuit Diagram . . . . .	161
B.5	Respinode Circuit Diagram . . . . .	162
B.6	Spoxnode Circuit Diagram . . . . .	163
B.7	Accelnode Circuit Diagram . . . . .	164
B.8	EFM2500T Circuit Diagram . . . . .	165
B.9	DejaView Device Power and USB Interface . . . . .	166
B.10	DejaView Device Sensors . . . . .	167
B.11	DejaView Device Communication Connections . . . . .	168
B.12	DejaView Device Microcontroller Connections . . . . .	169
C.1	Accelnode PCB . . . . .	171
C.2	BTNode PCB . . . . .	172
C.3	Cardionode PCB . . . . .	172
C.4	EFM2500T Processor PCB . . . . .	172
C.5	Respinode PCB . . . . .	173
C.6	Spoxnode PCB . . . . .	173
C.7	DejaView Device Cutdown Version 2 (CD2) PCB . . . . .	174





# List of Tables

2.1	Analog BSN Comparison . . . . .	13
2.2	Digital BSN Comparison . . . . .	14
3.1	Guarded Capacitive Sensor Plate Design and Configurations . . . . .	31
3.2	Guarded Capacitive Sensor Plate Testing Results. These readings were gathered using an HP Oscilloscope model DSO3062A. The values were averaged over a period of 20-30 seconds for each design. . . . .	32
3.3	Comparison of Wireless Node Camera Interfaces . . . . .	36
3.4	Interface Components and related Energy Consumption . . . . .	39
3.5	Camera Sensors . . . . .	42
3.6	Camera Energy Comparison for the OV5642 Camera Sensor . . . . .	44
4.1	Size comparison of Input Signal, average temporal information and Reconstructed signal for Figure 4.3 $I_{vt}$ of 0.5 . . . . .	59
4.2	Extract of the CC2420 Radio Module Datasheet showing the related output power to current consumption of the radio module . . . . .	63
4.3	Some Correlation Values for Node Pairs . . . . .	75
4.4	Network Statistics . . . . .	76
4.5	Overall Averaged Information Entropy of all node information . . . . .	78
4.6	Costing Parameters . . . . .	84
4.7	Framework Implementation Comparison of Template Applications without hardware implementation . . . . .	100
4.8	Code Complexity . . . . .	103
4.9	Compiled Application Results . . . . .	105
5.1	Topics and Nodes providing the data . . . . .	122
5.2	Application Level Packet Structure . . . . .	123
5.3	Data Packet Key-Value Pair . . . . .	124
5.4	SHHP Keys and associated Data Values . . . . .	125
5.5	Sample Images and discussion of the assistance the memory can provide .	135
5.6	DejaView Sensor Selection . . . . .	136
5.7	Timing and Energy comparison of information capture . . . . .	141



# Nomenclature

$c_0$	Collision Factor
$d_{max}$	Maximum Transmission Distance
$\epsilon_{amp}$	Energy used for Transceiver Amplifier
$\epsilon_{sample}$	Energy used for Sampling Data
$\epsilon_{process}$	Energy used for Processing Data
$E_{elec}$	Energy used for Transceiver Circuitry
$E_{processing}$	Energy used for Processing
$E_{rx}$	Energy used for Reception
$E_{sampling}$	Energy used for Sampling
$E_{tx}$	Energy used for Transmission
$I_c$	Total Information Cost
$I_{ce}$	Information Energy Cost
$I_{cb}$	Information Bandwidth Cost
$I_v$	Total Information Value
$I_{vs}$	Spatial Information Value
$I_{vt}$	Temporal Information Value
$k_{data}$	Data in bits
$k_{header}$	Header data in bits
$p(col)$	Probability of Collision
$R_{max}$	Wireless Network Bandwidth
$RSSI$	Received Signal Strength Indicator
$T_{tx}$	Time of Data Transmission



# Abbreviations

ACQP	Acquisitional Query Processor
ADMR	Adaptive Demand-Driven Multicast Routing
ATEF	Analog Telemetry Efficiency Factor
BAN	Body Area Network
BSN	Body Sensor Network
CCM	Compact Camera Module
CNS	Central Nervous Response
COTS	Commercial Off The Shelf
CPLD	Complex Programmable Logic Device
CSMA	Carrier Sense Multiple Access
DSP	Digital Signal Processor
DTEF	Digital Telemetry Efficiency Factor
ECG	Electrocardiogram
EEG	Electroencephalography
EMG	Electromyography
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
FRAM	Ferroelectric Random Access Memory
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HIL	Hardware Independent Layer
HPL	Hardware Presentation Layer
IR	Interference Range
ITU	International Telecommunication Union
LPU	Local Processing Unit
LQI	Link Quality Indication
MAC	Media Access Control

MBC	Model Based Coding
MEMS	Microelectromechanical System
MEWS	Modified Early Warning Score
MIPI	Mobile Industry Processor Interface
MML	Minimum Message Length
MCU	Microcontroller Unit
MVC	Model-View-Controller
OEM	Original Equipment Manufacturer
OFDMA	Orthogonal Division Multiple Access
OSI	Open Systems Interconnection model
QoS	Quality-of-Service
RIP	Respiratory Inductance Plethysmography
RSSI	Received Signal Strength Indication
SHHP	Simple Helping Hand Protocol
SNR	Signal-to-Noise Ratio
SpO <sub>2</sub>	Saturation of Peripheral Oxygen
SWNF	Simple Wireless Node Framework
TDMA	Time Division Multiple Access
ToF	Time of Flight
UWB	Ultra-wideband
WSN	Wireless Sensor Network
WBAN	Wireless Body Area Network

# Declaration of Authorship

I, Dirk de Jager, declare that the thesis entitled ‘Enabling Technologies for Distributed Body Sensor Networks’ and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as given in the Publications section

**Signed**

**Date**





# Publications

- [1] M. Loudon, T. Ajmal, U. Rivett, D. De Jager, R. Bain, R. Matthews, and S. Gundry, “A ‘Human-in-the-Loop’ mobile image recognition application for rapid scanning of water quality test results,” in *First International Workshop on Expressive Interactions for Sustainability and Empowerment (EISE 2009)*, London, Oct. 2009.
- [2] D. De Jager, E. B. Mazomenos, A. K. Banerjee, K. Maharatna, and J. S. Reeve, “A low-power simplified-MEWS scoring device for patient monitoring,” in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2010 4th International Conference on*, pp. 1–4, Mar. 2010.
- [3] E. Mazomenos, D. De Jager, J. Reeve, and N. White, “A Two-Way time of flight ranging scheme for wireless sensor networks,” in *8th European Conference on Wireless Sensor Networks (EWSN 2011)*, pp. 163–178, Bonn, Germany, Feb. 2011.
- [4] D. De Jager, A. L. Wood, G. V. Merrett, B. M. Al-Hashimi, K. O’Hara, N. R. Shadbolt, and W. Hall, “A Low-Power, distributed, pervasive healthcare system for supporting memory,” in *1st ACM MobiHoc Workshop on Pervasive Wireless Healthcare (MobileHealth 2011)*, Paris, France, May 2011.
- [5] D. de Jager and J. S. Reeve, “Efficient information valuation and costing for distributed wireless sensor networks,” *IET Wireless Sensor Systems*, 2011, SUBMITTED.
- [6] G. Merrett, W. Hall, B. Al-Hashimi, D. De Jager, and N. Shadbolt, “Memory aid,” Great Britain Patent Application PCT/EP2011/066 049, Filed 15 September. 2011.



## Acknowledgements

Firstly, I would like to sincerely thank my supervisor, Dr Jeff Reeve for his support, insight and guidance throughout my thesis. His calm and reassuring direction along with encouragement throughout this time has been invaluable to me. I would also like to thank my second supervisor, Dr Nick Harris for his support during this time.

I would like to thank my colleagues in the ESD laboratory for their help, discussion, debate and counsel. They include Evangelos Mazomenos, Karim el Shabrawy, David Handford, Musta Imran Ali and Alex Weddell.

I would also like to thank Dr Geoff Merrett and Prof Bashir Al-Hashimi for their belief in me, as well as the support and funding they provided for me. I would also like to thank Dr Reuben Wilcock for all his input, support and constructive criticism.

My parents Gerhard and Karin de Jager have been a continuous source of emotional, educational and financial support for which I am always indebted. Finally, I would like to thank my fiancée Jessica for her unfaltering support and belief in me.



# Chapter 1

## Introduction

Body Sensor Networks (BSN) described by Guang-Zhong Yang [7] revolve around a central on body processor called the Local Processing Unit (LPU) which harvests data from numerous wireless sensor nodes on-body and in-body. The nodes then transmit the information further to a central monitoring server or other set of distant servers as shown in Figure 1.1.

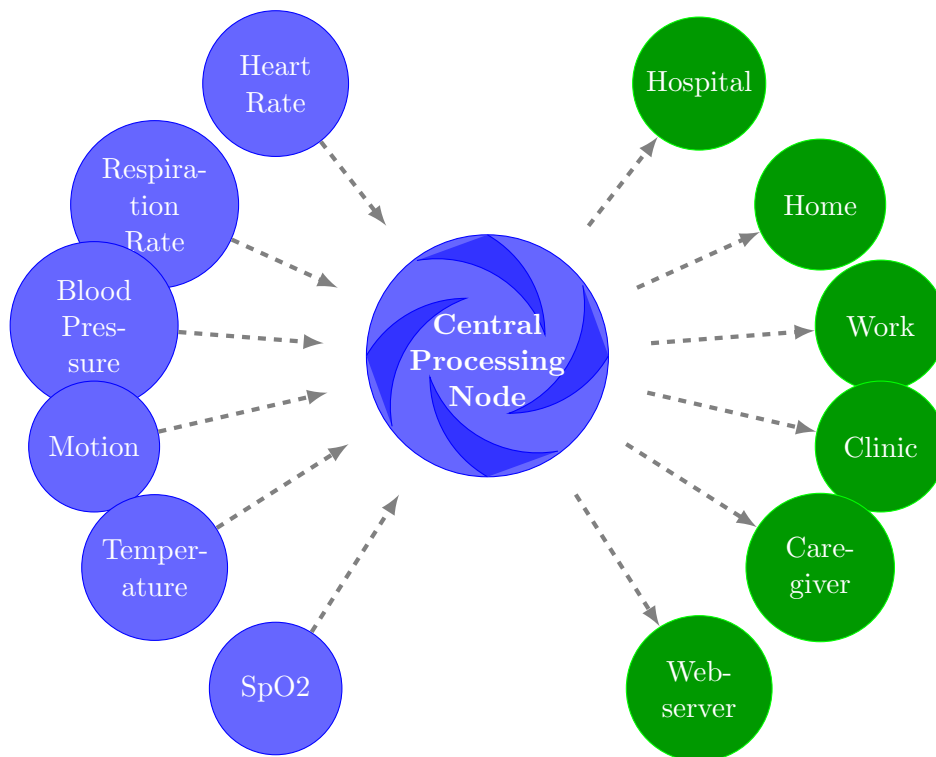


Figure 1.1: Centralised BSN

Although Key Technical challenges are still being investigated to enable these types of systems [8], and this serves as a good model for *collection* of data, it does not imply or

give good direction in terms of where the predominant processing, viewing or use of this information is performed. The term we describe as *distributed* body sensor networks in this text attempts to study and analyse where the processing of information should be performed, what the arguments are for and against the processing of the information close to the body, or far away on a distant server.

It is important to study this as the average age of the global population is increasing [9], long term manageable chronic disease is becoming the largest global killer [10, 11]. Current models of healthcare are intensive and cannot support the growing number of people required to be managed. Body Sensor Networks have the potential to be a disruptive technology enabling global care for all. This can be achieved through the ability of a BSN to monitor a patient continuously for extended periods of time enabling feedback and allowing self-managed preventative healthcare and decrease the burden of current healthcare providers [12] by allowing them to concentrate their efforts on more critical patients.

Distribution of node sensing, data collection, information management, and processing is investigated on different platforms. This work attempts to explain some other technical challenges that have prevented BSNs from gaining wide-spread acceptance, and how we can achieve this using different techniques.

## 1.1 Problem Statement

In this work, monitoring patients for chronic health conditions in the long term is investigated using the potentially disruptive technology of BSNs. Non-invasive sensor and sensor-nodes are only considered so as no medical procedure is required to implant or activate the BSN. The patient should be monitored without the limitation of being within a restricted environment such as the patient's home.

BSN Nodes are required to be small and lightweight so as to not encumber the patient who is to wear them, yet they are also required to operate for extended periods of time so as to not require continual recharging or replacing of batteries. The nodes should operate without restricting the patient with cables to cumbersome hardware (such as a portable ECG Monitor) or direct communication with a laptop which the patient is required to carry around. These are regarded as too encumbering for the patient. Smaller devices, the size of mobile phones are regarded as portable enough for a patient to carry around for continuously.

As the device size and weight is of utmost importance to prevent encumbrance, an efficient energy storage or harvesting system is required to be used by these nodes. This is not enough as although the energy storage or harvesting system might be extremely efficient, the usage of this energy is of utmost importance to ensure maximum lifetime

operation. This work does not investigate efficient energy storage or harvesting systems, only the efficient use of the energy provided.

To use the energy provided efficiently, the tasks which the node performs need to be investigated. Sensing of the physical parameters and processing of the acquired signals needs to be optimised in terms of energy to ensure that the device size and weight remains small and light while still providing robust and accurate information on the patient being monitored.

Not only are the devices required to be small and light, but the information gathered by the nodes is required to be continuous, accurate and error free. Common errors of on-body sensors are often time-limited motion artefacts which occur from patient movement independent of the sensor. To remove motion artefacts, and improve signal quality multiple sensing nodes can be used.

To address these issues, a universal body sensor network for healthcare is envisaged in Figure 1.2. In this system a patient is continuously monitored by a set of BSN Nodes for numerous conditions simultaneously. These nodes perform concurrent tasks around the human body to continuously inform the patient and assigned healthcare providers whether the patient is maintaining their health and well-being. The nodes operating around the body are not linked to a centralised controlling node, but act co-operatively in a distributed fashion. If any of the nodes are removed from the system, the other nodes will continue operating and fulfilling their required tasks. Using such a system, a heart specialist can request information about the cardiovascular state of the patient from the network while simultaneously a dietician can request information regarding blood pressure from the same network. The specialists can also gather historic information relevant to their inquiry from the patient.

From Figure 1.2, one can see the five different aspects of the system. Sensing information from the human body is the initial stage of diagnosis and monitoring and therefore a highly important step to creating an informed and correct assessment of the patient's health. Data can then be distributed among the nodes in a co-operative manner. The nodes then need to be able to distribute or assign specific processing tasks to information if they are not able to process the information themselves. The individual nodes will need to work together to solve larger problems such as activity recognition or classification in a collaborative way. All the information gathered by the distributed network then needs to be presented to the interested and required parties involved in the network.

Although this model appears to be a good solution to achieving distributed body sensor networks, it is fraught with difficulties. Wireless Sensor Networks are inherently application specific, and Body Sensor Networks due to the multiple use-cases, sensing requirements, and shareholders involved are even more so. The overall sensing tasks required to be performed by the network are often different and conflicting, let alone the processing requirements required from each sensed signal.



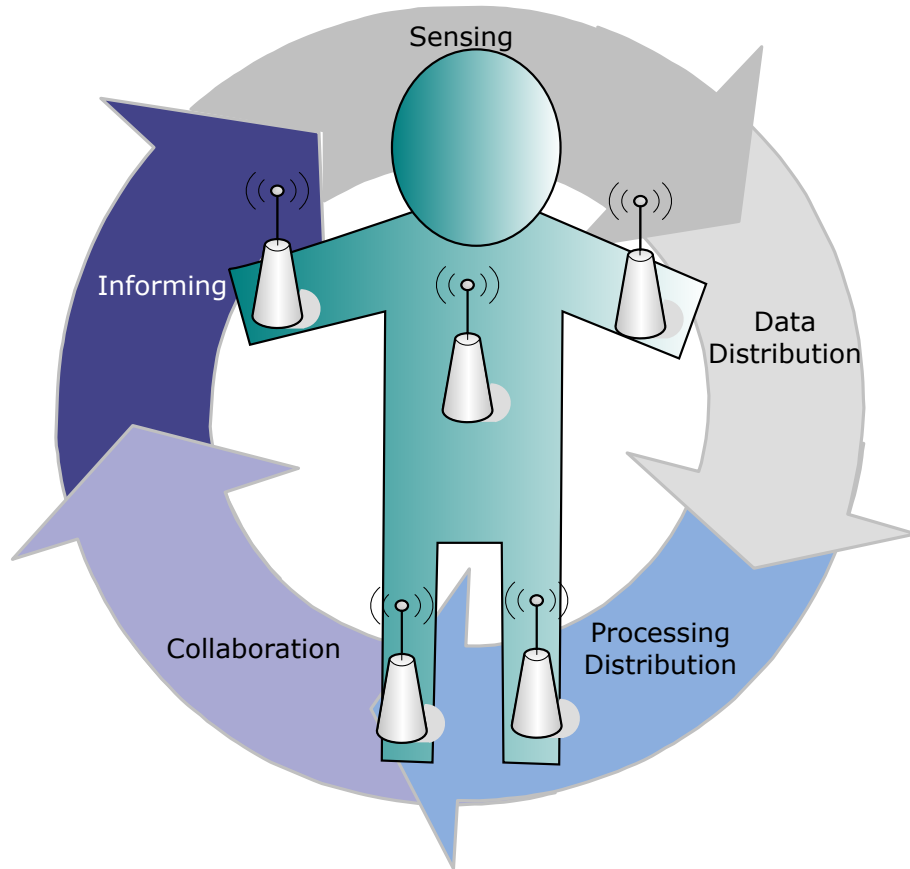


Figure 1.2: Envisaged Body Sensor Network Overview

For these light-weight sensors to operate in the long term, new designs and interfaces need to be developed to ensure energy efficiency and signal quality.

To distribute and disseminate data among different nodes, energy efficient frameworks need to be developed to allow individual nodes to compute the value and cost of their information within the network they are part of.

To alleviate much of the application specificity inherent in BSNs, a common platform is required for developers to use, which allows efficient and standardised access to the hardware components, while still providing efficient usage of hardware resources and simple usage thereby decreasing development time.

For the devices to collaborate and inform the interested parties, standards and system architectures are required to communicate the information in a robust manner.

It is thus important to investigate and understand the individual component requirements of a Body Sensor Network and build a network having understood these components. Developing these different components can bring insight into a technological system to realise such a complete system. Thus a set of enabling technologies for Body Sensor Networks is developed.

To ensure that these systems are practically viable and implementable, prototype systems using these tools are required to be designed and implemented.

## 1.2 Research Aims

Specifically this work investigates enabling technologies for distributed BSN along two dimensions of distribution in BSNs: vertical and horizontal distribution. How do we enable these forms of distribution, what are the challenges surrounding these forms of distribution and what are the advantages of using these types of distribution? Figure 1.3 describes what is interpreted as horizontal distribution, while Figure 1.4 illustrates vertical distribution.

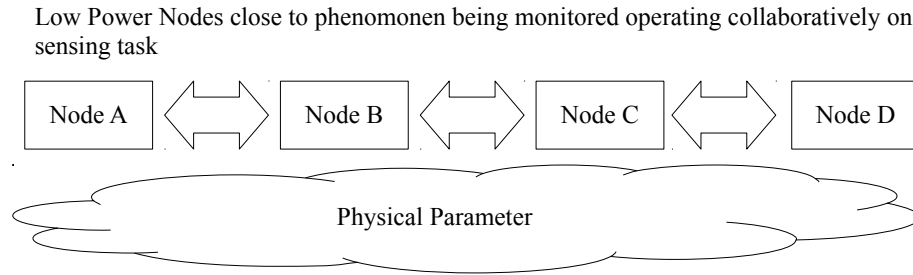


Figure 1.3: Horizontal Distribution

Horizontal distributed nodes are defined as nodes which are mostly all close to the physical parameters being monitored and are mostly computationally equivalent, but are predominately low power processing nodes as they are required to be physically close to the data source.

Vertical distribution is defined as the process of distributing data, processing and information vertically to heterogeneous nodes of differing processing ability and energy consumption. In these networks the processing nodes get more computationally powerful as the nodes get further away from the direct source of the information. These forms of networks are specifically important for Body Sensor Networks as low power wearable nodes are often computationally weak, while computationally powerful clusters are not easily wearable or physically close to data sources.

These forms of BSN distribution are attempted to be realised in hardware and software, and comparisons are drawn in terms of energy efficiency and processing ability. This work attempts to enable these forms of networks to be realised for long term monitoring, and thus this is also an important research aim.

This work investigates the dimensions of distribution by developing a set of tools for Body Sensor Networks and explore the physical, software and hardware elements of these networks.

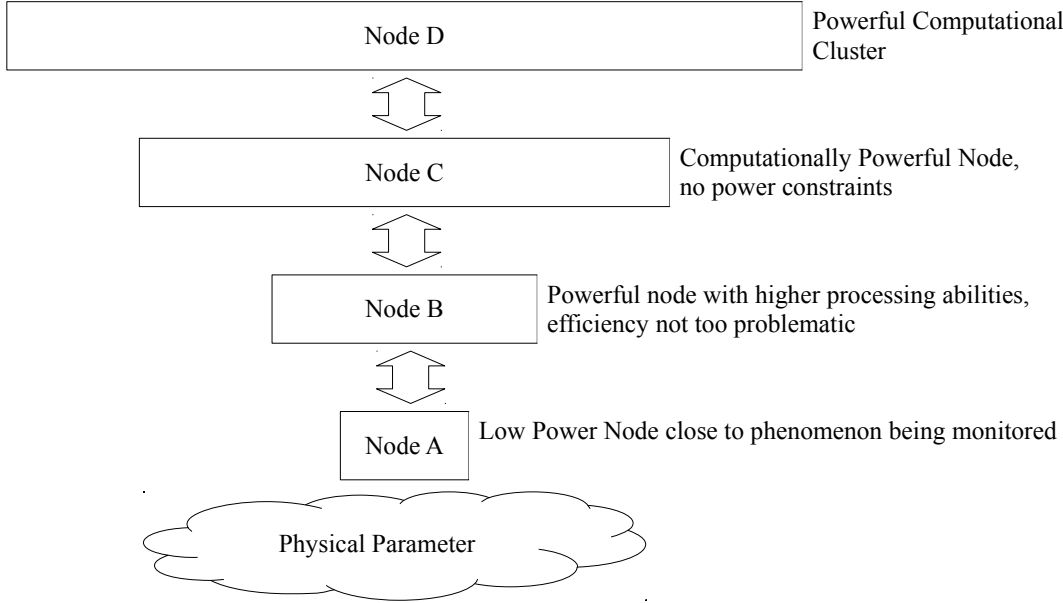


Figure 1.4: Vertical Distribution

### 1.3 Prototype Horizontal and Vertical BSN Platforms

To understand the pros and cons of these platforms, two prototype platforms are developed which implement horizontal and vertical distribution.

The horizontal platform is intended to be a long term distributed network platform to enable research of different sensors, sensor interfaces and software frameworks. This platform should be able to monitor multiple physiological parameters off of a patient and collaboratively process the information in the network.

Due to horizontal platforms being so close to the physical parameter being monitored, utmost care in low power design is required. Design of sensors, sensor interfaces, processing of captured data and forwarding of only the most important information are important design goals for these forms of platforms.

The prototype vertical platform forms part of a research project called DejaView [13] which aims to provide close to realtime memory assistance from analysing images captured in the on-body sensor network. This prototype system requires low power sensors, a high-resolution camera sensor and rapid-turnaround intensive processing of the images to enable the real-time assistance.

### 1.4 Scope and Contribution

In this work, several enabling technologies are addressed to achieve such a system in a manner similar to a toolbox. Using this toolbox of technologies, we develop two hardware

platforms to demonstrate horizontally and vertically distributed Body Sensor Networks.

This work consists of the following novel and unique contributions divided into three sections:

## Developing Low Power Sensors and Sensor Interfaces

**Capacitive Respiratory Sensor** In this work, we present a novel capacitive respiratory sensor. The sensor is a low power sensor, which can display a simple plethysmography profile and respiratory rate information of a patient.

### **An Efficient Camera Interface for Computationally Limited Microcontrollers**

A energy efficient 5Megapixel camera interface allowing low power microcontrollers access to a range of camera sensors which has been tested to up to 5Megapixels of image data.

## Software Frameworks supporting Distributed Sensor Networks

### **Information Valuation and Costing System for Low Power Microcontrollers**

An information valuation and costing framework which enables horizontally distributed nodes to evaluate the novelty of their captured information before transmission into the network.

### **Node Framework Architecture for multiple hardware platforms**

An implementation of a framework allowing multiple hardware architectures to run the same code in a way that is energy efficient and simple to understand is presented. This contribution comprises of the architecture as well as the implementation of the architecture.

**Tiered Vertical Architecture for Processing** In this work a vertical architecture design for distributing processing in Body Sensor Networks is presented.

## Prototype Horizontal and Vertical Body Sensor Network Platforms

**A prototype horizontally distributed platform for data capture** A prototype horizontal platform for data capture is realised and implemented in this section using the Information Valuation and Costing and the Node Framework Architecture. This system then demonstrates the a proof-of-concept and initial prototype of this platform.

**A prototype Platform for processing** A realisation of the tiered vertical architecture using the camera interface is presented.

Body Sensor Network research contains several other critical enabling technologies which are not presented in this work.

From Figure 1.2, the section on Informing or feedback to the end users is not explored in this work. This is an important part of the network, as it is critical to achieving network usage and is shown later to guide the sensing and processing tasks, but is not explored here and is left to future work.

## 1.5 Document Structure

This work is divided into three different sections: Developing Low Power Sensors and Sensor Interfaces; Software Frameworks supporting Distributed Sensor Networks; and Prototype Horizontal and Vertical Body Sensor Network Platforms. In these sections, we investigate forms of distribution in these areas related to distributed Body Sensor Networks and attempt to define a system of analysis to unite an approach to solving these problems. Figure 1.5 shows an overview of the thesis sections.

In Chapter 3, we design and investigate a sensor and sensor interface: a physiological measurement device for respiration rate (also known as plethysmography) and a worn on-body camera interface to monitor a person's interactions with the outside world, called Dejaview.

In the following chapter, we develop an information valuation and costing approach along with specific value and cost models allowing us to distribute information around the network. Using this system, we value and cost the information before transmission, and thus enable the network to distributively analyse the information in-network. Having developed a way of valuing and costing the information, we investigate how the information is communicated around the network.

Differing from normally homogenous wireless sensor networks, BSNs are often required to be heterogenous due to their vast and differing processing requirements as shown in Chapter 3. We develop and analyse the Simple Wireless Network Framework (SWNF) and show how it can run on vastly heterogenous platforms, while still maintaining a simple, understandable framework. The final section of Chapter 4 describes a vertically distributed architecture for BSNs.

The final section comprises two different prototype BSN platforms which we developed to test the sensors, architectures and protocols developed in Chapters 3 and 4. In the first platform, we also develop an array of low power healthcare sensors for on-body

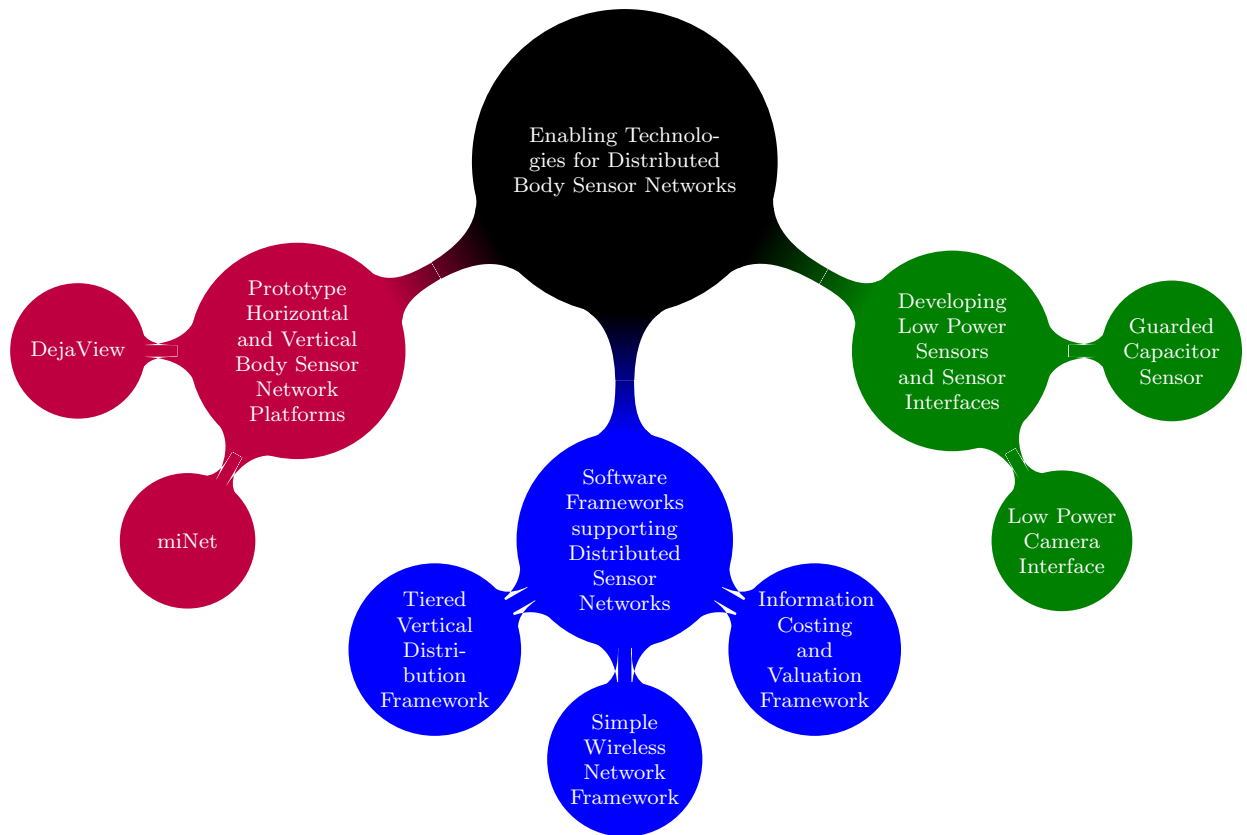


Figure 1.5: Mindmap of Thesis Outline

monitoring. We then demonstrate that these systems work with initial examples of their functioning.

Conclusions and Future work is presented in Chapter 6.



## Chapter 2

# Body Sensor Network Platforms for Healthcare

This chapter looks at the past research in relation to BSN platforms specifically for healthcare applications. The initial systems considered show the previous technological approaches and thinking towards the problems associated with BSN technology. These approaches typically address the "last mile problem". The following platforms then considered in this chapter show the associated approaches and problems addressed in the current thinking of BSNs. The individual platforms described in this chapter address some of the problems shown in Section 1.1, but are not designed to address the same set of aims as in Section 1.2 and thus not ultimately suitable.

Later chapters in this work contain specific research related to those chapters and this section should thus not be regarded as a complete literature survey, but rather an initial background review of BSN platforms related to healthcare.

A Body Area Network (BAN)<sup>1</sup>, or Body Sensor Network(BSN) is a network of small wireless sensor nodes which collect data about the physiology of a human being. This type of wireless sensor network is very different from many other Wireless Sensor Network Environments. The diverse requirements from numerous standardised sensing devices and systems, along with harsh physical restrictions of the devices are challenging problems which need to be addressed before any real-world success in this field is achievable.

Many different BAN, or BSN platforms have been demonstrated in the literature [14–18]. Many of these systems demonstrate the ability of a small wireless node to capture biological signals, and transmit the signal to a computer or laptop in close proximity known either as a gateway or a personal server, where data is first transmitted to the local gateway, which is directly connected to the internet, or in the case of the personal

---

<sup>1</sup>The term Wireless BAN or WBAN is the same type of network, but this research will refer to it simply as a BAN



server, the data is collected, captured and stored before being further transported onto the required destination.

There has been a lot of research over many years which implement wireless telemetry systems or devices aimed at collecting data wirelessly from the human body to a computer. These systems conquer the ‘last mile problem’ and attempt to remove motion artefacts developed from the extra weight of cables. These artefacts are created from the weight of the cable affecting the sensor contact with physical surface being monitored and appear as noise on the signal as shown in Figure 2.1. Systems vary greatly depending on whether they are implemented in the analog domain, or the digital domain, or a combination of both. Looking at the tables reproduced from [19] in tables 2.1 and 2.2, one can see that these systems vary tremendously with different complexities, sizes and abilities. These systems can be seen as initial or first generation BSNs, as they merely remove the use of cables, and do not consider multiple sensor nodes monitoring the same person in a distributed fashion.

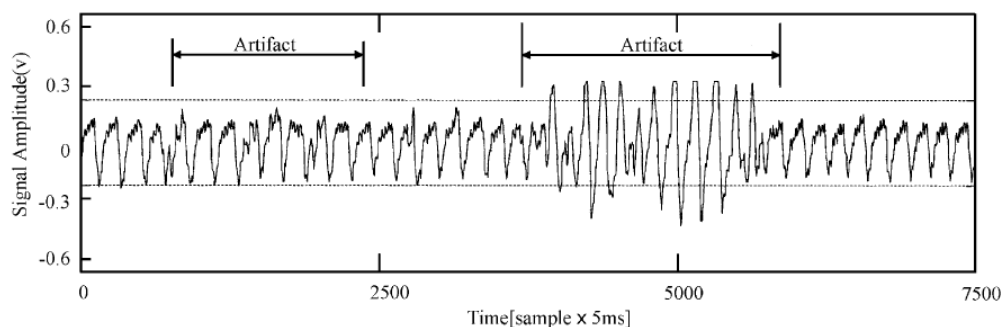


Figure 2.1: Example of time-limited motion artifacts on SPO2 signal taken from [20]

Table 2.1: Comparison of analog body area network systems reproduced from [19]

Institution	UCLA	Michigan	Aachen	Tokyo U.	UCLA	Clev. Med.
No. of Analog Channels	8	7	2	1	1	8
Telemetry Link Freq (MHz)	2400	94-98	88-108	80-90	3200	902-928
Modulation Scheme	DSSS/ O-QPSK	TDMA/Analog FM	FM Stereo	Analog FM	Analog FM	FSK
Power Supply	3V	+/-1.5V	+/- 1.4V	3V	Inductive	9 V
Max. Power Dissipation	97mW	-2.05mW	-	10mW	13.8mW	-
Transmission Range	9 meters	a few meters	a few meters	~ 16m	<1 meter	<46 m
Sensitivity	1 - 15mV	0.1 - 5mV	-	-	0.015 - 15 mV	-
Total Bandwidth	4 kHz	10 kHz	-	-	10 kHz	-
Dimensions (cm)	6.5 x 3.1 x 1.5	0.22 x 0.22	2.5 x 1 x 0.5	1.5 x 0.8	0.5 x 0.5 x 1	6.4 x 5.1 x 1
Total Weight (w/battery)	66 gr	> 1.1g	>3.1g	>0.1g	<1g	>68g
ATEF	0.00562	7.32 (1-gr batt)	*	*	0.72	*
Reference	[19]	[21]	[22]	[23]	[24]	[25]

\* Insufficient Data

Table 2.2: Comparison of digital body area network systems reproduced from [19]

Institution	UCLA	NASA	Kansas State	Lboro	Harvard	Duke
Number of Analog Channels	8	10	5	8	2	12
Communication Standard	Zigbee	Bluetooth	Bluetooth	Bluetooth	Zigbee	802.11b
Processor	8Mhz TI	7.36Mhz PIC	25Mhz PIC	40Mhz AMD	8Mhz TI	66Mhz AMD
On-board Memory (bytes)	1 M	32 M	2 M	256 k	1 M	
Power Supply	3 V	3 V	6 V	9 V	3 V	3.3 V and 5 V
Power Dissipation (streaming)	97mW	360mW	3.44 W	4.5 W	~ 100mW	4.0 W
Transmission Range	9 meters	10 meters	-	-	9m	9m
Signal Interpretation	Neural Spike	-	-	-	-	-
Total Bandwidth	4 Khz	~300 Hz	250 Hz	-	120 Hz	31.25 kHz
Dimensions (cm)	6.5 x 3.1 x 1.5	12.9 x 10 x 0.2	-	-	5.8 x 3.2	5.1 x 8.1 x 12.4
Total Weight (w/battery)	66 gr	166 gr	-	-	~66gr	235g
DTEF	0.06748	0.0006024	*	*	0.001309	0.04309
Reference	[19]	[26]	[27]	[28]	[29]	[30]

\* Insufficient Data

In the systems listed in tables 2.1 and 2.2 the authors created two parameters called the Analog Telemetry Efficiency Factor (ATEF) and the Digital Telemetry Efficiency Factor (DTEF) to estimate the overall value of the sensor systems. They are calculated as can be seen in Equation 2.1 and Equation 2.2. In these equations,  $BW$  represents the aggregate communications bandwidth (KHz),  $BR_{eff}$  is the effective communications bitrate taking into account compression,  $d$  is the maximum distance (meters)  $m$  is the mass of the sensor (grams) and  $P$  is the average power dissipation (milliwatts).

$$ATEF = \frac{BW * d}{m * P} \quad (2.1)$$

$$DTEF = \frac{BR_{eff} * d}{m * P} \quad (2.2)$$

A Body Sensor Network, as proposed by Guang-Zhong Yang [7], aims at a network of implantable and wearable devices called BSN Nodes, which capture individual signals, perform low-level processing and then transmit their data to a Local Processing Unit (LPU), where the data is processed further and fused together to be transmitted along to a central monitoring server through numerous different ‘complex’ wireless media, such as WiFi, WiMAX, Bluetooth, GPRS, GSM. This is illustrated in Figure 2.2. This architecture uses a star topology where the LPU is regarded as the coordinator of the network such as in Figure 2.3.



Figure 2.2: Body Sensor Network Overview taken from [7]

The differences between the original telemetry systems and a modern concept of a BSN, relate more to the establishment of a set of standards and middleware which allow for

efficient, transparent and pervasive networks of devices working together to monitor the human body. As there are multitudes of devices used to monitor different physiological parameters, along with the very different types of healthcare providers needing to analyse the data in different and varying ways, creating a single interfacing platform is a challenging and difficult task. There have been several large research groups working on BSNs and BANs, and recently the IEEE have started to ratify a new BAN Standard as part of the IEEE 802.15 low power wireless standards [31]. The following section gives an overview of the current platforms and standards in the literature.

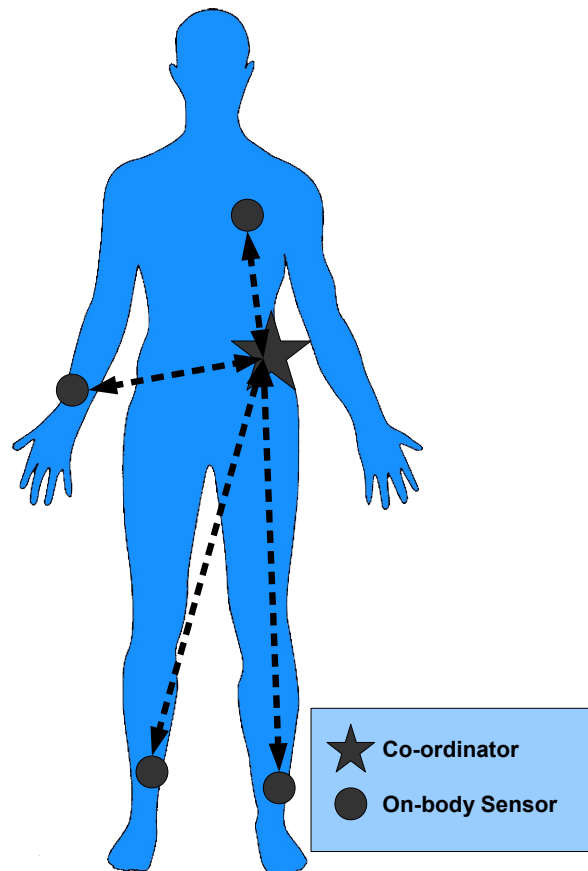


Figure 2.3: Star topology BSN

## 2.1 The Human++ Research Program

The Human++ research program [14, 32, 33], which is run by IMEC in Belgium, aims to create and deploy highly miniaturised and autonomous sensor networks which are carried around by an individual human: a Body Area Network. Their interpretation of a BAN provides medical, lifestyle, assisted living, sporting and entertainment functionality to the person concerned. To enable this BAN, they are using techniques from experience in ultra-low power communications, Microelectromechanical systems (MEMS) energy scavenging, low power design and packaging to overcome what they believe are the largest current obstacles. Obstacles to achieving this goal include:

**Size** To be able to transport the devices around the human body, the devices need to be of a size where they are not intrusive to the human.

**Energy Autonomy** Current battery energy density does not provide enough energy to be used for long periods of time, and improvements are required.

**Network** The interaction between sensing devices and devices/end-users which perform actions from the sensed values need to be improved thereby creating “closed loop disease management systems” [33].

**Intelligence** The devices need intelligence to “store, process and transfer data”.

Currently, several prototypes and proof-of-concept systems have emerged from the Human++ project, most notably a pulse oximeter which is powered only by a watch-like thermoelectric generator [34] which can be seen in Figure 2.4. This thermo-electric generator is both small, and can gather around  $30\mu\text{W}$  of energy per square centimeter of contact with human skin, while the low power pulse oximeter, uses  $89\mu\text{W}$  for a 15 second measurement interval.

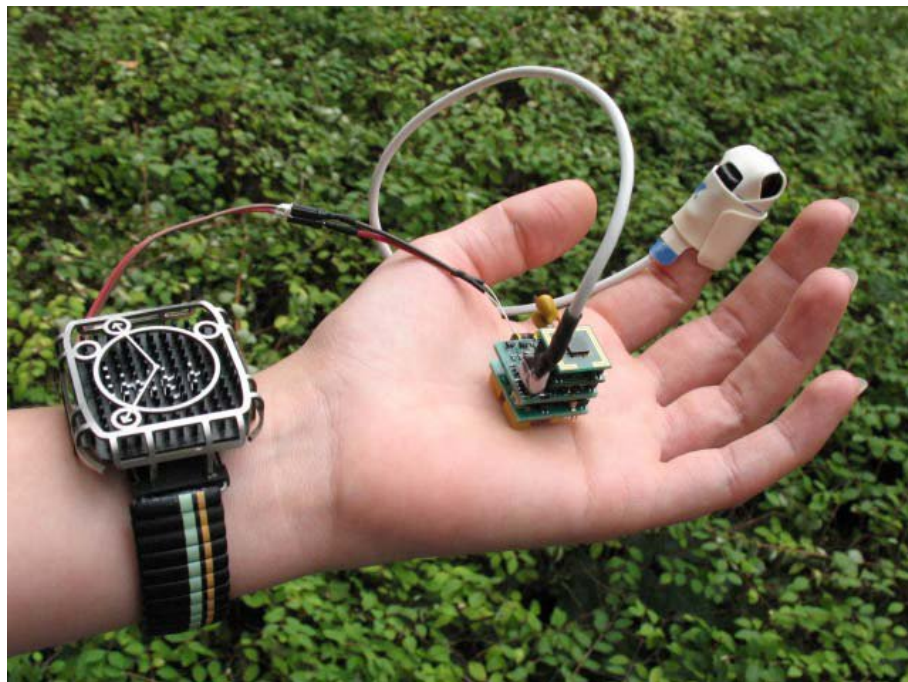


Figure 2.4: Thermoelectric watch (left), wireless radio and analog circuitry (middle), and COTS finger SpO<sub>2</sub> meter (right) taken from [34]

The Human++ project has also developed a prototype of an intelligent plaster which contains a microcontroller, radio and antenna. The device is optimised for transmission very close to the body. Figure 2.5 shows an image of the device. This device is currently being developed to include low power sensors and an efficient energy system.

The Human++ project believes that the current wireless technology standards of Bluetooth (IEEE 802.15.1) [35,36], Zigbee (IEEE 802.15.4) [37,38], proprietary radios (such

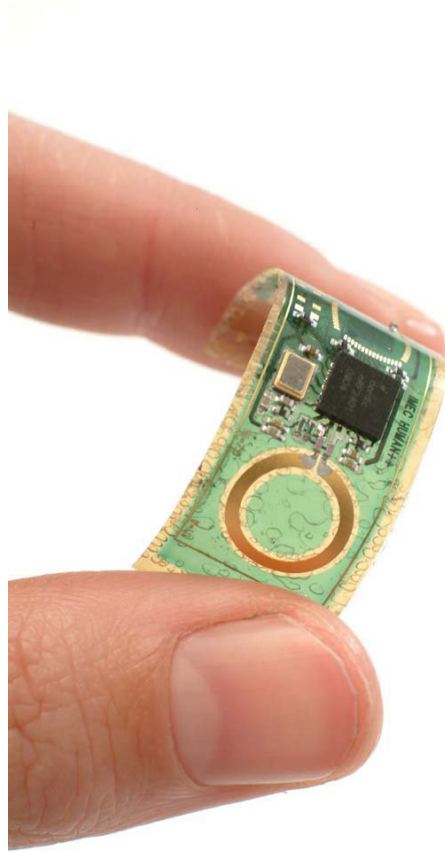


Figure 2.5: Intelligent Plaster with radio and antenna showing. Reproduced from [32]

as the Nordic [39] and Texas Instruments [40]) are still too power hungry and energy inefficient to meet the requirements of a BAN. Current radios which have a power efficiency of 100-1000nJ/bit [32] are still regarded as an order or two of magnitude too high. Due to this, they are investigating the use of low power Ultra-Wide Band (UWB) communication in the 3.1GHz and 10.6GHz range. Using pulse-based UWB, the research aims at making low complexity, low power transmitters, as the receiver has to deal with the complexity overheads in UWB, which the LPU can provide as it has a less stringent energy budget.

## 2.2 The Smart and Aware Pervasive Healthcare Environments Project

The Smart and Aware Pervasive Healthcare Environments (SAPHE) project [41] is a project led by Imperial College London, with partners BT, Philips, Cardionetics, Docobo and the University of Dundee. SAPHE's goal is to develop a new generation of "pervasive healthcare and lifestyle monitoring systems to allow for early detection and prevention of acute events" [42].

The SAPHE Architecture consists of three core components:

**Ambient Sensing** The Ambient Sensing component is a network of non-invasive sensors which track and follow patients through their homes and natural environments. The sensors capture patient information on activity, sleeping patterns, gait and posture.

**Wearable Sensing** The wearable sensors are a group of on-body sensors which measure information about important physiological readings depending on the patient's conditions. The SAPHE project does not intend to provide a full range of different wearable sensors, but rather a sensor node architecture for OEM suppliers to be able to easily connect devices to.

**Interface/Decision Support** SAPHE uses a “distributed inferencing paradigm” [42] which consists of a two-tiered approach. Tier 1 comprises of either a home gateway node which captures and coordinates all information from the Ambient Sensing and Wearable Sensing networks - or a portable device fulfilling the same role for a highly mobile patient. Tier 2 comprises of remote data servers which capture all information from the remote home gateways and collect individual long-term trend information and pooled population data.

The SAPHE Project finished in early 2009. This project combined both a body sensor network, based on UBIMON [8] with a pervasive sensing environment. UBIMON is a simple star topology network using a Personal Digital Assistant as an LPU.

## 2.3 BioMobius

BioMOBIUS is a recently released research platform developed by the Technology Research for Independent Living Centre [43]. Designed as a rapid prototyping tool for biomedical research, BioMOBIUS is a platform comprising of hardware (Computer, Sensors) and Software (Applications and Services). The BioMOBIUS hardware elements incorporate the use of the SHIMMER mote platform for Wireless Sensing [44] as well as Tactex sensors. BioMOBIUS is freely available for research use [45]. The SHIMMER mote platform is a mote platform developed by Intel, and currently available for purchase in the UK via a third party organisation called Shimmer-Research.

The BioMOBIUS architecture is built upon the EyesWeb Project [16] which was created by the University of Genoa, and therefore many of the components and paradigms are the same or similar. BioMOBIUS uses the ideas of *Blocks* and *Patches*.

A Block is the core functional element which has one or more inputs and outputs, data processing and parameters which can be adjusted during run-time via a Graphical User



Interface (GUI). Blocks are created by both BioMOBIUS developers and application developers. These blocks incorporate low-level functional components such as interfaces to hardware, data-processing and specific graphical interfaces.

A Patch comprises of a number of blocks in a flow-chart format. A patch is created by a high-level user such as a clinician or researcher for use by therapists for analysing patient data. Patch creators use a GUI to drop down pre-created blocks in the order they want it to work to perform high-level patient analysis.

Due to BioMOBIUS being built upon the EyesWeb Project, it is orientated around the use of cameras analysing human position and posture, thereby geared towards muscle and skeletal ailments, problems and disorders. This system is therefore not directly able to monitor for continuous periods of time, but rather to diagnose for shorter periods of time.

## 2.4 CodeBlue

The CodeBlue Architecture [15] is a middleware software platform for medical sensor networks based on an earlier TinyOS version 1 [46]. The node architecture can be seen in Figure 2.6.

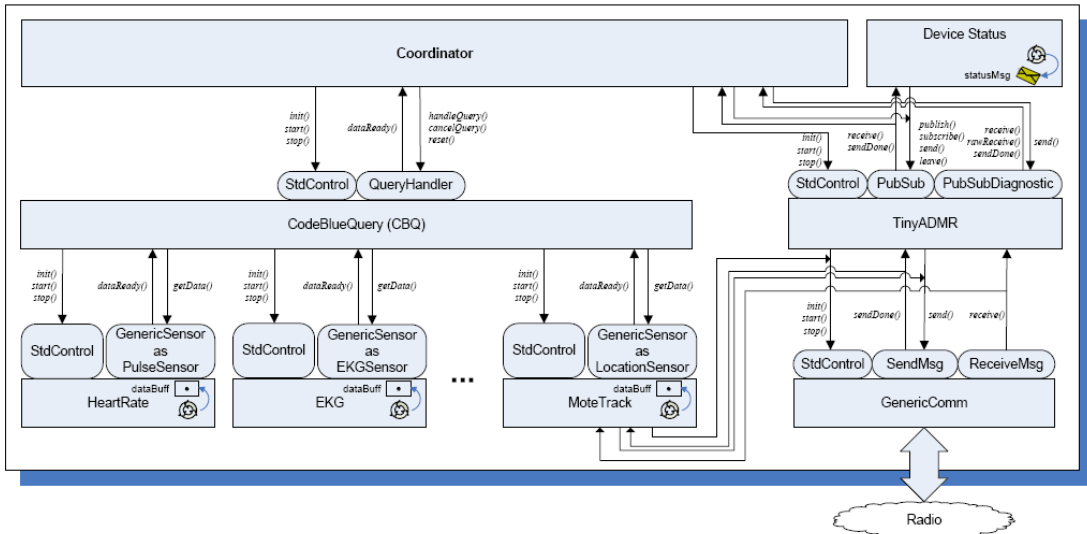


Figure 2.6: CodeBlue Architecture [15]

There are four main components of the CodeBlue Architecture: the Publish/Subscribe Routing layer, the Discovery Protocol, the Query Interface and Location tracking. These can be seen in Figure 2.6.

The Publish/Subscribe messaging paradigm allows all the sensors to publish which information (Topics) they have available, while other devices can then subscribe to that

information source and the data will be routed from the sensor to all the parties which are interested in that information source. This method is well suited to the Medical environment as many different medical end-users (such as doctors, pharmacists, nurses) require different information sets from different patients. The publish/subscribe routing protocol is based on Adaptive Demand-Driven Multicast Routing (ADMR) protocol [47].

The Discovery Protocol allows the individual sensors to broadcast information about its individual state, sensors connected and other meta-data. This information does not change continuously, and therefore is not sent often ( $\sim 30$  second intervals).

The CodeBlue Query (CBQ) layer allows end-point devices to request the sensor nodes to communicate information along a pathway relating to the instruction of a specific CBQ query. There is no textual input language for CBQ, as the information contained in the query is represented in the tuple  $\langle S, \tau, chan, \rho, C, p \rangle$ . Where  $S$  is a set of node ID's which should transmit data,  $\tau$  represents which sensor attached to the node should send information,  $chan$  is the ADMR channel which should be used to transmit the information,  $\rho$  is the sampling rate which the device should sample the sensors data at,  $C$  is the number of samples which should be recorded and  $p$  is a filter predicate.  $p$  allows simple on node processing of the signal, which allows the device to only send information if it matches the filter conditions. The filter only allows for comparison of two variables in  $p$ . So for example, if a doctor would like to request heart-rate information (sampled at  $45Hz$ ) about three patients in a ward (connected to devices 6, 7, 9) when their heart-rates fall below 45 or rise above 180, which he will be listening on ADMR channel 23, the tuple will look as follows:  $\langle \{6, 7, 8\}, HR, 23, 45, \infty, (HR < 45)OR(HR > 180) \rangle$ .

The final architecture component of the CodeBlue project is the decentralised localisation system called MoteTrack. Motetrack implements a distributed localisation algorithm which gives accuracy of approximately 1 metre. Motetrack uses a scheme whereby a device which requires localisation information, listens to a set of fixed beacon nodes, and acquires Received Signal Strength Indication (RSSI) from the beacon nodes. The RSSI Information is then compared to a distributed database which is replicated across all of the beacon nodes.

## 2.5 Discussion

Initial prototypes of Body Sensor Networks have been realised. These systems have all focused attention on a single stakeholder within the system: the patient. Most of these systems, apart from CodeBlue, use a centralised concept of either a LPU or other device to combine the information before being transmitted upstream to a hospital or caregiver.

The core focus of the Human++ program is investigating techniques and tools to enable BSNs, while the core focus of SAPHE is to create tools to enable care for the elderly.

BioMobius appears to be a tool for biomedical researchers to explore movement and gestures. CodeBlue is a distributed network system which does address some of the requirements of multiple shareholders/caregivers, but does not have any feedback system to the patients themselves and is intended for use within a hospital.

To enable selective and indicated preventative and self-managed care, assistive tools are required to give all stakeholders access to the patient's individual information. A distributed BSN is such a platform which can assist not just an individual stakeholder, but multiple stakeholders with any level of abstraction. Such a system would require distribution of information to allow for different stakeholders' data interests, and collaboration of different sensors for solving different tasks within the system. Current BSNs which have been realised in the literature do not satisfy these conditions and we therefore propose to develop a system which is described in Section 1.1.

In the Human++ Research Program, four key obstacles have been identified to realising BSNs: Size, Energy Autonomy, Network and Intelligence. To ensure small size and energy autonomy efficient use of the energy available to the node is imperative. The nodes thus needs to have small and energy-efficient sensors which are further investigated in Chapter 3 and energy and resource efficient processing which is further investigated in Section 4.1.2. The "Network" obstacle describing the behaviour between nodes and end-users to ensure that the system behaves as a closed-loop system is addressed in Chapter 4 with inter-node behaviour being investigated in Section 4.1.2 and an architecture for end-user interaction being shown in Section 4.3. This architecture is then also initially implemented in Section 5.2. Intelligence as defined by the Human++ Project is addressed in Sections 4.1.2 and 4.2 whereby a dissemination protocol and a software platform is presented respectively.

The SAPHE project highlights an architecture consisting of three core components to address the problem of home monitoring. The critical difference between the system and this work is that of the intelligent environment, and not monitoring the patient at the same level of care outside this intelligent environment. Ambient sensing is thus not further considered in this work. Wearable sensing is also shown to be of utmost importance in the SAPHE project, and similar work to ensure sensor interfaces are standardised is shown in Section 4.2 while physical sensors and sensor interfaces are investigated in Chapter 3. The third important component of the SAPHE project: Interface/Decision Support, is of importance to Section 4.3 of this work as the system attempts to achieve similar functionality.

The BioMobius project highlights the importance of simple tools for developers and defined architectures to enable rapid prototyping of BSN systems. This work shows how structured software components such as the system shown in Section 4.2 is important for rapid development of BSNs.

The CodeBlue architecture is horizontal distribution platform providing four main services: data distribution via the Publish/Subscribe routing layer, data discovery via the Discovery Protocol, Further data distribution and node intelligence via the CBQ layer and Location Tracking using MoteTrack. In this work, the Publish/Subscribe routing layer and CBQ layer use ADMR which is comparable to the work presented in Section 4.1.2. The Discovery and MoteTrack protocols are not deemed important in this work and are not further investigated. The CodeBlue project does provide a comparable horizontal platform to the miNet platform presented in Section 5.1.

This chapter has provided a background to comparable BSN platforms while highlighting common obstacles to realising BSNs. In this work, we investigate these common obstacles: Energy Efficient on-body sensing; distributed communication protocols; software frameworks which enable these technologies and architectures to close the loop to the end-user. The following chapter investigates Energy Efficient on-body sensing, while Chapter 4 investigates the communication protocols and software frameworks.



## Chapter 3

# Developing Low Power Sensors and Sensor Interfaces

In this Chapter a low power sensor and interface to be used in BSNs are investigated and developed. Low Power Sensing is imperative for long term use in BSNs and has been shown in Chapter 2 to be a common goal of similar platforms. Size of the sensing device is also of utmost importance to the design, and ensuring that the sensor is not cumbersome or intrusive are further design goals for BSN sensors.

Developing sensors for BSNs requires an investigation into the sensor size, energy consumption and data output. A balance is then required to ensure high data quality with low data quantity, low energy usage and small size.

To demonstrate the design and development cycle of sensors for BSNs, two example designs are shown in this chapter. The first example is that of a respiratory sensor for continuous monitoring. Current respiratory monitoring sensors are cumbersome due to the requirement of a chest strap which surrounds the patient, and reasonably energy inefficient due to the associated complexity in measuring inductance.

The second design and development example is of a low power sensing interface for a camera sensor. Camera sensor interfaces are normally part of energy hungry Digital Signal Processors (DSPs) which are able to sufficiently process the large quantities of data provided by the sensor. Modern energy efficient Microcontrollers (MCUs) save energy by not using a high speed clock and are thus inherently not fast enough to process all of the incoming data. By investigating and developing a dedicated high speed, low power interface for the MCU, it is possible to use the sensor for long term monitoring applications by enabling the sensor in short bursts, capturing the data and powering the sensor down to be operated again at a later stage.

### 3.1 Guarded Capacitor Sensor

There are currently several different systems of monitoring respiratory rate, known as plethysmography. The most common is called Respiratory inductance plethysmography (RIP). The most common units given for this measurement is in breaths per minute. We wish to develop a sensor which gives a similar type of reading, without the need for a complete chest strap which is required for RIP, but rather a patch of some sort.

RIP measures the changes in chest cavity size [48,49]. RIP is currently the gold standard in measuring plethographic signals.

As RIP uses a chest strap which is required to stretch around the whole patient, it can encumber a patient in certain situations. We wish to develop a patch-like sensor for respiratory monitoring which is low power for long term use. An investigation into Capacitive sensors has thus been done.

Guarded Capacitive Sensors have been used in numerous biomedical applications including ECG [50], EEG [51,52] and EMG signals [53]. These systems generally use the capacitive plates as a capacitive electrode.

In this work, a guarded capacitive sensor is used not as an electrode, but to detect movement of the skin surface of the chest as the chest exhales and inhales. Using capacitive sensing allows the physical parameter to be sensed energy efficiently and with small size and low component count. This further assists the sensor in being non-invasive and is thus suited for long-term use.

#### 3.1.1 Guarded Capacitive Sensor

Capacitive sensing has become a popular sensing technology for its low power and non-direct sensing ability. These sensors are prolifically found in modern mobile handset touch-screens and are used by NASA [54] as a distance sensitive skin called a capaciflector. The principle behind the directed capacitance field can be seen in Figures 3.1 and 3.2.

A Normal parallel plate capacitor which is seen in Figure 3.1 has the equation for Capacitance  $C$  given in Equation 3.1.

$$C = \epsilon_r \epsilon_0 \frac{S}{d} \quad (3.1)$$

where the surface  $S$  is the width  $X$  times the length  $Y$  and the distance  $d$  is the distance between the two plates. The relative permittivity,  $\epsilon_r$ , is the relative permittivity of the material between the two plates while  $\epsilon_0$  is the electric constant approximated as  $8.854 * 10^{-12} Fm^{-1}$ .

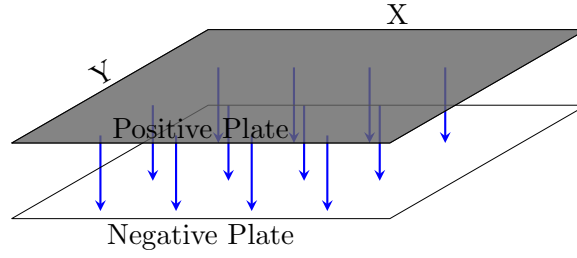


Figure 3.1: Parallel Plate Capacitor

If one places a new plate between the two parallel plates, and drives this plate with the same voltage as the top plate, the electric field developed between the two plates has to travel around the driven plate as shown in Figure 3.2. The driven plate, known as a *guard* is driven by a voltage buffer or such to achieve the voltage guard. By creating a guard, the field is forced to interact with the environment outside of the plate, and thus any material with different permittivity will change the overall capacitance and thus form a guarded capacitive sensor.

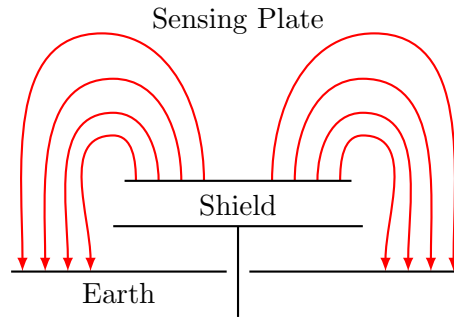


Figure 3.2: Guarded Capacitor Sensor

From the equation given in 3.1, for a large capacitance  $C$ , one needs to ensure both  $d$  is small and  $S$  is as large as possible. Because this sensor is forcing the field away from the sensing plate, it is not directly clear how the field is affected by changes in permittivity of objects interacting with the field, or how  $S$  and  $d$  change the capacitance of the sensor. It is believed that as the electric field lines are forced away from the opposite plate, we can direct the field lines further towards the normal of the plate by increasing the surface area of the shield. It is however clear that the sensor will change capacitance as objects begin to interact with the electric field, and thus this sensor can be used as a distance measuring tool shown in [54, 55].

Using such a sensor close to the skin surface at the chest allows one to measure breathing by measuring the change of skin tension between the fixed capacitive sensor and the skin, shown in Figure 3.3. Although we can use the sensor to translate the physical parameter into a changing electric measurement, we still need to measure the change in the value of capacitance.



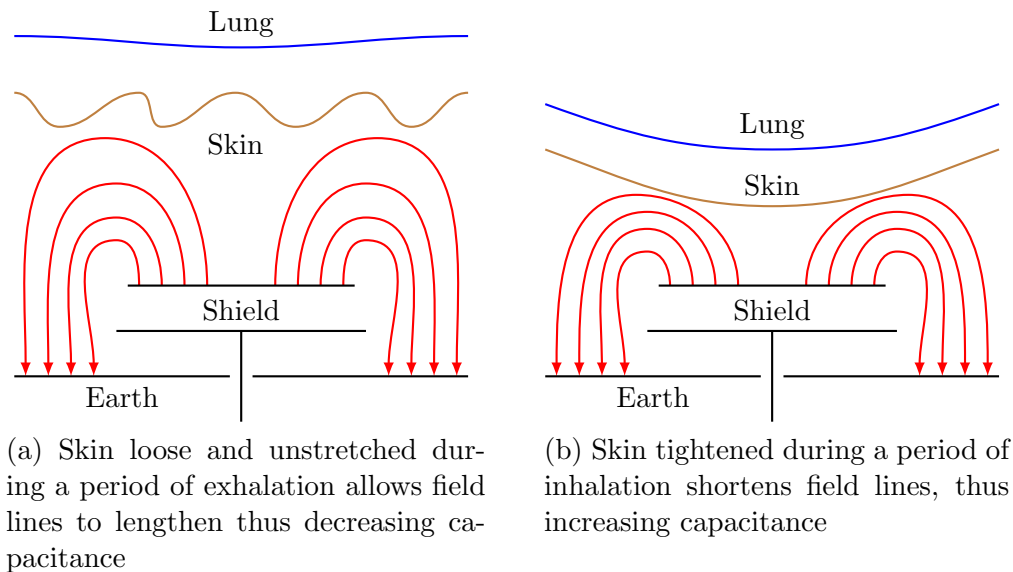


Figure 3.3: Two states are shown for the sensor during inhalation and exhalation. The effect of the inhalation and exhalation on the sensor are shown

### 3.1.2 Measuring the Capacitance Change

There are several methods to measure the change in capacitance. We would like to use the method which uses the least power as we intend to use the sensor in a continuous mode of operation on body. Of the several methods used to measure capacitance change, we have selected oscillator based capacitive sensing.

A relaxation oscillator implemented using a Schmitt Trigger can be seen in Figure 3.4, the frequency of oscillation  $f$  is given by the equation 3.2. By fixing  $R$ , the frequency  $f$  changes with inverse proportion to  $C$  as shown in Figure 3.5 for a feedback resistor of  $1\text{M}\Omega$  and capacitance around  $10\text{pF}$ .

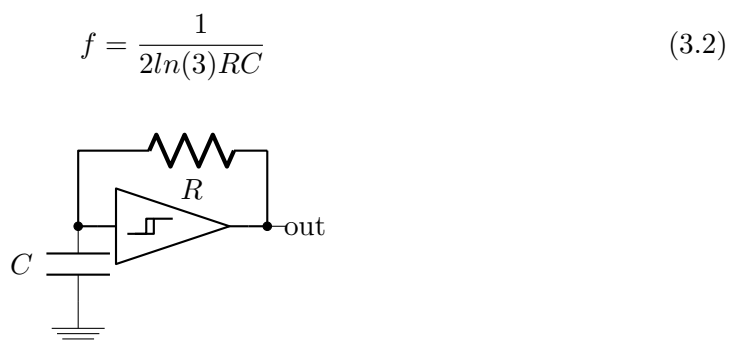


Figure 3.4: Relaxation Oscillator

The output of this sensor is a pulse train with frequency relative to the sense capacitor  $C$ . By measuring the frequency one can detect the capacitance of the capacitor. This interface is ideal for a microcontroller without an Analog to Digital Converter, and can

thus be interfaced directly to a Digital IO pin and counted against a timed clock period. By measuring the number of incoming clocks against the time period, one can get a measurement of the capacitance.

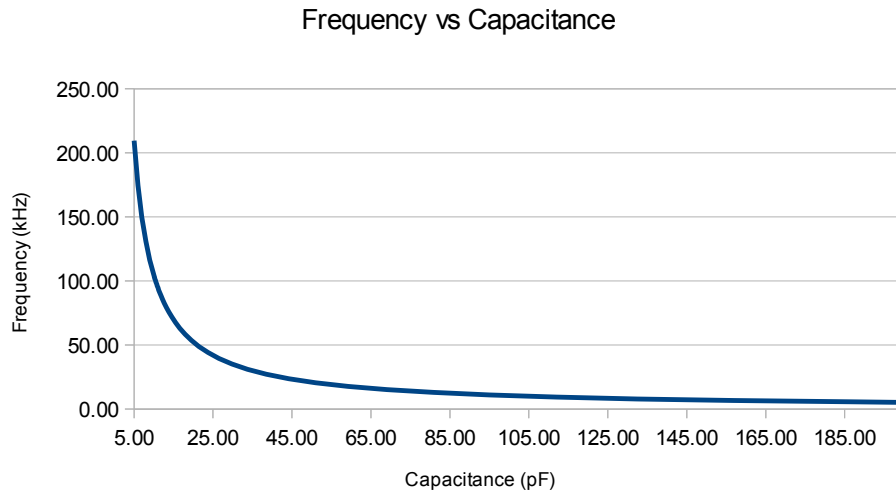


Figure 3.5: Theoretical Plot of Frequency vs Capacitance

### 3.1.3 Prototype Sensor System

A realisation of this circuit using two op-amps for a single-supply configuration can be seen in Figure 3.6. *Op1* is the Voltage follower for the shield plate. *Op2* is a comparator setup to act as a Schmitt Trigger. Values selected for  $R_1$  and  $R_2$  change the oscillation mid-point which the signal oscillates around, and are generally set to the same values.  $R_3$  is the feedback resistor of the relaxation oscillator, and  $R_4$  is the Schmitt Trigger gain.

For our prototype, we used two values for  $R_3$  and tested the circuit with a known capacitor value.

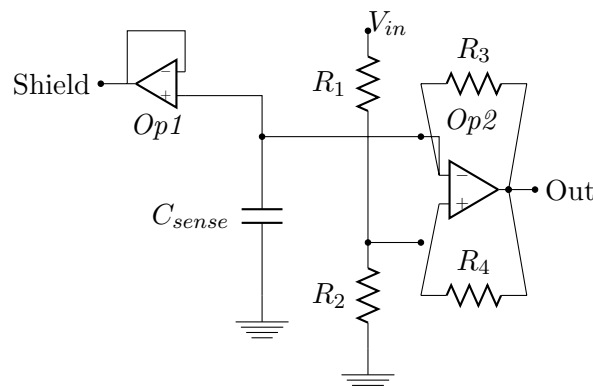


Figure 3.6: Capacitive Sensor

The output of the circuit in Figure 3.6 is fed into the input of an MSP430 RF2500T wireless sensor node. The node is programmed with a framework called SWNF and uses SHHP to communicate the data into a node network or PC for visualisation.

### 3.1.4 Results

To test the whole system, the relaxation oscillator was first tested to ensure it was working as had been expected. Then the sensor plates were tested, and finally the respiration of the patient being monitored was tested.

The relaxation oscillator was compared against a set of known capacitances to test the response of the relaxation oscillator for small capacitances. The results can be seen in Figure 3.7.

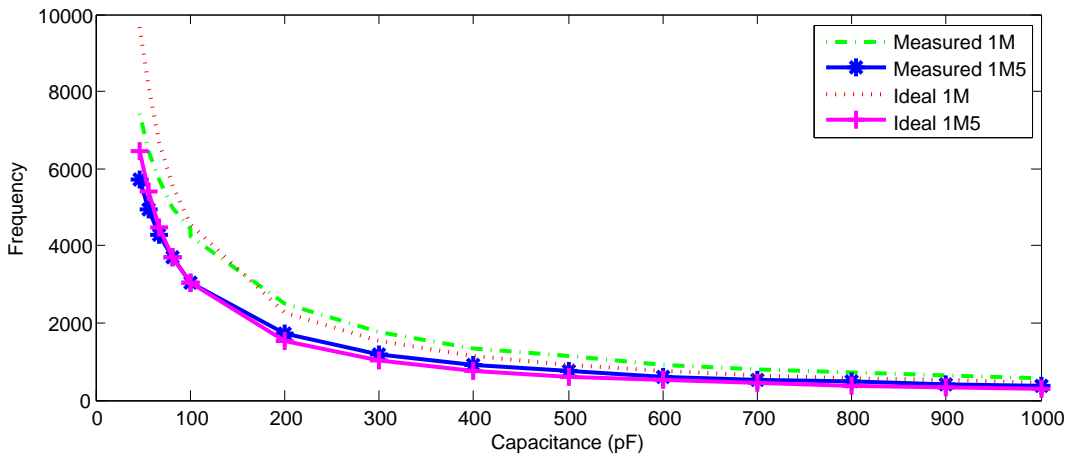


Figure 3.7: Relaxation Oscillator tests using fixed values of capacitance

In Figure 3.7 we compare two oscillators with different feedback resistors against a set of known capacitor values. The capacitor values were selected from a capacitive decade counter and a set of sub-100pF capacitor values. The figure shows that the relaxation oscillator follows the ideal reasonably well until the values go below around 50pF or 4.5KHz. At this point the measured values deviate considerably from the ideal. This is probably due to stray capacitance in the circuit. Although the capacitance deviates from the ideal, we can still sample points against the curve fitted to the measured results to obtain approximations of the capacitances.

Using equation 3.1, we investigate different sensor plate configurations to evaluate the usefulness of the sensor in terms of both distance and size of the sensor. Three configurations shown in Figure 3.8 are tested to measure the best performance.

To understand the optimal configuration of the sensor plates, we developed six different sensor plate configurations shown in Figure 3.9 and Table 3.1. First we tested their capacitance change using a free air capacitance and then placed a container filled with

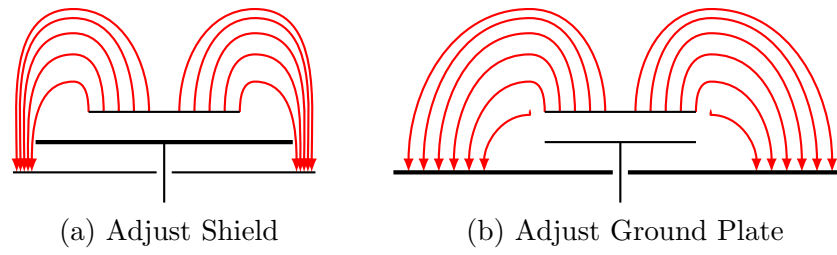


Figure 3.8: Testing Sensor Plate Configurations

salt-water on them to observe the total change. Salt-water was used as its permittivity closely approximates the human body [56] in comparison to air.

For the two non-three plate capacitors, we submersed a connector in the salt water mixture. This test demonstrates the difference in capacitance change of the plate sizes and evaluates the optimum plate configuration. The results of the test are shown in Table 3.2.

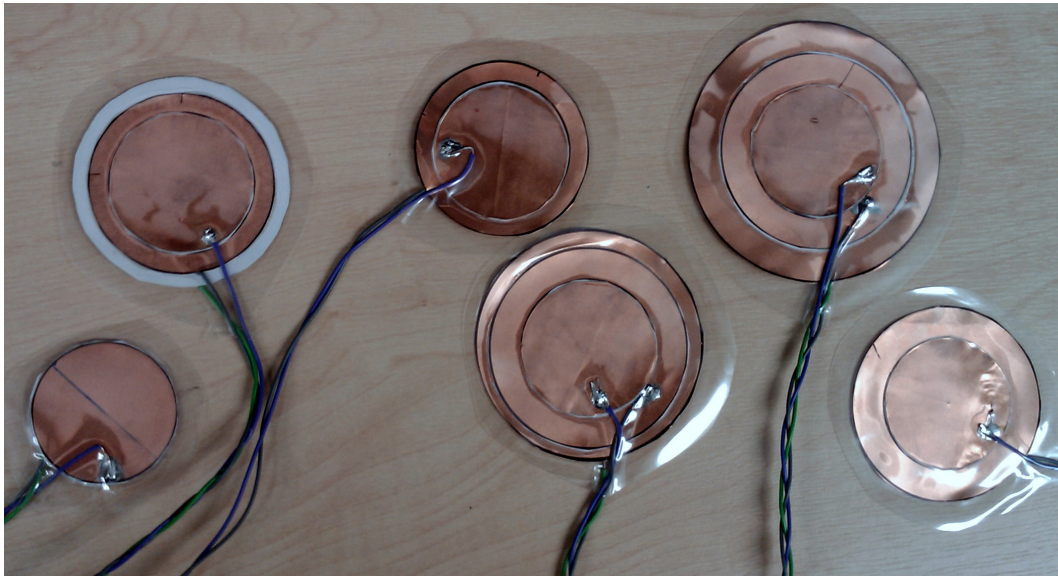


Figure 3.9: Capacitive Sensor Plate Designs. 1-6 follows from left to right

Table 3.1: Guarded Capacitive Sensor Plate Design and Configurations

Design	Plates	Radius size (mm)		
		Sense Plate	Guard Plate	Earth Plate
1	3	20	20	20
2	3	20	25	30
3	2	20	25	-
4	3	20	30	35
5	3	20	30	40
6	2	20	30	-

In Table 3.2 we used the data gathered in Figure 3.7 to fit the final capacitance values for the sensor plates in the two configurations to understand the change in capacitance value shown in Figure 3.10.

Table 3.2: Guarded Capacitive Sensor Plate Testing Results. These readings were gathered using an HP Oscilloscope model DSO3062A. The values were averaged over a period of 20-30 seconds for each design.

Design	1M0 (KHz)		1M5 (KHz)		Capacitance (pF)	
	Air	Water	Air	Water	Air	Water
1	15.15	12.93	12.66	10.73	17.06	21.25
2	16.32	14.73	14.35	12.04	14.92	17.98
3	20.89	13.70	19.07	13.07	10.39	18.04
4	15.29	13.90	13.52	11.64	16.25	19.15
5	14.57	12.84	12.58	10.42	17.62	21.76
6	21.74	15.16	20.01	12.85	9.79	16.89

Figure 3.10 shows a comparison of the frequency differences of the sensor configurations. From the figure, it is evident that configuration three and six exhibit the greatest difference in frequency and are therefore the best configurations to use in further respiratory tests. The differences between configurations three, six and the rest of the configurations are due to the grounding terminal changing dynamics of the system. Configuration three and six use a grounding terminal similar to an Electrocardiogram (ECG) which is attached to the human body via an electrode. Thus for best results, a single physical contact is required with the subject for respiration. A large noise signal was detected on one of the sensor plates when the device was not grounded to the body of fluid using configurations one, two, four and five, and almost eliminated when used with the sensors with a grounding terminal. This is due to the non-grounded device picking up stray electromagnetic signals.

Respiration testing was performed using controlled breathing of a test subject at rest. Figure 3.11 shows a result of a test taking five long and deep breaths.

To improve the signal to noise ratio, multiple relaxation oscillators and sensors were connected to the same node to record multiple signals simultaneously. This is shown in Figure 3.12.

Signal processing is required to capture the correct waveform from the data sequence. This is demonstrated in Figure 3.13.

### 3.1.5 Conclusions

This work presents a guarded capacitor sensor for respiratory measurements which acts as a low power sensor for continuous breathing monitoring. The sensor can be regarded

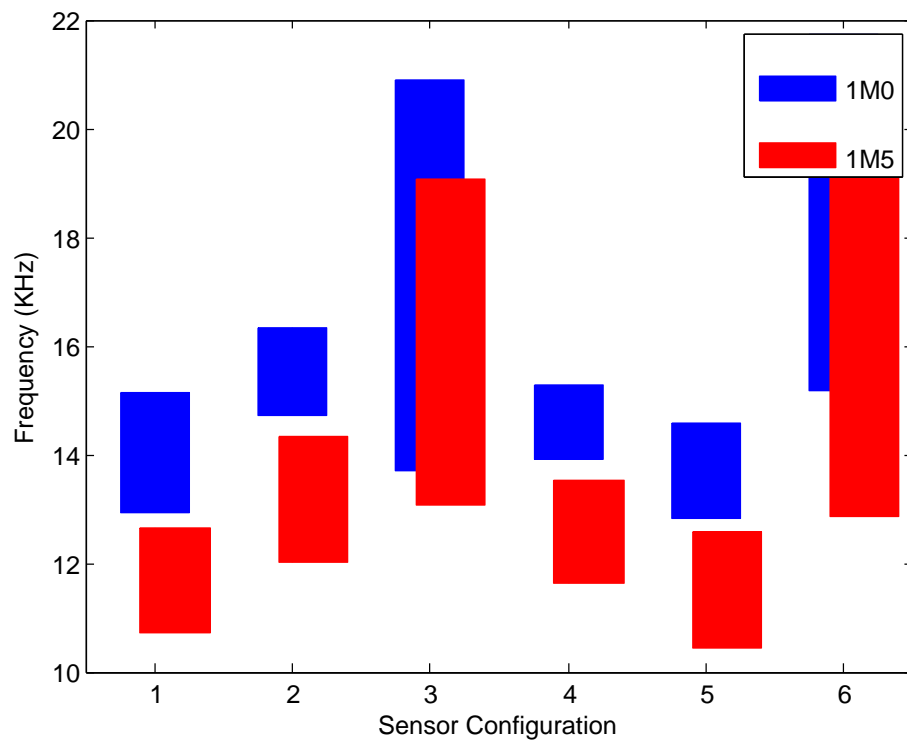


Figure 3.10: Capacitive Sensor Frequency ranges

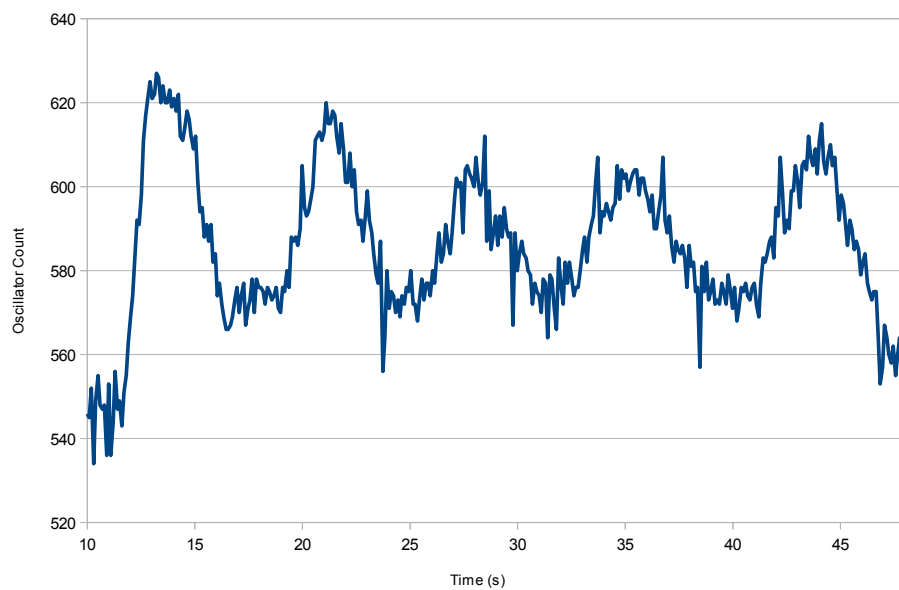


Figure 3.11: Result of 5 slow breaths taken using Sensor Design 3 with 1.5M Oscillator feedback resistor

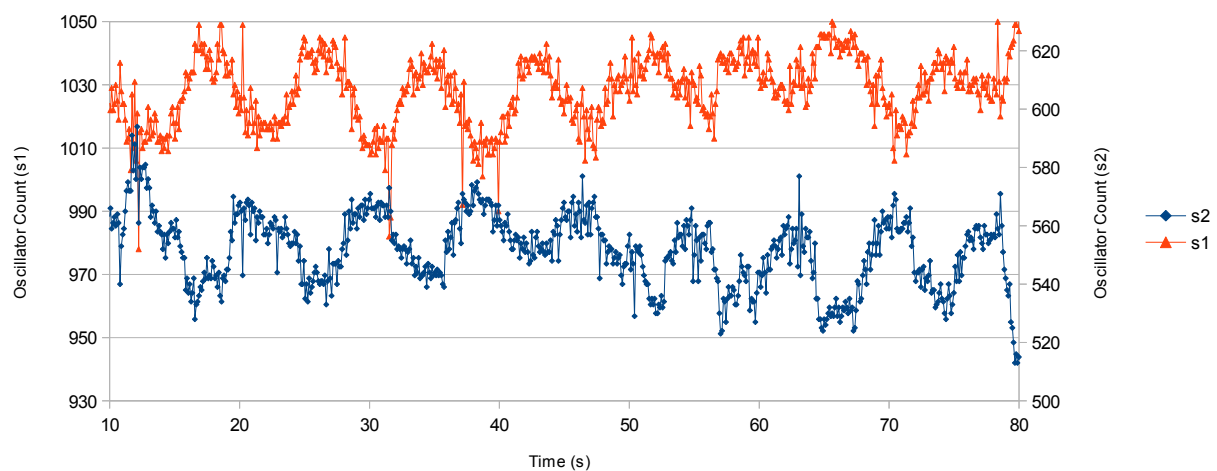


Figure 3.12: Result of 9 slow breaths taken using Sensor Design 3 & 6 with 1.5M & 1M feedback resistors. The signals are in anti-phase as some measured areas of the body, such as around the diaphragm between the ribs, force the sensor plate closer to the body during exhalation as opposed to inhalation and thus display an inverted signal.

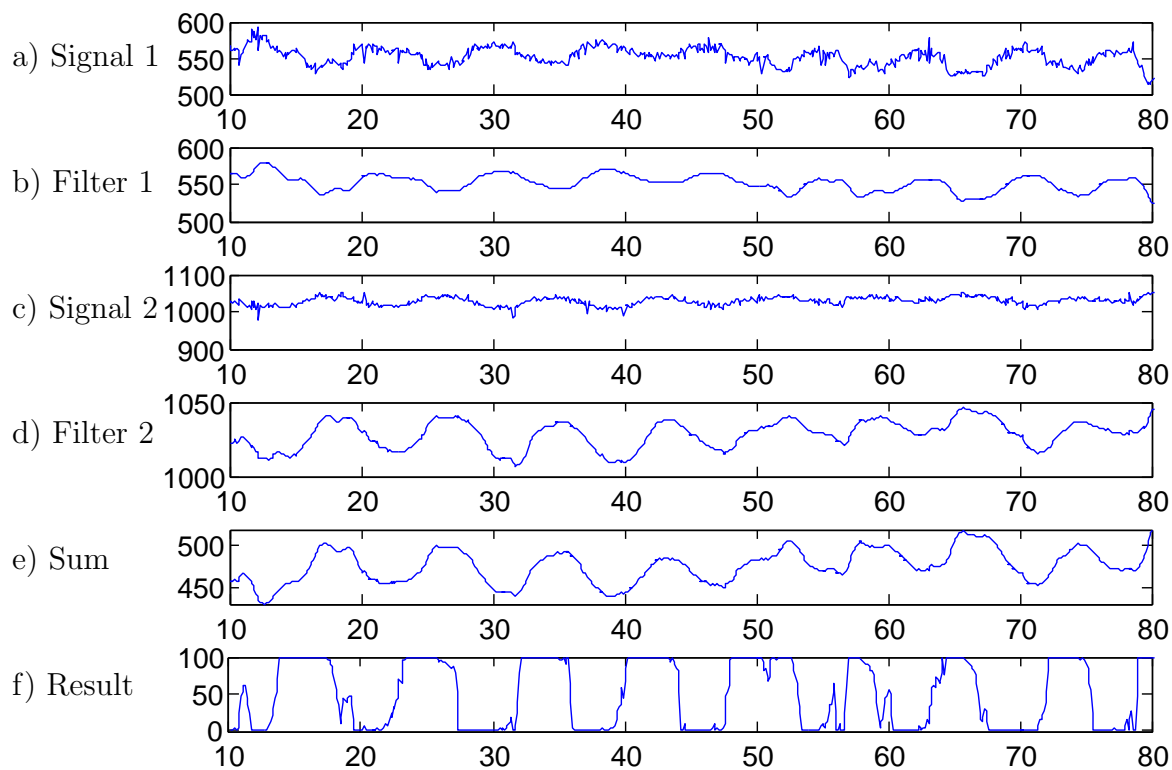


Figure 3.13: Illustration of the processing on the respiratory signals. a) and c) show the raw signals, while b) and d) show the 10-tap low pass filter result of the respective signals. Signal e) shows the b) signal inverted and summed with the d) signal. The final signal f) shows the output result.

as a low power sensor as no energy-expensive analog-to-digital conversion or filtering is required to capture the signal. By using an array of guarded capacitor sensors, noise associated movement artefacts can be reduced thus making the sensor effective in monitoring where highly active motion is being performed. Using an array of sensors which are separately connected, and correlating the signals together should remove the uncorrelated motion artefacts if the array is placed with enough spatial separation.

This sensor provides a plethysmograph signal of low data rate, requiring an update in the order of around 1Hz.



## 3.2 Low Power Camera Interface

Several interfaces have been presented to enable microcontroller to camera sensor connections. Comprehensive surveys [57, 58] show that the highest resolutions for wireless sensor networks are in the range of 1.3 Megapixels. Table 3.3 shows a comparison of the major interfaces developed in the literature.

Table 3.3: Comparison of Wireless Node Camera Interfaces

	Mesheye [59]	Cyclops [60]	CMUcam3 [61]	DSPcam [62]	Citric [63]
Camera Specifications					
Size	640x480	352x288	352x288	1280x1024	1280x1024
Imager	ADCM-2700	Agilent ADCM-1700	OV7620	OV9653	OV9655
Hardware					
MCU	Atmel AT91SAM7S	Atmel Atmega 128	NXP LPC2106	Analog Black- fin	Intel PXA270
Speed	55MHz	4MHz	60MHz	600MHz	624MHz
Camera Inter- face	MCU	Custom CPLD	Averlogic FIFO	MCU	MCU

Three approaches appear to be realised for such systems. The first approach uses a low power microcontroller to enable a high speed DSP to capture and process the image. This approach has a high component count and can be undesirable for small wearable systems or low power wireless sensor networks. The second approach uses a high powered microcontroller with a built in camera sensor interface. This approach requires smaller component count, but uses higher sleep currents and therefore is undesirable for long term use. The third approach uses an extremely low power microcontroller with dedicated camera interfacing hardware to allow the low power microcontroller access to the high speed camera data. This work details an implementation of this approach.

Several high power microcontrollers exist with camera interfaces integrated into the device. These relatively high powered microcontrollers generally operate efficiently at speeds over 200MHz. These applications are not directly suitable for this low power application, as we are attempting to use a low frequency, low power microcontroller under the lowest power possible.

The requirement of complex power management systems required for these high powered microcontrollers, combined with the relatively high deep-sleep currents makes these microcontrollers unsuitable for our application.

### 3.2.1 Parallel Camera Interface

The common interface to modern CMOS cameras has several names including BT. 601 Interface [64], MIPI Legacy Camera Parallel Interface and the ITU 656 Interface. The interface follows a principle shown in Figure 3.14. It is interfaced with eight or ten pixel data lines and has three outgoing clock lines along with the data lines. The master clock is driven from the external communicating device to the camera, which then uses this clock to derive the further clocking and data signals. The vertical clock line delineates a start of a frame, while the horizontal clock represents each line in a frame. Each pixel clock represents a full byte of data (or 10-bits). It should be noted that in most modern colour schemes each pixel is represented by at least two bytes meaning that the horizontal line stays high for as many pixels times the bytes per pixel colour scheme is being used. The pixel data is only valid while the horizontal clock is high. In some systems the vertical clock is also kept high for the whole frame.

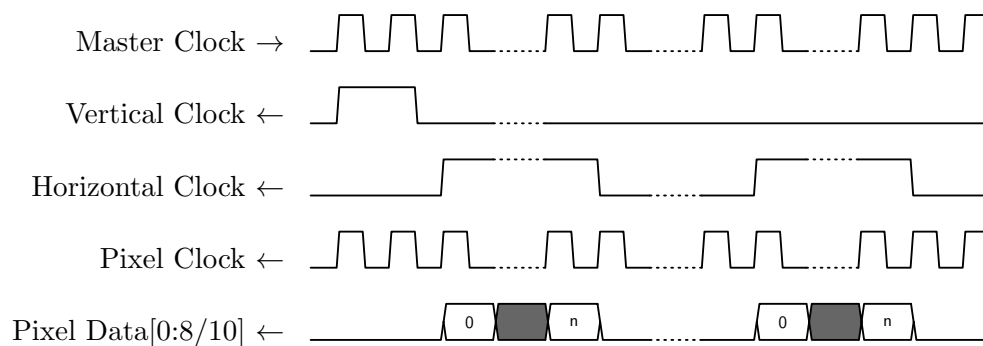


Figure 3.14: Camera interface clocking signals

Camera data is sent from the camera sensor without buffers, and therefore the receiving device is required to match the speed of the camera sensor. The ITU BT.601 Interface standard specifies that this should be around 13.5 MHz.

Along with the parallel interface commands are communicated to the device via an external I<sup>2</sup>C or SCCB Interface. These signals set the parameters for colour encoding, and other features of the Camera Sensor such as image width and height, and any further processing features.

### 3.2.2 The Low Power MCU Interface

The ability to interface a microcontroller to the camera interface is limited by the requirement of high speed data storage. Using architectures such as those presented in [61] with newer high resolution Frame Buffer ASIC's such as the Averlogic AL460A-7 require high quiescent currents in the range of 100s of milliamps rendering them undesirable for low power applications.

To solve this problem, a separate low power interface was developed to provide this storage and have that accessible by the Microcontroller. Figure 3.15 shows the overview of the camera interface.

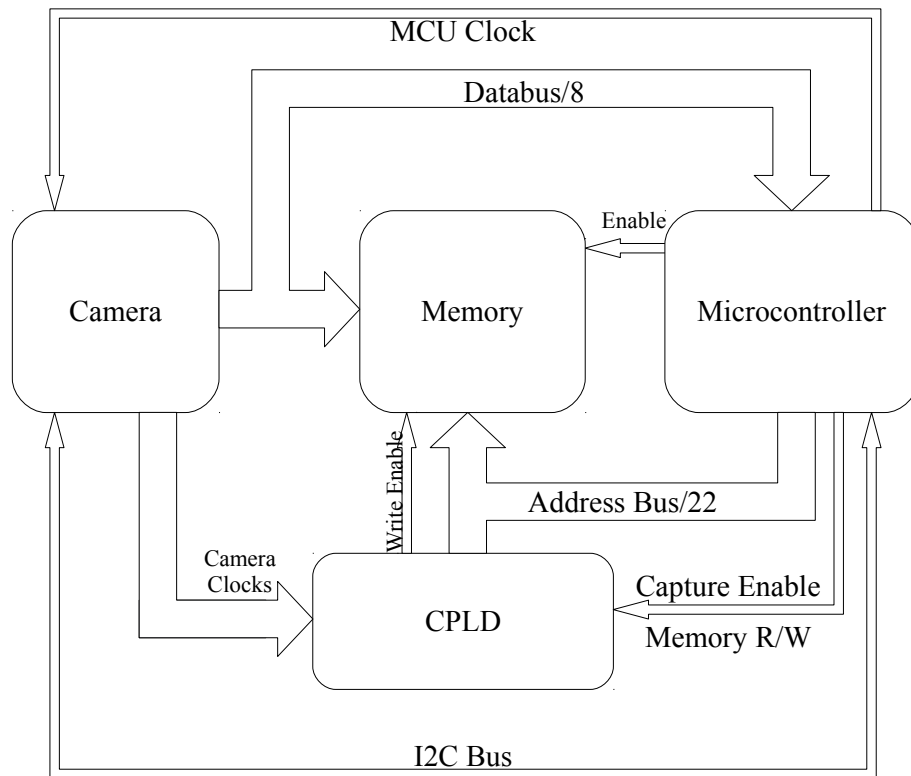


Figure 3.15: Overview of Low Power Camera Interface

The system consists of four main components: a low power Complex Programmable Logic Device (CPLD) and a Low Power Parallel SRAM for the interface and buffer, the camera and microcontroller. Using this setup allows one to write rapidly to the SRAM buffer which can then later be accessed by the MCU. By ensuring the use of low power components allows the interface to be very energy efficient and thus enable long-term continuous periodic capture of discrete still images.

Although this architecture is similar to the interface presented in [60], it differs in several aspects. This low power interface does not have a common data bus between all components, as the data bus is not shared with the CPLD. This allows for a CPLD with a lower pincount reducing the total size requirements of the components. The Camera Clock is also not driven by the CPLD, but rather the MCU further reducing the required components and pincounts. The interface does not include a Flash RAM and the intention is to use the SRAM as a buffer for processing, not to store the image for long term use.

The selected components for the interface are a Cypress CY62177EV30 MoBL ultra low power static ram, a Lattice Mach 4000ZE low power CPLD, an Energy Micro

Table 3.4: Interface Components and related Energy Consumption

Device	Part	Supply Voltage	Active Current	Sleep
Interface				
SRAM	CY62177EV30	2.2V→3.7V	4.5mA*	3 $\mu$ A
CPLD	Mach 4032ZE	1.8V	50 $\mu$ A	10 $\mu$ A
Other Components				
Camera	OV5642	2.6→3V	270mA	25 $\mu$ A
MCU	EFM32G280F128	1.8→3.8V	220 $\mu$ A* <sup>†</sup>	2.75 $\mu$ A

\* Based on a frequency of 1MHz

<sup>†</sup> Running Prime number calculation

EFM32G280F128 and a 5Megapixel OV5642 Camera sensor as shown in Table 3.4. Three Voltage planes are used along with three low-sleep currents (  $\sim 500$ nA) switchable voltage regulators (LP5951) to ensure extreme low power when not in use.

### 3.2.2.1 CPLD

The CPLD was used to increment to the correct memory address and frame enable switch to ensure data from the camera was correctly written to the SRAM. This functionality allows the high speed from the camera direct access to the memory data bus while still being accessible to the microcontroller at a much reduced speed. It is important that the CPLD could operate at the camera speed as well as having as low power consumption as possible.

Figure 3.16 shows the programmed code of the CPLD to interface the camera to the memory and microcontroller. As the CPLD is the sole controller of the memory interfaces output enable and write enable control lines, the CPLD has two modes of operation which are: frame capture and MCU control. In MCU control mode, the MCU has to send commands to the memory to read or write a byte from the memory. In frame capture mode, a single frame is captured to memory by setting the address of the memory in time with the data driven to the memory interface.

Each Interface box in Figure 3.16 contain code to control the interface between the specific device and the CPLD. The Camera Interface is a straight through interface, merely renaming the connections to ensure the code is easily understandable between the external interface and internal lines. The Memory interface ensures that the memory read and write lines from the MCU Interface override the signals from CPLD. This is to ensure that if the MCU attempts to read or write the memory, the camera interface does not write to the same memory lines causing unknown behaviour. The MCU Interface

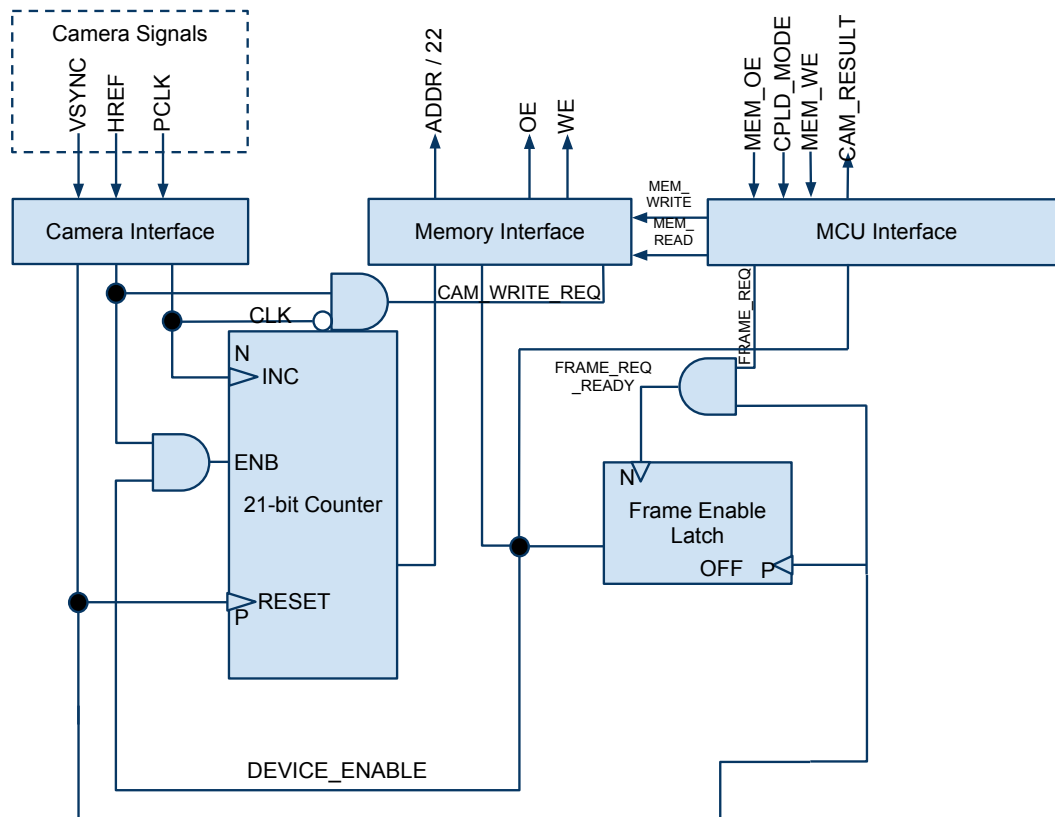


Figure 3.16: CPLD Code

controls the signal lines between the Microcontroller and CPLD. It ensures that the memory read and write lines are disabled if the device enable signal is high.

### 3.2.2.2 Memory

The memory requirements for the interface are: low power, high speed and with a parallel interface to data. The speed of the memory must be at least that of the pixel clock so as to capture data from the camera to the memory.

The selected memory has a minimum write enable low time of 35ns requiring the pixel clock to run at a maximum of 28 MHz.

### 3.2.3 Prototype

To test the interface, a prototype system was built with a separate camera socket for testing different camera sensors. Data output was via USB through which the images are able to be downloaded and analysed. Figure 3.17 shows an image of the camera test board and associated test cameras mounted on their respective sockets. Circuit diagrams for the prototype can be seen in Appendix B. The sockets are compatible

with the Sparkfun SEN-00637 camera module mounting, but have a further 10 lines for further power and control lines.

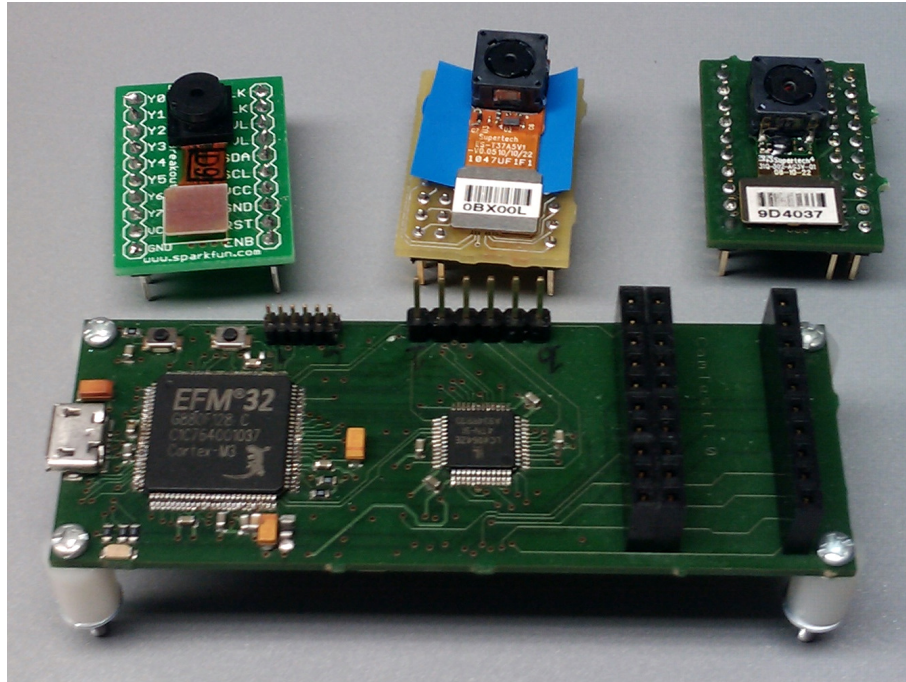


Figure 3.17: Camera Interface Test Board

The components used for the interface and their related power consumptions are shown in Table 3.4. In the table it can be seen that the sleep states for the CPLD and camera are relatively high. To ensure an even lower power of operation, the camera, memory and CPLD were attached to separate low sleep current ( $\sim 1\text{nA}$  LP5951) voltage regulators to turn off the devices when not in use.

Three camera sensors listed in Table 3.5 were used to test the interface. The cameras differed in resolution and current draw, with the highest resolution camera being a FRAMOS C50M5642P30 containing an Omnivision OV5642 camera sensor. Although the output data from the cameras is presented as data to the interface with three clock signals, the data output can use different formats depending on several factors including: polarity, colour and compression format. Both of the Omnivision cameras supported a JPEG output, while the Hynix HV7131GP only supported a VGA frame output with no compression. Due to the JPEG compression enabled on the Omnivision cameras, the output data is of a ratio between 10 and 20 times smaller than the full frame output (depending on the detail within the image). Another reason for the large increase in the camera current for the High-resolution Omnivision cameras is due to the Voice-Coil Motor (VCM) lens driver of the Compact Camera Modules (CCM) compared to the fixed focus of the Sparkfun camera. Having an automatically focused lens improves image quality at the cost of energy.

Table 3.5: Camera Sensors

Camera Module	Sensor	Megapixels	$I_{on}$	JPEG
SEN-00637	HV7131GP	0.3M	~30mA	
C30M3640P30	OV3462	3M	70mA	✓
C50M5642P30	OV5642	5M	270mA	✓

### 3.2.4 Results

Results of some of the functional tests can be seen in Figure 3.18.

To test the systems energy usage, the system current draw was tested using an instrumentation amplifier connected across a  $1\Omega$  resistor. To test the energy efficiency only the OV5642 camera sensor was used to compare all the different resolution modes.

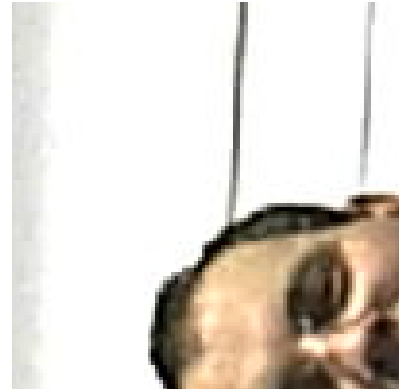
Figure 3.19 shows a comparison of the current draws of a single frame of the OV5642 Camera Sensor for different resolutions. These frames are all compressed using the camera module's inbuilt JPEG compression engine. It can be seen from these plots, that the initial turn on of the camera, focusing and stabilisation: lumped together as setup time in the figure, requires the most energy. The energy used in the actual frame capture, once the setup has been completed, is a comparatively small portion of the overall energy used in capture. Although increasing the resolution of the image capture does take more time, it does not increase the overall length of capture significantly. This means that for a given energy source the number of images captured is mostly dependant on the frame time setup energy as opposed to the image resolution. This suggests that it is not energy expensive to increase the resolution of the image capture, and higher resolutions are more energy efficient in terms of Joules per pixel than lower resolutions.

The setup time consists of several steps for camera initialisation. These include focusing the camera, adjusting the white balance, and programming the camera with initial setup parameters for frame capture. The commands are programmed via the  $I^2C$  Bus Interface and consists of numerous different instructions provided by the hardware vendor. The number of instructions and timings between different turnon stages recorded by the microcontroller are shown in Table 3.6.

The comparison presented in Table 3.6 shows a reasonably static turn on delay. As the camera is designed to be used as a 5 Megapixel camera, the difference in  $I^2C$  commands is due to more settings being required to scale the image down correctly. The increase in Turn On Delay as the resolutions increase is due to setup and stabilisation of the cameras internal Automatic Exposure Control (AEC) and Automatic Gain Control (AGC). The difference in time between the UXGA and 5M frame captures is due to the  $I^2C$  commands combined with the AEC and AGC settings requiring more time than the default 5M settings. The individual frame times scale approximately linearly with increase in frame



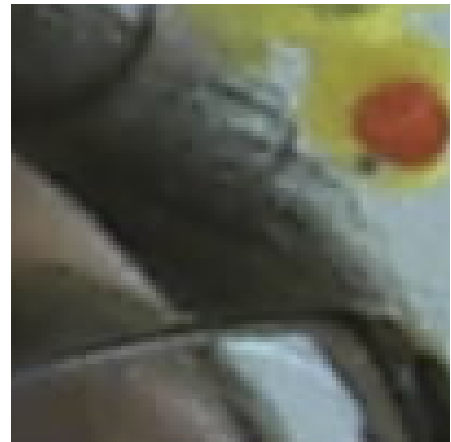
(a) Full 720p Image



(b) 720p 100x100Pixel Closeup



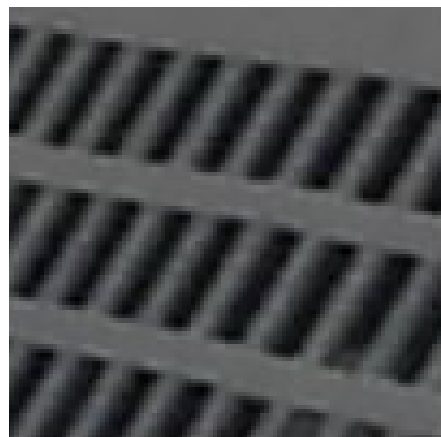
(c) Full XGA Image



(d) XGA 100x100 Pixel Closeup



(e) 5M Image



(f) 5M 100x100 Pixel Closeup

Figure 3.18: Sample Images captured using the interface



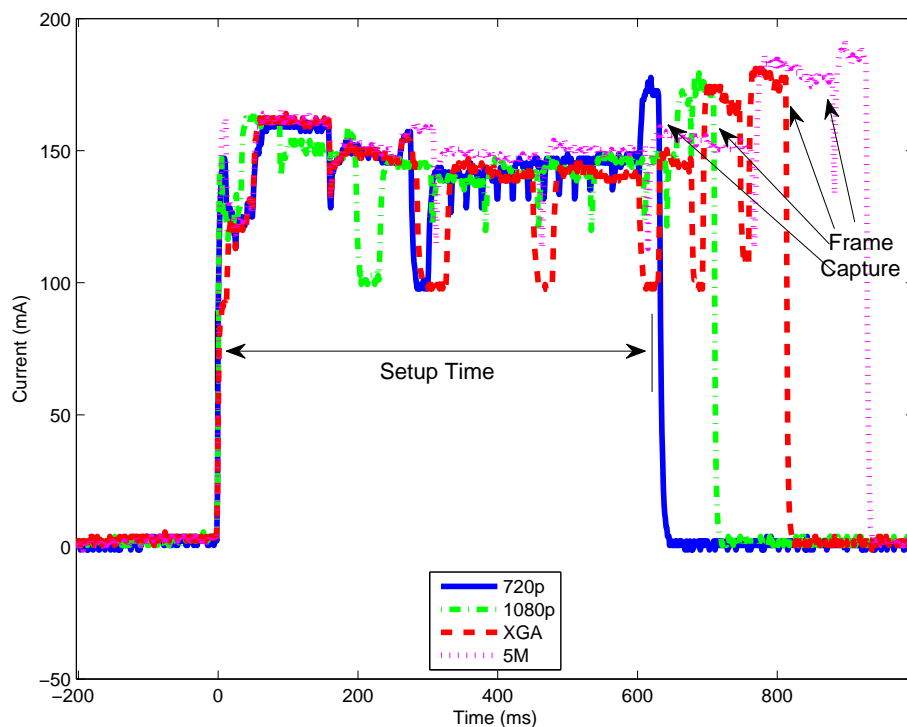


Figure 3.19: Current draw of different compressed image resolutions for OV5642

Table 3.6: Camera Energy Comparison for the OV5642 Camera Sensor

Name	Resolution	$I^2C$ Instructions	Instruc-tions	Turn On Delay (ms)	Frame Time (ms)	Total Energy for Capture (mJ)
720p	1280x720	505		801	26	333.34
1080p	1920x1080	497		832	52	374.07
UXGA	2048x1536	485		895	87	418.41
5M	2592x1944	463		881	105	522.28

size, while the energy consumption is constrained by the initial turn on time being the largest contributor to energy.

Using the data provided in Table 3.6 and Figure 3.19 along with the data of the common Lithium Polymer (LiPo) battery technology, allows us to predict the total number of images we can capture using this interface. This is presented in Figures 3.20 and 3.21. In these figures we assume that the LiPo battery self-discharges at 5% per month and the device is turning off the peripherals during its sleep cycle allowing the device to sleep with a realtime clock using only  $6\mu A$  per second. The LiPo battery is a 1000mAH battery with a nominal voltage of 3.7V. It should be noted that these calculations are based on purely capturing the images to the SRAM, and none of the calculations include any further processing or transmission of the images.

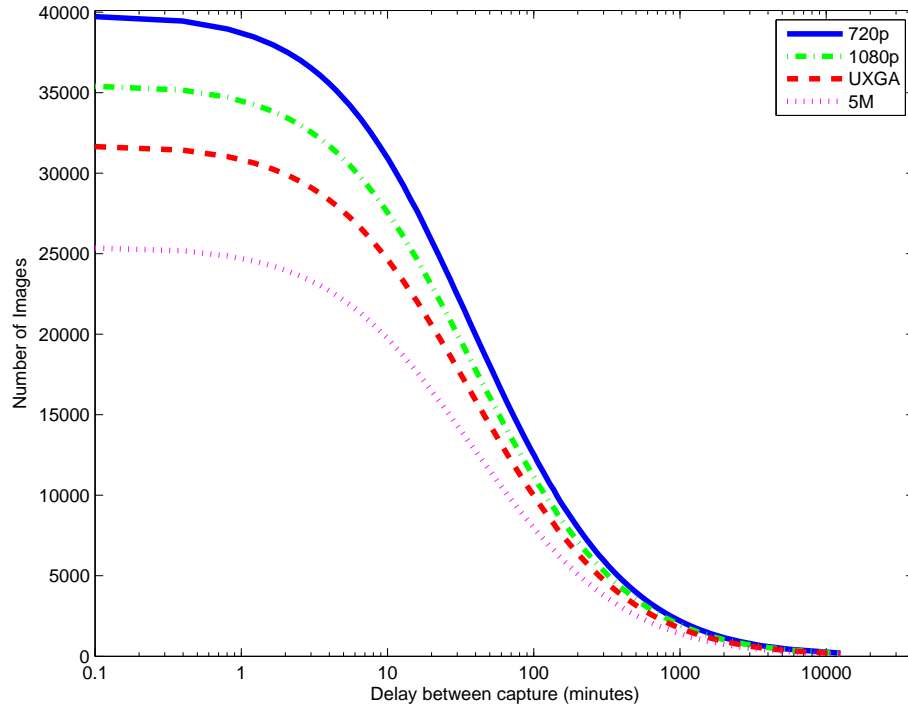


Figure 3.20: Comparison of Delays between image capture and number of images for a 1AH LiPo Battery

Figure 3.20 shows the time delay between frame capture in comparison with the total number of images captured. In this figure it can be seen that for capturing images between large time periods, higher resolution images can be captured at a similar total energy cost as low resolution images, and thus it is recommended to capture the high resolution images as the information quantity is higher.

The graph presented in Figure 3.21 shows the total number of images captured in relation to the total lifetime of the system. From the graph it is clear that the overriding 5% per month self-discharge rate of the LiPo battery limits the lifetime of the system to around 36 months (3 Years). If operating around these lifetimes is required, it is recommended to capture higher resolution images as the information cost of the higher resolution images is low, while the information value is higher.

### 3.2.5 Conclusions and Future Work

In this section a low power interface for high resolution camera sensors has been presented. The interface implements the common BT. 601/ITU.656 Interface in a low power fashion, allowing small microcontrollers to interface with these high resolution camera sensors. We have implemented the interface in a prototype and showed how much current and energy the interface consumes. This interface allows for sub 50MHz 8-, 16- and 32- bit microcontrollers to access 5Megapixel and greater image sensors and thus

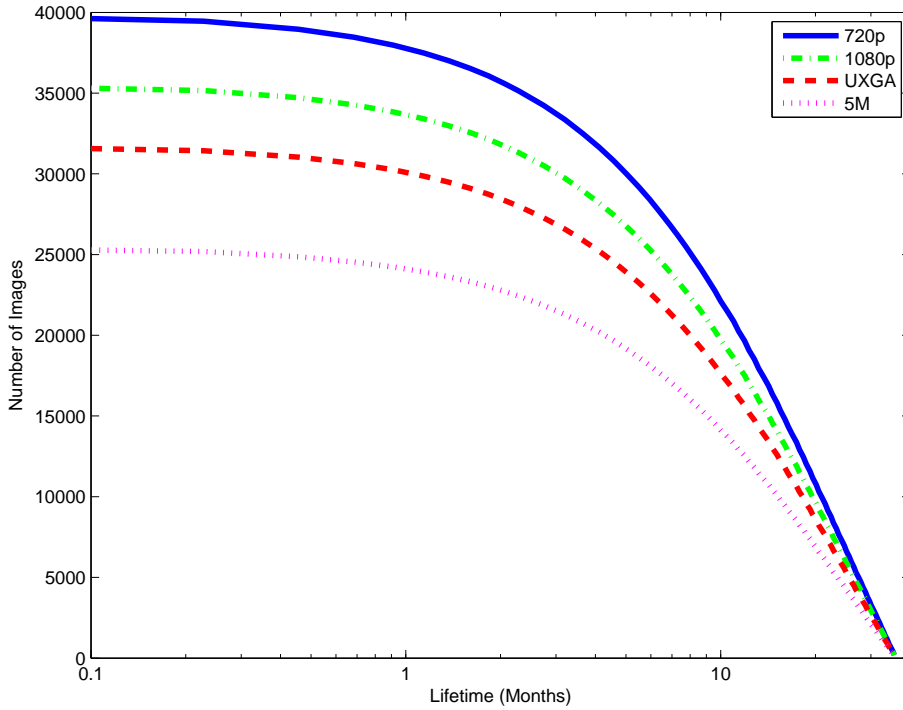


Figure 3.21: Comparison of the total lifetime for capture using the interface for a 1000mAh LiPo Battery

makes it a suitable interface for wireless sensor nodes. The interface can support camera sensors which provide an 8-bit parallel data interface with a maximum resolution of 2000x2000 YUV images, or if the camera sensor contains a hardware JPEG compressor, images below 8MBytes.

Figure 3.19 suggests that reduction of setup time can reduce the total energy used substantially. Due to the setup time including the AGC and AEC settling times, decreasing these times could decrease the overall setup time. This can be achieved using low power light level sensors to detect the correct white balance settings and program these using further  $I^2C$  commands. Increasing the  $I^2C$  communication clock speed could also decrease the total time. This will be done in future work.

### 3.3 Discussion

In this chapter we have designed and developed a novel respiratory sensor and a camera interface for computationally limited, but energy efficient microcontrollers.

On-body BSN sensors are required to be small, unobtrusive and energy efficient for long term use. The multiple requirements of these sensors make them difficult to realise, and thus a new approach is often required to realise small, unobtrusive and energy efficient sensors.

To develop the novel respiratory sensor, a capacitive sensing technique was selected as opposed to an inductive sensing technique which inherently uses lower power and fewer components, thus making the device smaller and more unobtrusive than previous techniques. As the sensing technique is new for respiration sensing, designing the sensor plates is shown, and the operation of the sensors is investigated. It can be seen that each sensor signal is noisy and prone to motion artefacts. To decrease the effects of the motion artefacts and to increase the signal to noise ratio (SNR), multiple sensors can be used together to provide a robust sensor which is still small, unobtrusive and energy efficient. The energy efficiency of each sensor device is limited by the supply current of the two operational amplifiers which typically require between  $10 - 25\mu A$  each [65]. This sensor is thus suitable for long term use in BSNs.

A camera interface normally only used with energy expnsive DSPs is then designed and developed. This interface is shown to be energy efficient and able to be used with low power MCUs. Although the interface allows a MCU to use and access the data captured using the interface, due to the nature of the low power MCU not often being computationally powerful, it is not clear whether the data collected should be further processed on the MCU.

Having captured the sensed signals and made these signals available to the MCU, it is unclear what further processing and actions are required on these signals. The following chapter describes three software frameworks to ensure the information forwarded into the BSN is new and interesting to the BSN, the sensors can easily be accessed by software developers and the information gathered from the sensors is accessible by computationally powerful clusters for processing.



## Chapter 4

# Software Frameworks supporting Distributed Sensor Networks

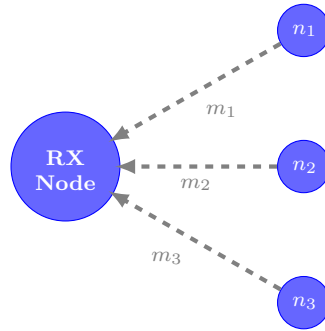
In this section, three different software frameworks for BSNs are investigated. These frameworks are presented as components to enable distributed BSNs. The first framework is an enabling technology for nodes to estimate the quantity or value of information contained in various different signals, and this is compared to the amount of data currently being transmitted through the network. This allows the nodes to know whether their information should be injected into the network, or whether to discard or store the information. This framework is proposed, designed and analysed in Section 4.1. This work is not part of any collaboration.

The second framework enables heterogenous devices to be programmed with the same code, allowing different hardware to use the same code. Similar to systems such as TinyOS, it allows for code to run on numerous platforms, but differs in its approach in being a coding framework as opposed to an operating system to ease the burden of complexity, understandability and lines of code. It also allows a simple and clear representation of the code deployed on the node. In Section 4.2 we propose, design, implement and analyse the framework. Although this framework is based on the Unified Framework [66], the rest of the design, implementation and analysis has been solely done by the author of this work. This software presented in this section is available for download at [67].

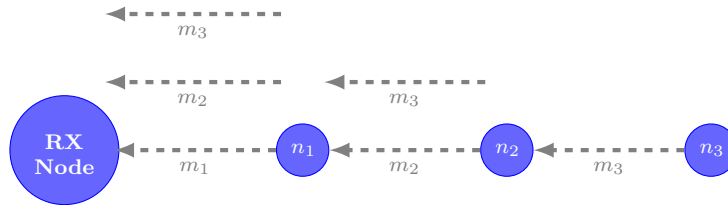
The third framework presents an architecture to show how processing of data gathered by a network of nodes can be distributed vertically through further networks to enable large amounts of processing of simple data rapidly. Section 4.3 proposes this framework and outlines the architecture. This framework is part of the Dejaview project [13] a system to aid memory dynamically by continuously capturing contextual information and providing that information to the user via a mobile handset. The framework outlined in this section has been designed by the author.

## 4.1 Information Costing and Valuation Framework

Wireless sensor networks are a generic tool employed in numerous large scale physical process monitoring functions such as environmental, structural and habitat monitoring [68] and not just BSNs. In these types of generic monitoring processes, the physical process is monitored by many static sensor nodes spread over a large spatial area. The sensor nodes continuously monitor the process and transmit the data back to a base-station or information sink which then further analyses the information. Normally, this sensing task generates large amounts of data proportional to the time the process is being sensed and the number of nodes participating in the sensing task. The number of messages generated by the nodes increases polynomially with the density of the network, as un-coordinated nodes increase packet corruptions due to collisions, and co-ordinated nodes require more co-ordination overhead packets, shown in Figure 4.1a. Nodes operating in multi-hop networks are required to re-transmit their neighbour's data which can also polynomially increase the amount of data to be transmitted, shown in Figure 4.1b.



(a) Number of messages sent to receiver is not linear, as message  $m_1$ ,  $m_2$  and  $m_3$  collide. Assuming  $m_1$  is the only message received, in the following time slot  $m_2$  will be received although  $m_3$  was also transmitted and only in the next time slot will  $m_3$  be received.



(b) In the multihop scenario, the number of messages increases polynomially as the nodes increase

Figure 4.1: Messages increase polynomially with both increase in nodes as well as increase in hops

Although there is a large amount of data to be transmitted, often very little information is contained within the data stream. Data and Information are separated in this text in terms of cost and value. A datastream of 1024 samples all containing the value zero

contains very little information, but 1024 data points. A lot of research has addressed the differences, and information can be calculated in numerous ways including: Shannon Information Content [69], Fisher Information, Chernoff Information [70] and Kolmogorov Complexity [71]. These methods are complex, computationally expensive and require that either the data set is known or describe a measure of uncertainty surrounding a random variable. They all use reasonably complex and intensive mathematical methods and are thus generally unsuitable for computationally low power processors.

The sensor nodes used in these forms of networks are often computationally limited, and thus any processing of the data on the node should not be complex as it will severely inhibit the performance and energy efficiency of the whole system. Minimal complexity of any algorithm used is preferred even at the cost of some loss of information.

As we are dealing with wireless sensor networks, the constrained bandwidth of the network is also generally too limited to transmit all of the data to the final destination, so either an intelligent choice needs to be made to select which information is the most important under the given constraints, or the signals require data compression.

Data compression can be divided into two classes: lossless and lossy compression. Lossless compression guarantees data fidelity, but does not guarantee maximum compression as noise signals are also transmitted across the data path and thus can result in large amounts of unnecessary data, while lossy compression relies on decreasing the data quantity while sacrificing some quality, but attempting to ensure that the valuable information is preserved.

Slepian-Wolf distributed source coding theory [72] shows that lossless compression of separate but correlated data sources can encode the data at the same rate as an ideal encoder which has access to all the separate data sources. This suggests the best case scenario for these types of wireless sensor networks, but has implementation problems highlighted in [73]. Slepian-Wolf coding systems currently also require a large amount of processing at the decoder side, and therefore are not suitable for low power distributed systems without dedicated hardware.

Binary Bayesian Estimation and Distributed Compression-Estimation techniques described in [74] show several different strategies for employing lossy compression of the data sources, but still require large processing facilities at the fusion center which also are currently too processor intensive for current low power wireless sensor networks.

Compressive Sensing [75, 76] also known as Compressive Sampling, is another modern technique which promises extremely high compression without the overhead of pre-processing data. Compressive sampling relies on the fact that the signal is *sparse*, and therefore can be represented in a different basis using few components with many unknowns: an underdetermined system of equations. Using the different basis, one can



solve the system of equations using the  $L_1$  norm or  $L_0$  norm, to preserve sparsity. Compressive sampling shows that the Nyquist sampling theorem does not necessarily apply if one knows that the signal is sparse. It does however, require completely random samples to be taken. Reconstruction of the compressively sampled signal is highly computationally intensive, as one has to convert to the basis functions and solve the system of equations which makes it unsuitable for low power, *distributed* wireless sensor networks.

Model Based Coding (MBC) is an information coding technique whereby the signal being monitored is fitted to a model or previous understanding of the underlying data. Then instead of using the sampled signal values, the model parameters are used as descriptors for the signal. MBC is often used in low bit rate video coding and has been included as an object coding system in MPEG-4 [77].

As compression systems rely on repetition or redundancy in the data, it is important to look at the different forms of correlations in the dataset. Assuming the nodes are fixed in space, there exists a physical relationship between the nodes which could contain spatial correlation. Over time, as the nodes continue their sensing task, there exists a relationship between the sensed values which could have temporal correlation in the signal. By taking advantage of the correlation inherent in the process being monitored, the amount of data transmitted by the nodes can be decreased and therefore increase the lifetime of the sensor network. By enabling the nodes to model the data spatially and temporally, the individual nodes need only to transmit data to update the network model of the information, thus decreasing the data transmitted. Current systems still appear to rely on advanced algorithms which terminate at the sink or fusion centre, and in this work we show a simple self-regulating system which allows for distributed spatio-temporal modelling with little processing which still achieves good results.

Low power data reduction/compression techniques for wireless sensor networks are numerous. There exists three different classes of data reduction/compression systems which rely on inherent signal correlation: temporal, spatial and spatio-temporal. Temporal systems use techniques such as Adaptive Sampling [78], Data Differencing/Delta encoding [79] and other techniques [80]. Spatial Systems tend to use spatial clustering [81], [82], while Spatio-Temporal Systems use combinations of these [83].

Wireless sensor networks are predominantly lifetime limited by their energy source, and thus the value of the information generated can be compared to the energy spent on the measurement and transmission. This economic or utility based value/cost model is often employed in Wireless Sensor Networks, and examples of these can be seen in [78, 84].

The Acquisitional Query Processor (ACQP) [85] for sensor networks is a framework developed within TinyDB [86] to treat the sensor network as a relational database, and developed a language and related algorithms to support this analogy. A query is generated by the user, which is then optimised using network information previously gathered by the root node of the network. ACQP uses a cost-based optimizer to minimise

the data cost of delivery, which is estimated from the root node of the network. The query is disseminated to the network nodes, and when the nodes feed the information back to the sink, data can overflow. When this condition occurs, data requires prioritisation, and it is shown in this work [85] that a delta-based scheme computing the time difference in samples delivers the best results.

In the work by Merrett et al. [87], messages are given priorities, while the node is given power levels. Using a predefined rule based system, the node compares the power level available at the node, to the message priority of the message and transmits its or its neighbour's messages based on whether it has the correct power level, sacrificing lower priority messages on the network if the power level of the node is low.

The system called BBQ, developed by Deshpande et al. [84], shows an efficient probabilistic model based coding technique for sensor networks which is designed for database related declarative queries. Due to the statistical nature of this system, one still needs a complex processor able to calculate the query plan and store the previous statistical information which makes it undesirable for distributed WSNs.

Tree based Polynomial Regression (TREG) proposed by [83] uses a regression model to describe the data which is updated in a tree-like structure. Each clusterhead node holds information about the lower leaf nodes in a regression model, and when requested from the fusion center, transmits the co-efficients of the model, thereby decreasing data transmitted.

Shannon-Nyquist Entropy, a measure of the amount of information generated at a source, shows that maximum entropy is obtained when there is an equal probability of each outcome. This is shown in the example of a fair coin toss, where the probability of the coin landing on heads or tails is  $\frac{1}{2}$ . If the probability of the coin is biased either way the overall entropy and thus information generated by the source decreases. If one were to extrapolate this concept over a set of sensor nodes monitoring a physical process, one can see that the information is generated by the physical process and not the specific sensor nodes. Thus to ensure the least amount of information, and thus the maximum entropy, the sensors must generate a set of symbols which have equal probability of occurring, or the symbol length must be as close as possible to the entropy of the symbol. As an example, a set of a thousand nodes monitoring a binary process should not report a thousand answers which are the same, but rather a single answer to minimise the data throughput. Similarly, if this binary process were to change, not all nodes should report that the signal changed, but rather a single answer should be given to minimise data throughput. This type of network would minimise the data throughput over the physical process rather than the network of devices. A form of an equally probable spatio-temporal sensing field. One would thus like the sensors to transmit 'new' information continuously throughout the network, and not transmit redundant or similar information.

The goal of the framework presented in this section is thus to transmit only new information in the network which is valued against an approximate global network cost thus providing the most out of the network.

In most sensor network deployments it is important to start sensing and transmitting data at or close to the optimal settings immediately, as it is difficult and laborious to manually tune the network to the optimal settings. Model based approaches require a training set of data against which to train the network, while rule based systems require a set of rules. This framework should also operate without the use of a finely tuned training set to operate close to optimally. The only initial settings required should be those for the specific hardware being used, that is the energy cost and radio transmit parameters.

#### 4.1.1 System Overview

In this work a system of valuation and costing of a node's information before the data is transmitted across a wireless medium is presented. Figure 4.2 shows a flowchart of the different stages of the system operation on the wireless node. This system allows an individual node the value of its own information relative to the overheard network information, and thereby have an understanding of whether it can decrease the in-network uncertainty by transmitting or withholding information. This requires a node to have transmit as well as receive abilities and thus increases the complexity required in a sensor node beyond transmission only.

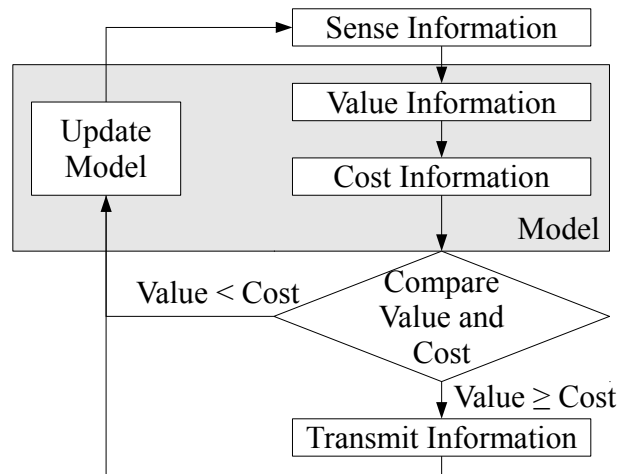


Figure 4.2: On node Information Valuation and Costing model overview

The internal rule-based information model consists of some simple statistics of the data it has received regarding the parameter being sensed, and a set of required processing tasks to perform on the data before transmission. These tasks are described as rules which are run in order, and result in a binary decision to transmit or withhold the information.

Once the information has been transmitted or withheld, the model is updated and the node repeats the sensing task.

This approach allows for simple rule-based modelling of the data. Parameters for the rules can be dynamically updated and modified in the network at runtime to increase or decrease data-rates or information value relative to the specific sensor. It is important to note that the rules do not change the data directly, but modify the information relative to value and cost. Some parameters are required to be determined at runtime, but not pre-determined by running a training set of data. The values for the hardware parameters which are pre-determined and programmed before startup are: sensor parameters, radio parameters and energy consumption parameters.

The rules thus represent a model for the information valuation of the data which is run pre-transmission of a data packet.

This system stands independent of routing tasks, and it is assumed that routing systems such as Directed Diffusion or Semantic Routing Trees are employed to evaluate the routing path.

The rule based model consists of two equations which are calculated and compared with each other before transmission. The first equation calculates the value of the data in the packet it is about to transmit, while the second calculates the cost of the data packet to the sink with respect to the network. If the value of the data is low with respect to the cost, then the packet should not be forwarded into the network, as it will not benefit the co-operative network greatly. On the other hand, if the value of the packet is high with respect to the cost, send the packet. Once the packet has been transmitted, update the model with the new sensor parameters.

By running a value and cost comparison of each data point the node intends to transmit, the node can assess the relative value and cost of its information against the network of nodes, and not just the node itself. The two models for information value and cost are described in Sections 4.1.2 and 4.1.3 respectively.

#### **4.1.1.1 Differences from Previous Methods**

Delta Encoding is a form of data compression, and decreases the total information transmitted, by only recording the difference between one sample and the next instead of the full sample value. In data streams where signal values often vary with small differences, delta encoding can greatly decrease the amount of data being transmitted. This system uses deltas to assess the value, but instead of transmitting every small delta value, only transmits the sample and not the delta when it is greater than the energy and bandwidth cost.

ACQP in TinyDB [85] uses a delta-encoding system to order value in samples when the network is congested. It is therefore only run when the transmit buffer overflows. ACQP does not value the delta information against the cost of transmission on the node, it only performs a cost optimisation during the query plan before it has ever encountered congestion during the data delivery. In contrast, this system performs the information valuation and cost when the data is about to be transmitted, and therefore should have a more effective understanding of the data path to the sink.

The Rule-based Information value and costing system presented in [87] compares the message priority to the power priority of the packet to transmit, but requires hard coding of what is regarded as an important or un-important message defined by rules without the nodes evaluating the information transmitted. The power priority rules do not consider the cost of the transmission path, but only the level of power available to the node, and therefore do not support the network as a whole, but only individual nodes.

Similar to TREG [83], this work uses a model to describe the data, but as opposed to TREG, in this work, we still transmit the data and assume the receiving node can recreate the same data stream with the samples transmitted. In this work, we create a measure of the information contained in the data samples, and transmit that along with the data, as opposed to transmitting the polynomial co-efficients.

The next subsection deals with how the model operates in terms of Information Value and Information Cost.

#### 4.1.2 Information Value

In this work, we are trying to understand the value of the information in the network and will assign a variable  $I_v$  to this value for the sensor variable  $x$ . As wireless sensor nodes are spatially separated, their value should be defined in not only time, but space as well.  $I_v$  is thus composed of two independent components, which we will describe as  $I_{vt}$  and  $I_{vs}$ . These two components are described in equation 4.1. The values are summed together to represent the simplest form as they are separate and not further compounded by each other in anyway. It must be noted that the values for  $I_{vt}n$  and  $I_{vs}n$  are dimensionless and have been normalised thus representing values between zero and one. The final  $I_v$  is thus a value between zero and two.

$$I_v = I_{vt}n + I_{vs}n \quad (4.1)$$

The information value component relative to time, or *temporal value* describes a physical parameter as the information changes with respect to the node over time. It is a dynamic component which could change as often as the sensor node samples a signal. Valuing a

data signal with respect to time, or temporal valuation, is a technique used in many other protocols or network standards. Temporal valuation in a network can enable Quality-of-Service (QoS) in lossless links and streaming capabilities in lossy links. The information value component relative to space or *spatial value* describes how a parameter is limited in space, relative to other nodes, and therefore its spatial footprint over the network. Spatial valuation of information in a network creates a metric of spatial correlation between the sensor nodes, and thus allows for: the use as a value to control the Signal-to-Noise Ratio (SNR) of the sensing task; increasing the redundancy of the information generated; locating specific areas of spatial importance; and other factors depending on the application.

These two valuations can be described as mathematical models from which one can create many different models. To show the value and use of this system, two basic value models have been created to describe the temporal and spatial valuation. These are described in the next two subsections.

#### 4.1.2.1 Temporal Value Model

We wish to create a function which can be run on a low power sensor node which will *estimate* the value of the information gathered at the sensor node in terms of time. To do this, we need to investigate how the information value changes over time to see how the function should operate.

Let the sensed value be  $x$ , the valuation function is thus not entirely on time, but on data generated by the sensor as well  $I_{vt} = f(x, t)$ . Let us also assume that we have only  $m$  available memory slots in the sensor node. As we are only trying to establish the changes of the signal in time, the valuation function could be of the form:

$$\begin{aligned}
 I_{vt} &= a_0|(x_0 - x_1)| + \dots + a_m|(x_m - x_{m+1})| \\
 &= a_0|\Delta x_0| + \dots + a_m|\Delta x_m| \\
 &= \mathbf{a} \cdot |\Delta \mathbf{x}|
 \end{aligned} \tag{4.2}$$

where the vector  $\mathbf{a}$  of length  $m$  describes the weightings of change over the memory slots in the node, and  $\mathbf{x}$  describes the sensed values over the last  $m$  time slots. As we are using low power sensor nodes to perform all the calculations onboard the device, a compromise in processing is used to speed up calculations and minimise processing time on the node's MCU. A  $|\Delta \mathbf{x}|$  is used instead of the normal  $(\Delta \mathbf{x})^2$  as this has shown to be efficient, although approximate, on computing tasks such the magnitude calculation of the Sobel edge detector [88]. This valuation function has a similar composition to a digital FIR filter, except that we have set the input signal to the change in input signal

to represent the temporal change. By changing vector  $\mathbf{a}$ , (the taps or filter co-efficients in FIR filter terms) we can change the importance of the current and previous data values, and thus attain an estimation of the temporal value of the data on the sensor node, by looking at how the information has temporally changed over the last  $m$  time slots. The absolute value of the difference is used as we define information as a positive value.

The calculated value does not depend on which values have been sent as this information does not in any way represent the temporal value of the physical process, but rather a temporal value of network information. Incorporating the values which have been transmitted is left for future work.

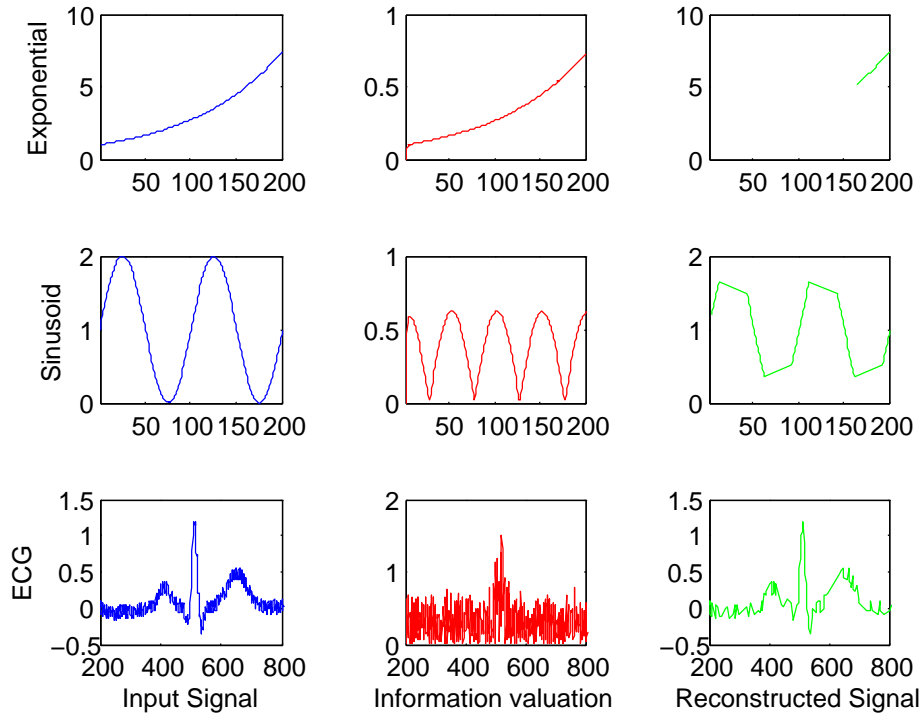


Figure 4.3: Temporal valuation and reconstructed signals of 3 different signals

To analyse this in graphical form, we can look at Figure 4.3 and Table 4.1. In this figure we have taken three arbitrary signals: a rising exponential, a sine wave and a single ECG beat, and analysed the signals information value according to the formula in Equation 4.2. We can see the input signals, information valuation of a ten memory slot information valuation filter with vector  $a = \{1024, 512, 256, 128, 64, 32, 16, 8, 4, 2\}$ , where  $a_0 = 1024$  is the coefficient for the most recent value. These values were chosen to represent recent rapid data changes being of extreme importance, and only used to demonstrate the system conceptually. An arbitrary threshold value of 0.5 has also been selected in this example. From this diagram we can see that the temporal valuation filter increases the priority of the message when the change increases beyond a threshold value. For the first row, an exponential growth in the signal, the valuation exponentially increases as well, while the reconstructed signal on the right hand side is only reconstructed once

the threshold information value of 0.5 is reached. The reconstructed signal is taken from the values which have surpassed the threshold value, and thus been classified as important. For the second row, a sinusoid input signal is used. Here it can be seen that the temporal value estimation appears to ‘bounce’ as the gradient of the signal changes. By extracting the components with the highest temporal value (steepest gradient), we can reconstruct the signal with the important frequency components intact, but we do lose the minimum and maximum values of the signal, as we are only recording periods of greatest change. The third set of signals is a synthesised ECG signal. We can see that the valuation oscillates around the information value threshold due to the noise on the signal, and reconstructs the temporally valuable parts of the signal. This method does preserve several key components, and allows one to compress the signal using very little processing.

Table 4.1: Size comparison of Input Signal, average temporal information and Reconstructed signal for Figure 4.3  $I_{vt}$  of 0.5

Samples	Mean $I_{vt}$	Reconstructed	Final Ratio
201	0.3143	38	18.9%
201	0.3935	78	38.8%
8686	0.3108	1549	17.8%

Table 4.1 shows an overview of how we have decreased the overall number of samples required to be transmitted, while still maintaining a number of important aspects of the signal we are transmitting. We can therefore decrease the number of samples sent temporally by the ratios shown in the right hand column of the table. In this example, we have reconstructed the signal from messages with a value higher than 0.5. By adjusting this threshold value, we can increase or decrease the amount of data transmitted which linearly adjusts the detail or resolution of the signal. On startup, the system should select a information value close to zero to allow most data points to be transmitted, and as the system continues to operate, this information value can be increased by the sink nodes to decrease the information throughput.

The temporal valuation for  $I_{vt}$  requires a final step of removing dimensions and normalising the value to between zero and one so as to allow it to be added to a different component of spatial value. This is done by dividing the maximum temporal value computed in a recent time period as shown in equation 4.3. This maximum temporal value and recent time period is required to be stored on board on the sensor node.

$$I_{vt}n = \frac{I_{vt}}{\max(I_{vt})} \quad (4.3)$$

Having created a temporal estimation of information value, we now look towards creating a similar model for the spatial value.



#### 4.1.2.2 Spatial Value Model

As we are dealing with spatially separated nodes, there is a physical relationship between the nodes which can be approximated mathematically. It is assumed that the signals that are monitored across the network are of the same signal type, ie. temperature or oxygen saturation readings, but not necessarily calibrated between them, and it will also be assumed that the nodes are stationary in reference to each other. There thus exists a fixed correlation in the signals received from the nodes depending on a number of factors, including the distance between the nodes, the physical interactions of the objects in their surrounds, and changes in the physical property which the nodes are sensing. It will initially be assumed that these nodes are fully connected, without loss on the transmission path, but further consideration of partial information is considered later.

As we are assuming that the nodes' data is not necessarily calibrated, we need to investigate the difference between the node's data and its neighbouring node's data. To achieve this, we take the differences between the current nodes data and their neighbours' data. Thus, for each node  $n$ , where  $\mathbf{d}$  is a vector of length  $k$ , and  $k$  is the number of nodes in the transmission and reception range (neighbourhood) of the current node,  $n_0$ . We have removed the first term, as the difference in information between the node's own information should be zero. The vector  $\mathbf{d}$  describes the importance of each node's information in relation to the other nodes lumped with calibration scaling if any (as any calibration offset would be removed from the differencing). This vector is thus closely related to the density of the nodes, as nodes which are very dense, should have more correlated signals (and thus lower  $\mathbf{d}$ -values to close neighbouring nodes), while nodes which are sparse, should have widely different signals (and thus higher  $\mathbf{d}$ -values associated with those nodes' signals). The vector  $\mathbf{d}$  is also affected by the nodes scaling calibration as any nodes which are not well calibrated in terms of scaling will have differing values.

$$\begin{aligned}
 I_{vs} &= d_1|(n_0 - n_1)| + \dots + d_k|(n_0 - n_k)| \\
 &= d_1|\Delta n_1| + \dots + d_k|\Delta n_k| \\
 &= \mathbf{d} \cdot |\Delta \mathbf{n}|
 \end{aligned} \tag{4.4}$$

Looking at equation 4.4, it is assumed that the network of nodes which we are involved in the sensing task can overhear all of the data of the network of neighbouring nodes. This is often not the case or virtually impossible and is also one of the reasons why we wish to value the information before transmission. To mitigate the need to transmit all the data of the neighbouring nodes, we assume that the data set is incomplete, and thus need to normalise the received data from the nodes from which the data was received otherwise well connected nodes would inherently have higher spatial value than poorly

connected nodes. It is assumed that the node will transmit at its given time, not having a complete data set, the spatial value will fluctuate depending on whether it has received all the other nodes' data. WSN's are notoriously prone to link failure, and relying on all the nodes' information before transmission is clearly not an optimal way to value the spatial information. It is therefore important to use a more robust and stable measure of spatial information valuation. We therefore present equation 4.5 to represent this.

$$I_{vs} = \frac{1}{nr} \{d_1|\Delta n_1| + d_2|\Delta n_2| + d_3|\Delta n_3| + \dots + d_{kr}|\Delta n_{kr}|\}$$

but as an example, values  $n_1$  and  $n_2$  might not have been received, therefore

$$\begin{aligned} I_{vs} &= \frac{1}{nr} \{0 + 0 + d_3|\Delta n_3| + \dots + d_{kr}|\Delta n_{kr}|\} \\ &= \frac{1}{nr} \sum_{i=1}^{kr} d_i|\Delta n_i| \\ &= \frac{1}{nr} (\mathbf{d}_{\mathbf{kr}} \cdot |\Delta \mathbf{n}_{\mathbf{kr}}|) \end{aligned} \tag{4.5}$$

Equation 4.5 is similar to equation 4.4, but takes into account that the information obtained from the neighbouring nodes is incomplete, and thus normalises the value relative to the number of nodes overheard. The value  $nr$  is thus the number of values used, and the value  $kr$  is the last node for which the current node has a value. This equation does require each node to store more information in the node's memory than it will use on every transmission and is therefore highly dependent on node density.

The information value generated from the above two equations is still relative to the signals being sampled. It is therefore required to normalise the spatial information value to be used across a network of different sensors. To enable this, the sensor requires knowledge of the resolution, minimum and maximum values of the signal being monitored. These values can be inferred from the data being sensed dynamically, as a sensor which detects a signal value greater than it has previously ever sensed will have a maximum information value. Therefore to normalise the sensed values, a division has to occur of the maximum previously sensed value in a specified time period in order to compare it to the normalised information cost. The normalisation of the spatial information is shown in equation 4.6.

$$I_{vs}n = \frac{I_{vs}}{\max(I_{vs})} \tag{4.6}$$

### 4.1.3 Information Cost

In this work, we are trying to send as much information as possible to any node or sink, with the minimal amount of energy available to the network nodes, thereby maximising the life of the network. The cost of the information is a sum of the vector's energy  $I_e$  and bandwidth occupancy  $I_b$ . This work regards bandwidth as a spatial representation of how much data can be sent across the network at a specific time. Bandwidth is an important limited resource, and overuse of bandwidth causes among other things, packet corruptions between nodes transmitting data. It is therefore costly for the nodes to overuse this resource, as over the air packet collisions are severely detrimental to the network energy consumption and performance. To model our information cost  $I_c$ , it will be represented as a sum of the bandwidth occupancy  $I_{cb}$  and the energy cost  $I_{ce}$  as per equation 4.7. Similarly to equation 4.1, the terms are summed to represent the simplest form and the values do not compound in any way, and the values of  $I_{ce}n$  and  $I_{cb}n$  are each normalised thus being values between zero and one. The total value of  $I_c$  is thus a value between zero and two.

$$I_c = I_{ce}n + I_{cb}n \quad (4.7)$$

The following subsections will describe the models created for the energy costing and the bandwidth costing.

#### 4.1.3.1 Energy Cost Model

Many WSN systems use energy costing as either a metric to measure the efficiency of an algorithm [89], or a measure to cost the information before transmission, which ranges from vastly complex systems, to simple models. To ensure that the node can perform the operations required, a simple model to estimate the amount of energy used in transmitting and receiving a single sample of information is used. The simple model is described in equation 4.8.

$$I_{ce} = E_{tx} + E_{rx} + E_{sampling} + E_{processing} \quad (4.8)$$

This model takes the values of  $E_{tx}$  and  $E_{rx}$  from LEACH [90] while adding the sampling and processing values  $E_{sampling}$  and  $E_{processing}$  in a similar fashion. The  $E_{elec}$  term refers to the energy per bit of the RX/TX circuitry, while the  $\epsilon_{amp}$  term refers to the energy per bit per metre squared to achieve a sufficient SNR for reception. The  $k$  term refers to the number of bits in the message, while the  $d$  term refers to the distance between the transmitter and receiver. This model applies to a free space  $d^2$  energy loss model for channel transmission.

$$\begin{aligned}
E_{tx} &= k(E_{elec} + \epsilon_{amp}d^2) \\
E_{rx} &= E_{elec}k \\
E_{sampling} &= \epsilon_{sample}k \\
E_{processing} &= \epsilon_{process}k \\
E_{rxtx} &= 2E_{elec} + \epsilon_{amp}d^2 \\
E_{mcu} &= \epsilon_{sample} + \epsilon_{process}
\end{aligned}
\tag{4.9}$$

$$E_{mcu} = \epsilon_{sample} + \epsilon_{process} \tag{4.10}$$

Using these equations given above, we can reduce equation 4.8 to the form given in equation 4.11, noting that as there are many summations which we do not want to perform on the node, we will abbreviate the sampling, processing, and RX/TX circuitry terms into two terms shown in equations 4.9 and 4.10.

$$I_{ce} = k(E_{rxtx} + E_{mcu}) \tag{4.11}$$

For this equation to be suitable, the node requires a metric which it can relate to the distance. This is not an easy accomplishment in wireless sensor networks as range-finding and range estimation are active research areas [91] with no clear best-use scenarios. Distance estimation methods are numerous and complex including techniques using Received Signal Strength Indication (RSSI) [92], Link Quality Indication (LQI) [93] and Time-of-Flight (TOF) [3].

Table 4.2: Extract of the CC2420 Radio Module Datasheet showing the related output power to current consumption of the radio module [94]

PA_LEVEL	TXCTRL register	Output Power [dBm]	Current Consumption [mA]
31	0xA0FF	0	17.4
27	0xA0FB	-1	16.5
23	0xA0F7	-3	15.2
19	0xA0F3	-5	13.9
15	0xA0EF	-7	12.5
11	0xA0EB	-10	11.2
7	0xA0E7	-15	9.9
3	0xA0E3	-25	8.5

A simpler approach to approximate  $I_{ce}$  is to use the radio's transmission power level which is often tabulated in the radio specifications relating power in decibels milliwatt (dBm) to current in milliamps (mA) as shown in Table 4.2. Using this current  $I_{tx}$ , along with the node voltage  $V$ , gives a good costing of the transmission power which the node can calculate rapidly. The equation is shown in 4.12. To convert to a joule per bit

format, we can multiply the current by the voltage and divide by the data rate provided by the radio module being used. Thus we have the energy cost for the data packet.

$$\begin{aligned}\epsilon_{dBm} &= VI_{tx} \\ E_{rtx} &= 2E_{elec} + \left( \frac{\epsilon_{dBm}}{R_{max}} \right)\end{aligned}\quad (4.12)$$

Using this equation as the total cost, allows us to estimate the information energy cost before transmission.

For this energy cost to be used by the sensor node, it is required to be normalised before use so it can be compared to the Information Value. To do this, the maximum energy cost of a packet is required to be calculated. This is done by looking at the maximum power setting given in table 4.2 and calculating the maximum  $I_{ce}$  using this value. The value used for transmission of a packet is then divided by this value. Only when the node changes the power settings or packet length is it then required to update  $I_{ce}n$ . The calculation of  $I_{ce}n$  is shown in equation 4.14.

$$I_{ce} = k \left( 2E_{elec} + \left( \frac{\epsilon_{dBm}}{R_{max}} \right) + E_{mcu} \right) \quad (4.13)$$

$$I_{ce}n = \frac{I_{ce}}{\max(I_{ce})} \quad (4.14)$$

#### 4.1.3.2 Bandwidth Cost Model

It is important to estimate the cost of the bandwidth occupancy before a message is sent, as bandwidth is a limited resource. The bandwidth the data will occupy during its transmission could prevent other more important information being transmitted or even damage another node's data being transmitted. There are many algorithms and techniques in WSNs which change how the media or channel is accessed. These Media Access Control (MAC) protocols use a variety of methods and techniques where the common techniques include Random Backoffs [95], Collision Detection, Clear Channel Assessment or Carrier Sense Multiple Access (CSMA) [96], Time Division Multiple Access (TDMA) [97] and Orthogonal Frequency Division Multiple Access (OFDMA) [98]. For simplicity, this work will assume a Random Backoff scheme.

As opposed to an energy cost, the bandwidth is a *temporal* cost in that only one node's information can be transmitted and received in a moment assuming co-located nodes using a shared communication channel. Multiple co-located nodes can transmit data at the same time, but it is assumed in this work that this would result in loss of all data

transmitted at that time. Nodes which are not co-located can transmit data at the same time without loss, and thus the bandwidth cost is a spatially separate and individual cost to each node. As it is difficult to measure the spatial location of several nodes within an area, a factor relative to the density of nodes to define part of the bandwidth cost will be used.

$$\begin{aligned} I_{cb} &= T_{tx} + p_{col}T_{tx} \\ &= T_{tx} (p_{col} + 1) \end{aligned} \quad (4.15)$$

The  $T_{tx}$  variable is calculated from the amount of data to be transmitted (including any radio preambles), divided by the wireless network bandwidth, shown in equation 4.16, thus reflecting the time over the air the packet is occupying. The term  $p_{col}$  is the chance of collision occurring due to the packet being in the air on the transmission medium. We can compute  $p_{col}$  using equation 4.17.

$$T_{tx} = \frac{k}{R_{max}} \quad (4.16)$$

$$p_{col} = \frac{c_0 n}{R_{max}} \quad (4.17)$$

In these equations  $n$  refers to the number of nodes communicating in the range, while  $c_0$  is the collision factor to increase the chance of collision due to nodes outside the communication range, but within the Interference Range (IR) [99] of the transmission range. The collision factor is multiplied by number of nodes  $n$  as it is assumed that the density of the nodes within the communicating range is similar to the nodes within the interference range. It should range from zero to  $R_{max}$ .

Taking equation 4.16 and equation 4.17 and putting them into equation 4.15, gives us equation 4.18.

$$I_{cb} = \frac{k}{R_{max}} \left( \frac{c_0 n}{R_{max}} + 1 \right) \quad (4.18)$$

In terms of implementation, the node cannot instantly know how many nodes are attempting to transmit at the same time, and therefore cannot know the cost of transmission relative to the neighbouring nodes. The best case approximation is to implement an in-memory table about the neighbouring nodes which can measure the number of packets transmitted by the nodes over an arbitrary fixed period of time. This table could be derived from the first neighbour entries of a routing table.

Using the in-memory table, a count of the number of nodes can provide the value for  $n$ . The number of packets over a fixed time period can provide an estimation of how occupied the channel currently is, and thus produce an estimation for the  $c_0$ .

Normalisation of the bandwidth cost is done by looking at the maximum  $I_{cb}$  calculated over a recent time period. It is shown in equation 4.19.

$$I_{cb}n = \frac{I_{cb}}{\max(I_{cb})} \quad (4.19)$$

We have created equations for the costing of the information in both energy and bandwidth terms. These terms can be seen as the network centric resource usage of the information to be sent. There are several systems which employ a similar resource ‘ticket’ for the resources, such as Pixie OS [100], but these ‘tickets’ extend to all resources available to the sensor node, such as the processor, sensors and are mostly focused around energy.

The costing equations presented in this work, separates out the costs in both time and energy, thereby trying to pre-empt the direct energy costs which could occur by increased chance of collision, as well as trying to decrease the overall system latency, by incorporating the value vs the time costs.

In networks where multi-hopping is required, the cost of transmitting a packet through the network can increase greatly. Traditional WSN’s were perceived to have a lower total cost of transmission when implementing a multi-hop architecture, but this has been shown to be a fallacy as the cost required of listening and re-transmitting is greater than the initial cost of transmission. Under normal conditions it is much more energy cost-effective to transmit to the final destination as soon as possible with as few hops as possible. Multi-hop networks do have the advantage of being able to transmit an overheard message to further nodes which might not have had the range to hear the message and thereby increase the robustness of the network at a large cost. This cost is often under-estimated, and the energy cost of repeating messages is often larger due to only the transmission energy being taken into account [101–103], or the time listening and waiting with the radio receiver on not being taken into account [104].

The cost of information in a multi-hop networks increases greatly in terms of bandwidth and energy as the time occupied over the air is a multiple of the number of hops and the energy consumed is a multiple of the number of hops. Multi-hopping also has hidden costs, such as the extra time each individual node is required to listen for incoming messages in both synchronous and asynchronous networks, and the bandwidth occupation of the data over the air for nodes within the interference range. To implement multi-hopping, we need to modify the costing equations to include a parameter: number of hops,  $h$ .

$$I_{ce} = k(hE_{rtx} + E_{mcu}) \quad (4.20)$$

$$I_{cb} = \frac{hk}{R_{max}} \left( \frac{c_0 n}{R_{max}} + 1 \right) \quad (4.21)$$

The equations in 4.20 and 4.21 show the multi-hop forms of the information costing equations used.

The next chapter will explore these costs or benefits against a set metrics in simulations.

#### 4.1.4 Simulated Value/Costing using real data

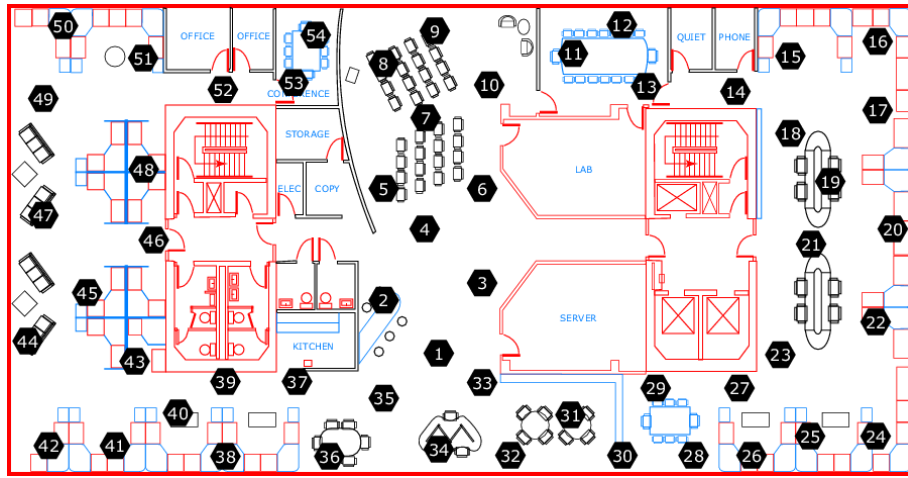


Figure 4.4: Intel Lab Research Mote Layout taken from [105]

To evaluate the information valuation and costing system, we simulated the system using the Intel Research Lab Data [105] shown in Figure 4.4. This dataset was selected as opposed to a BSN orientated dataset as it is difficult to find datasets which provide BSN data over long periods of time. It was decided that although the physically sensed parameters were different, the concepts of the network were similar enough to a BSN data to show that the system presented in this section worked.

The data has been used by numerous different papers to test algorithms similar to this [84] in energy efficient approaches. Data were collected from 54 sensor nodes using Mica2Dot nodes running TinyOS 1.0 and TinyDB. Data was collected once every 31 seconds from the sensor nodes. In the data, epoch information is sent which describes a time instant for the data sample relative to each node. Epochs are sequentially incremented by each node, and a missing epoch for a specific node describes information which has been lost to the receiver. It is unknown how or why the information was lost or not received, but assumed that the data packet was lost in transmission due to collision, other loss mechanisms, or removed from the dataset due to transmit buffer



overflow prioritisation described in [85]. Epochs are unsigned 16bit integers, and thus return to zero every 65536 epochs ( $\sim 23.5$  days). To analyse the data correctly, some data-correction operations were performed. These included:

- **Removing non nodes data** There are several measurements for nodes not in the dataset: 56,57,58
- **Removing erroneous values** There are several measurements which do not correspond to realistic values of measurements. These include humidities below zero, above 100% and temperatures above 100 °C. Measurements which contained these errors were discarded as other errors within the packets are likely.
- **Remove Timegaps** As the Epochs and timestamps are misaligned: the timestamps do not correspond sequentially to the epochs gathered from the nodes so that it is difficult to assess whether the timestamps match the measurement, and likely that the timestamp matches when the data was received. There is a large gap in the epochs around the 21st April. This appears to be due to the network being restarted or some other error with the nodes reporting from epoch  $\sim 29000$ . This is a critical flaw as the epochs overflow the maximum value and thus start from the beginning again. This creates uncertainty to which values the epochs are assigned.
- **Removed Node 26** This node has flawed data which is difficult to reconstruct into the dataset as the times and epochs of this node's data are duplicated over time and the epochs do not match any of the other nodes data points. To simplify evaluation which does not affect real-time results, this node's data has been removed.

Figure 4.4 shows a map of the nodes, while Figure 4.5a shows a spatial density map of nodes within the area. The spatial density map was drawn by accumulating the number of nodes within a radius of 8 meters of every point on the map. Looking at the spatial density map, allows one to visualise the concentration of nodes per unit area.

Using the aggregate connectivity data supplied by [105], we can draw a connectivity map of the whole network as shown in Figure 4.5b and compare that to the density map. The connectivity data is a set of three points indicating the transmitter, receiver and the connectivity between the two. The connectivity is an asymmetric relationship providing the “probability of a message from a sender successfully reaching a receiver” [105]. This map is drawn by overlaying lines between pairs of connected nodes with weighting that of the given connectivity between the two nodes. This map uses only the connectivities which have a probability greater than 0.1 as it assumes that a connection with a probability of a packet reaching the receiver of less than 10% does not represent a connection. This modification merely highlights the well-connected areas. The connectivity map

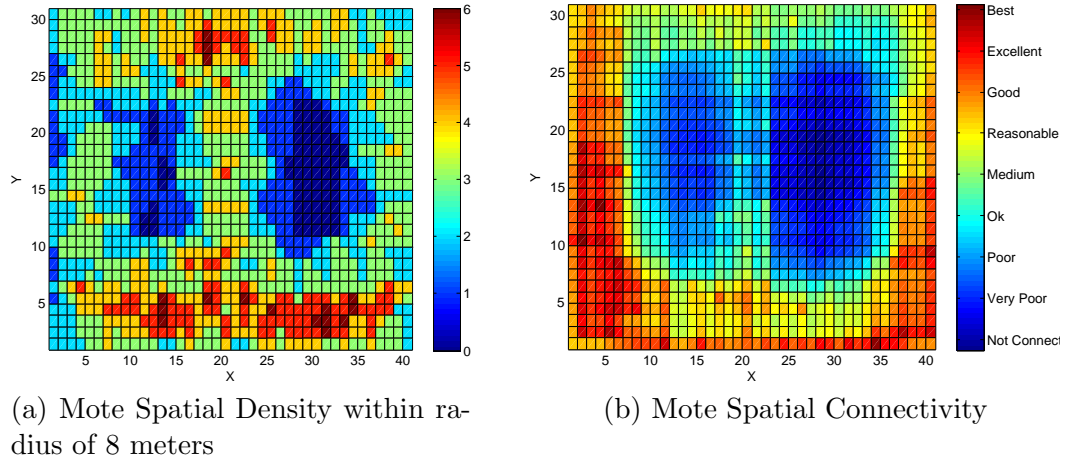


Figure 4.5: Intel Lab Dataset Details

thus shows a areas which are well connected, and highlights spatial areas which have poor or no connections across them.

Comparing the two maps in Figure 4.5, we can see that there are two large blue non-connected spots where there are no nodes. This shows that connections do not exist across these areas as well. The comparison shows that there is poor to reasonable connectivity along the top of the map where there are 3-6 nodes; there is reasonable to good connectivity along the side-edges of the map where there are 2-4 nodes; and there is medium to reasonable connectivity along the central bottom area with a high density of 4-6 nodes. This shows that there is only some correlation of dense nodes and reasonably high connectivity along the bottom of the map, while the area of the central top of the map shows reasonably dense nodes with poor connectivity.

From the comparison of the connectivity map to the density map, one can evidently see that the intuitive notion of adding more sensor nodes to increase connectivity is not necessarily true. It shows that a complex relationship exists between connectivities between nodes and their placement. To simplify this relationship, the connectivity of the nodes can be related to the cost of energy consumed in transmitting a message and the cost of bandwidth used when transmitting a message. Thus in areas of poor connectivity, the amount of energy used to transmit a message is higher (To increase the SNR), while the bandwidth is also higher (As the message needs to be re-transmitted).

Assuming this simple relationship between connectivity and information cost, the connectivity map shows areas which are poorly connected are expensive in terms of information cost. This is independent of the sink node and radio range as the data provided is between pairs of nodes and nodes with poor connections (below 0.1) are not shown in the map. Although it might be better to directly model the relationship of the connectivity between nodes and compare the findings against the dataset, this map provides an adequate representation of the cost of information.

From the connectivity and spatial density maps, and knowing that the nodes transmit data at a set rate of 31 seconds, we can infer the areas which are expected to have the greatest bandwidth occupancy, and thus the highest rate of packet collisions. This area is located in the bottom half of the map, and investigated later in subsection 4.1.4.2.

Figure 4.6 shows the individual averaged signals of each sensor type over all time.

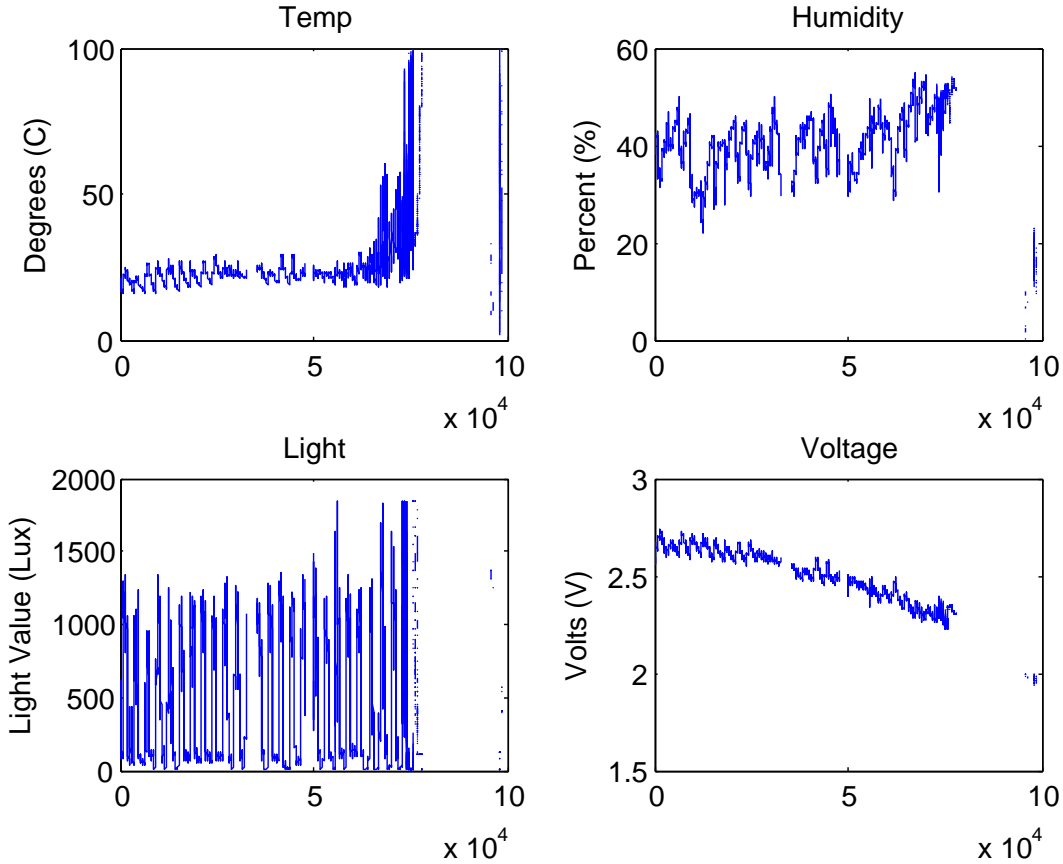


Figure 4.6: Individual measured signals averaged over all time

By accumulating the packet count every hour, we can visualise the bandwidth occupancy over time. This is shown in Figure 4.7. This figure shows the decline in the bandwidth occupancy over time, and two significant ‘dead’ spots where very little or no data was received. Although it is uncertain why no data was received during this time, it does point to a critical failure point in the whole network. This type of critical failure is not explained in the dataset, but could be attributed to a failure close to the sink node of a node which relays large amounts of data - a weakness common to tree-based networks and Scale-free networks [106]. The bandwidth plot also shows three bandwidth spikes associated around days 4, 10 and 12. The two lines shown in this Figure show corrupted and uncorrupted packets. The bandwidth graph also shows that uncorrupted packets captured over the time period do not correspond well to the full time-period, and thus values do not complete the full 38 days, but rather a shorter 25 day period.

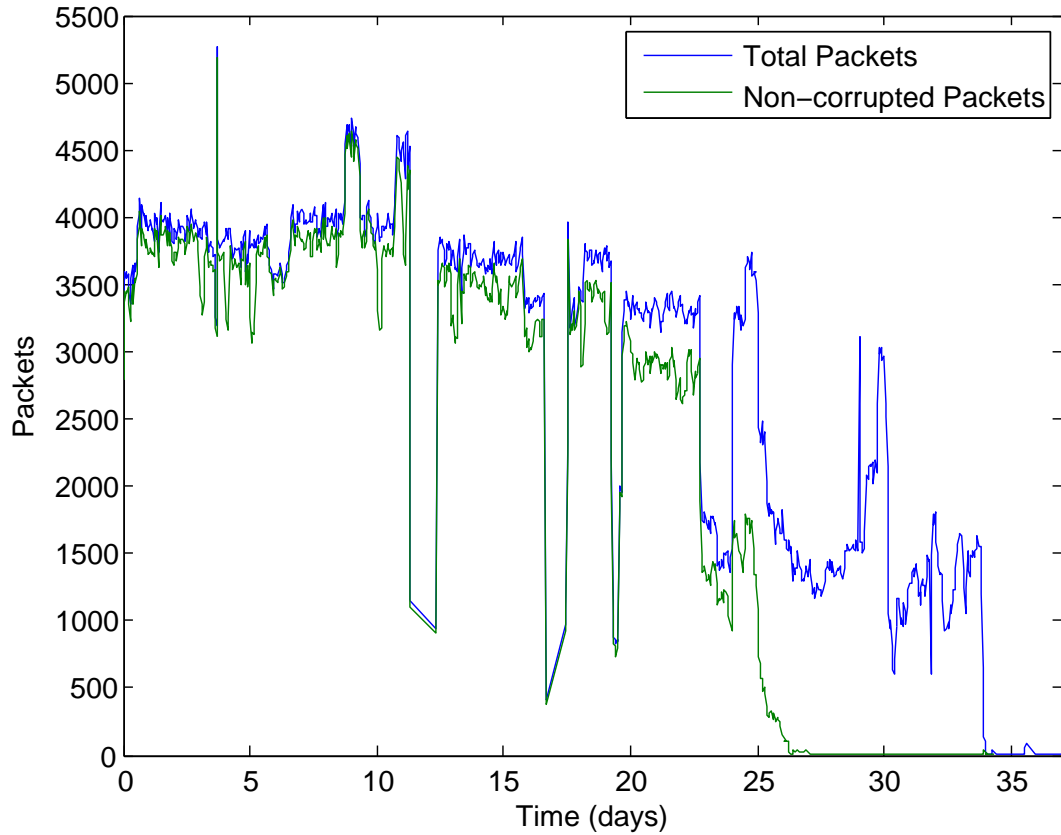


Figure 4.7: Overall Bandwidth Occupancy for entire network show in packets per hour

Investigating each data sample's time captured relative to their epoch number shows a few erroneous areas where epochs do not correspond to the times which the datapoints were captured. Under normal conditions, we can see that the nodes are time-synched to under 17 minutes accurate, and often epochs are accurate to under 10 minute differences.

To understand the output of each node, a graph of the error packets received, packets received and missing epochs from the dataset are shown in Figure 4.8. In this graph, each node is assumed to have measured each datapoint at a 31 second interval over a period of just over 35 days calculated from the date and time of the first value received to the last value received giving a total epoch count of 98874. The missing epochs is thus a count of datapoints which the sink node received no data against. An error packet is defined as a packet which contains unrealistic values for the dataset as specified in the beginning of section 4.1.4. In the dataset, several duplicate epochs were received for the same node and thus the missing epochs is shown to describe what data is missing from each node. In this graph, it is interesting to note that the nodes numbered between from the mid-twenties to late thirties appear to have the highest count of error packets, representing the nodes positioned in the bottom central area. The nodes numbered in the late forties have the lowest count of error packets along with the most data received and thus represent the most successfully connected nodes. These nodes are positioned on the far left hand side of the map.

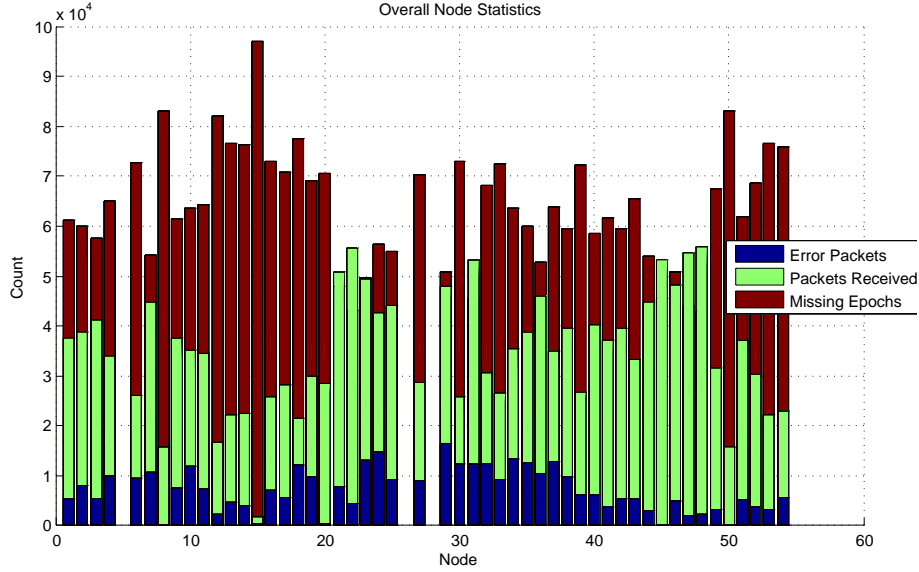
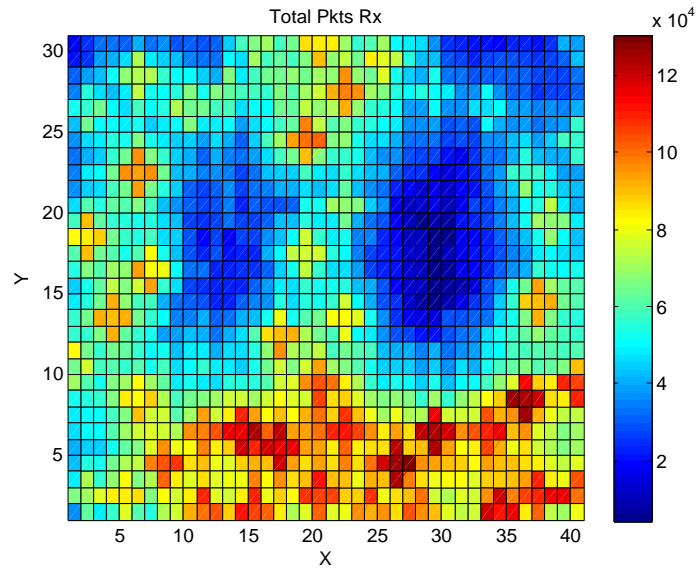


Figure 4.8: Node Statistics taken from raw data

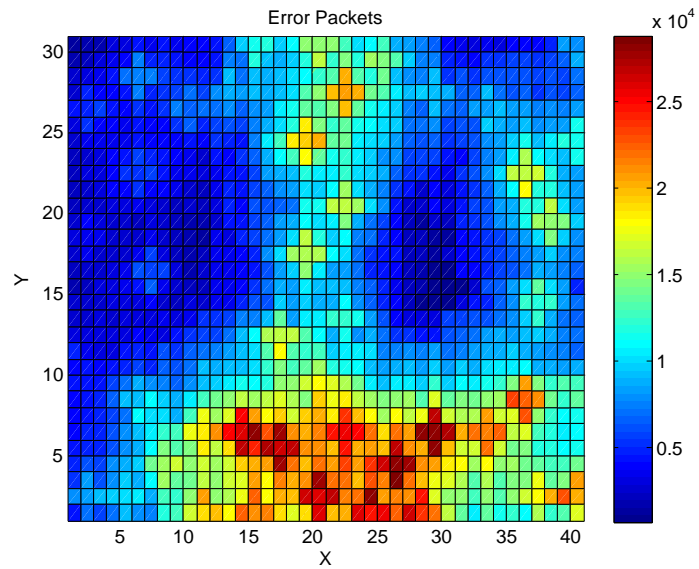
Figure 4.9a shows the total packets received by the nodes relative to their position. This map was generated by plotting the number of packets received from the nodes against the positions of the nodes. Each node has a spreading function within 8 meters of the node, equal to the inverse of the distance to the node, and thus the map is biased by node density. From this map, we can see the area where the most packets were received during the data collection. This area occurred mostly in the lower centre of the map, while the top horizontal subsection appeared to have relatively few samples corresponding to that area. By comparing this map to the density map in Figure 4.5a one can see that the map corresponds reasonably well in the bottom central area, but the top right hand corner does not correspond well at all and the top central only corresponds slightly. This map also shows little correlation to the connectivity map shown in Figure 4.5b.

Figure 4.9b shows a spatial representation of the error packets received. An error packet is logged at the sink, and is defined as a packet containing an unrealistic value: such as a temperature over 100°C or humidity below 0%. A similar spreading function was used as in Figure 4.9a for image clarity. If we compare this map to the packets received map (Figure 4.9a), we can see that although the sink received a large amount of data from the bottom central area, the same area contained a large proportion of error packets. The error packets are close to the assumed sink node (numbered one), and this could explain the distribution of the error packets as the sink could overhear erroneous packets, but other nodes relaying messages would not relay erroneous packets. Another explanation would be linked to the increased noise of packets colliding due to the increased number of nodes being within the interference range.

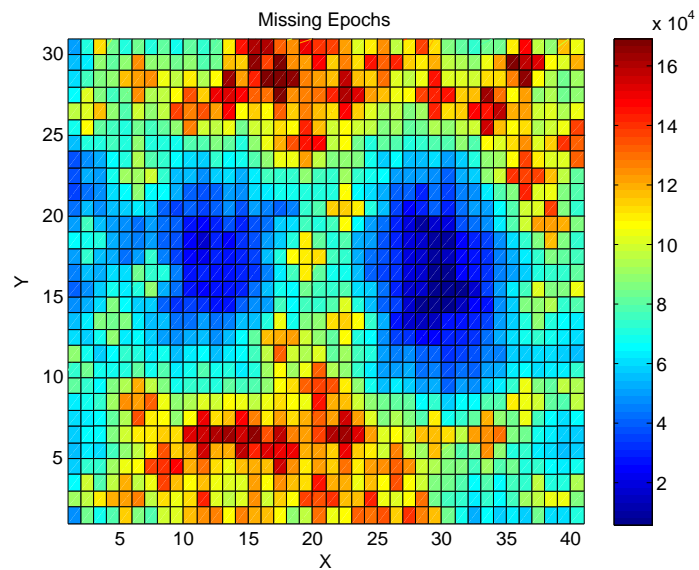
Figure 4.9c contains a map of the missing epochs for each node. The missing epochs were calculated in Figure 4.8 and a similar spreading function to Figure 4.9a is used.



(a) Total Packets Received Map including error packets



(b) Error Packets Map



(c) Missing Epochs Map

Figure 4.9: Packet Maps

Looking at this map, one can see a large number of missing epochs in the top central to top right corner area and bottom central to left corner area. One can also notice that the bottom right hand corner and the left hand side of the map have few missing epochs which corresponds well with the high connectivity associated with these areas shown in Figure 4.5b. By comparing the error epochs map to missing epochs map, one can see that although more packets were received in the lower central area, it did not greatly improve the total number of epochs received. A comparison of the missing epochs to the total packets received map shows that the top right corner which received very few packets was missing similar numbers of epochs as the bottom central area which received large numbers of packets, but were highly erroneous.

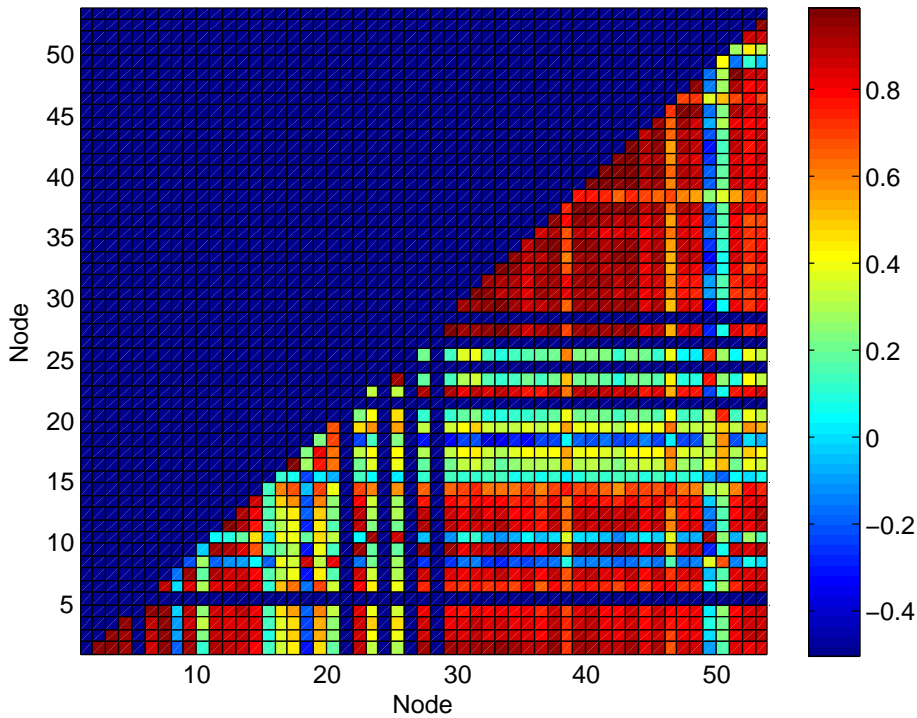


Figure 4.10: Matrix Map of Node correlation

Figure 4.10 shows a map of the overall correlation of the sensor nodes where a value of 1 indicates perfect correlation between a node pair. The correlation value was extracted between pairs of nodes using the Matlab `corrcoef` command for the individual readings (temperature, humidity, light and voltage) which were then averaged for a total correlation value. The map was initially set to -0.5 so that any value at this level contains no values for the node pair. From this map one can see a large amount of correlation over the entire data set by looking at the number of values above 0.5. The two largest area of “redness” indicating high correlation are found in the bottom right between nodes 30 to 48 and nodes 1 to 15; and in the top right hand corner for the nodes between 30 and 48. These nodes are spatially located in the bottom central to left hand side area of the map. Nodes which have negative to poor correlation are nodes 8, 10 15 to 20, 49 and 50. These nodes are all spatially located on the top half of the node map. Some of the

actual correlation values of these nodes are shown in Table 4.3. The whole table is not shown as it contains 1176 rows of correlations.

Table 4.3: Some Correlation Values for Node Pairs

Node 1	Node 2	Humidity	Voltage	Temperature	Light	Total
8	37	-0.1709	0.4038	0.4406	0.4187	-0.0127
8	30	-0.1563	0.3535	0.4113	0.3073	-0.0070
8	31	-0.1360	0.2922	0.3952	0.2708	-0.0043
20	30	0.1925	0.8190	0.4978	0.4693	0.0368
20	31	0.2395	0.8086	0.4747	0.4377	0.0402
20	37	0.1765	0.8266	0.5455	0.6105	0.0486
8	20	0.8233	0.3578	0.8245	0.7470	0.1814
30	37	0.9048	0.9669	0.9412	0.8426	0.6938
31	37	0.9218	0.9566	0.9007	0.8821	0.7007
30	31	0.9757	0.9898	0.9817	0.9364	0.8878

From the dataset, the node-pair with the lowest correlation (nodes 15 and 49) and the node-pair with the highest correlation (nodes 1 and 33) is shown in Figure 4.11. In this figure, one can see that Node 15 only has data up until around epoch 10000, while the other three nodes signals are mostly correlated over the entire time period.

From these graphs, we can clearly see there is correlation between many nodes in the data set. This implies a large amount of redundant and unimportant information is being transmitted across the network continuously. We can also see from looking at the data that the majority of packet errors are from areas where the node density is high and the data correlation is high. These graphs also show that the central top to right hand corner area, which can be seen in Figure 4.9c has several nodes with the most missing epochs which is an area where the connectivity is poor and correlation is low (Nodes 8, 15-20).

Temporally, samples have been captured much higher than the required sampling rate to reconstruct the signals. This creates a great deal of unimportant data transfer at the cost of precious energy resources. There is a great deal of spatial correlation in the system, showing that densely co-located nodes are not all required to transmit their data, as their signals are extremely similar across all time.

Using the data provided in this section, we can draw up a table to estimate statistics about the data captured over this field in time and space. The data is given in table 4.4. Using these values allows us to estimate the overall effectiveness of the data capture over a spatio-temporal field under energy and bandwidth constraints.

We wish to establish metrics to measure the effectiveness of a sensor network over a certain area over a certain time period. Basic metrics to measure this are accuracy,



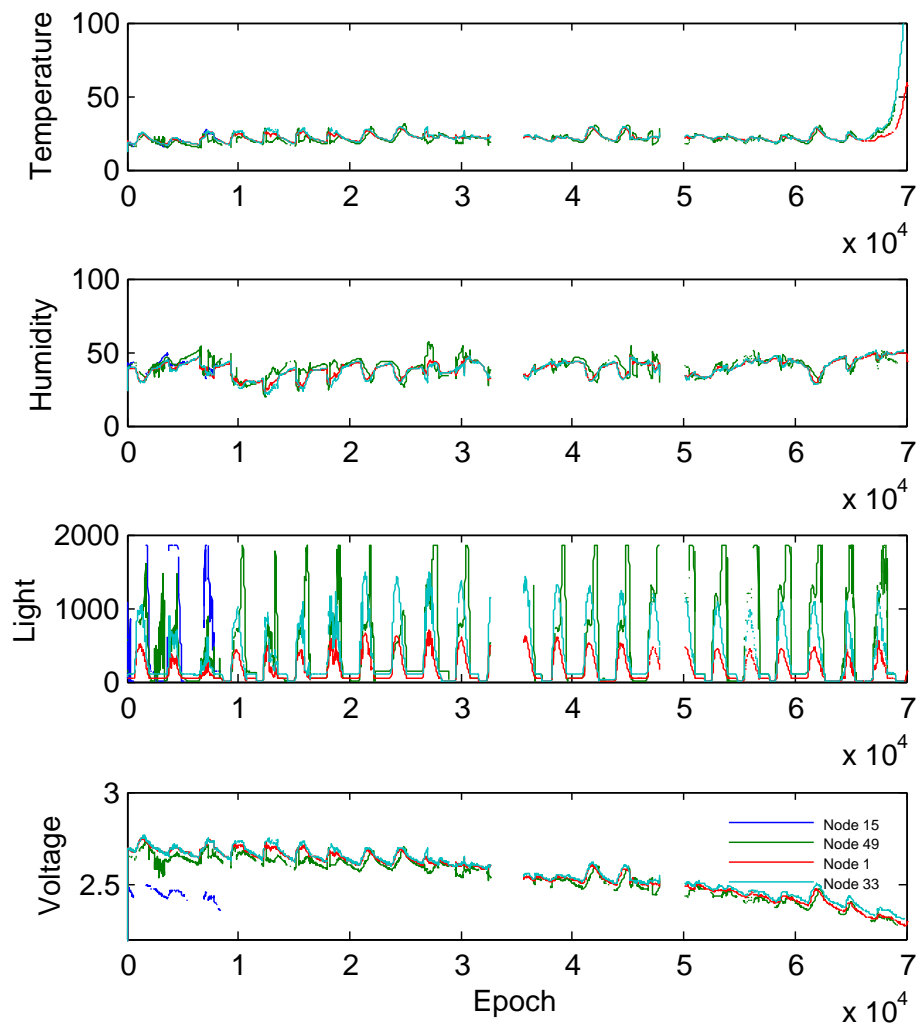


Figure 4.11: Comparison of best (Nodes 1 and 33) and worst (Nodes 15 and 49) correlated nodes

Table 4.4: Network Statistics

Time	
Start Date and Time	28 Feb 2004 00:58:48
End Date and Time	05 Apr 2004 11:14:27
Epoch	
Calculated No of Epochs	5 339 196
Total Missing Epochs	1 492 647
Mean Lost Epochs per node	27 139
Mean Error Packets per node	6 469
Packets	
Total Readable Packets	2 219 799
Total Error Packets	388 184
Mean Packets per node	40 360

resolution in space and time and energy- and bandwidth- consumption. Another important factor to consider in the evaluation of a network is how fair the network treats participating nodes. Accuracy and resolution are related to the information value, while energy and bandwidth consumption are related to information cost.

To evaluate the proposed system, we first need a gold standard against which to evaluate it. As the data provided is reasonably ‘raw’ with large missing sub-subsections (epochs) we are required to reconstruct the data from the sample points available. We need to assess how much information is contained in the data, how well the data can be compressed losslessly and if we were to capture the information lossily, the error in the signal in relation to the original signal.

To measure the information value, the shannon entropy [69] of the dataset is investigated using equation 4.22 and the entropy distribution in Figure 4.12a. The signals are then constructed from the low frequency Fourier components and compared to the original signals to get an understanding of the RMS error of the FFT/iFFT which is shown in Figure 4.12b. The number of low frequency components are shown in the x-axis of Figure 4.12b.

$$H(X = x) = \sum_{i=1}^n p(x_i) \log_b \left( \frac{1}{p(x_i)} \right) \quad (4.22)$$

To investigate the information cost of the data, we first look at the energy and bandwidth used in capturing the raw data, and compare that to the information cost and bandwidth cost of the data collected.

Assuming that the cleaned data is the entire data set, we can calculate the information content and entropy of the data received in the network. Figure 4.12a shows the entropy distribution of the values for the four captured signals for all the nodes over the whole time. We can thus calculate the amount of data generated from the network for each measurand and the average entropy of each sample in bits. This is shown in table 4.5. The entropy shows the minimum number of bits in which each sample can be stored to ensure lossless transmission of the entire signal. It should be noted that in this information set, we are not recording data related to the time, position and location of the nodes, as this would have an entropy of zero.

Table 4.5 shows that the minimum capacity in which we can store and thus transmit the complete data set is thus the sum of those four components: 6MB.

As we are not attempting to transmit and store the full signal losslessly compressed, we require a factor of increasing and decreasing the information to gracefully increase/decrease the value of the information without compromising the signals. By taking the FFT/iFFT of the signals using different component counts, we can analyse the difference of the signals produced at input and output to see how much important information

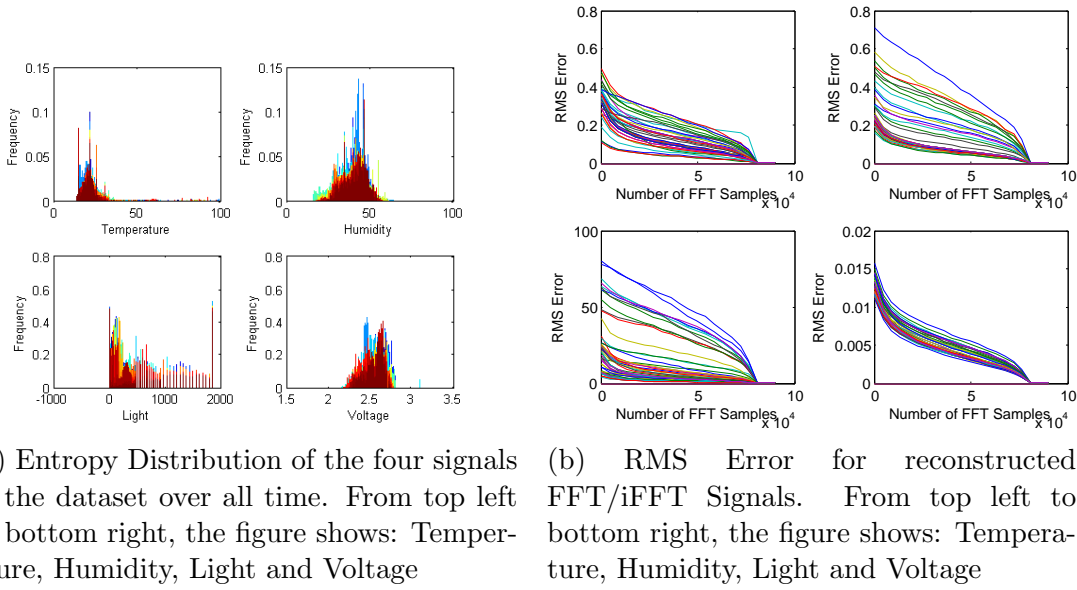


Figure 4.12: Entropy and RMS Error of Intel Data

Table 4.5: Overall Averaged Information Entropy of all node information

Measurand	Number of Symbols	Independent Sample Entropy (bits)	Data (MB)
Temperature	8287	10.03	2.025
Humidity	1447	9.05	1.827
Light	136	5.28	1.067
Voltage	80	4.90	0.989

is contained in the signal. The RMS error values between the original signal and the reconstructed signal are shown in Figure 4.12b. From these values, one can see that the RMS continually decreases as the number of FFT samples used increases. One can see reasonably good reconstruction (RMS Error below 0.3 for all nodes temperature; RMS Error below 0.5 for all but one nodes humidity; RMS Error below 75 for all nodes lux level; and RMS error below 0.012 Volts for all nodes) of the signal if one uses the number of samples at the first knee on the graph in Figure 4.12b at around 10000 FFT samples for all the signals. Increasing the number of samples beyond this point does not decrease the RMS error beyond linearly until around 90000 samples where the RMS error quickly falls to zero.

The dataset contains a large amount high frequency noise which can be removed by using the largest set of frequency components of the signal. The high frequency components can be regarded as noise as the signals (Temperature and Humidity) being monitored, do not physically change at the frequency of the components associated with the noise, these signals are inherently lower frequency signals changing slowly. Investigating Figure

4.12b shows that using approximately 1500 samples reduces the majority of the noise in the signal and produces reasonably accurate reconstructions of the signals of which any further increase in samples does not decrease the RMS beyond linearly.

#### 4.1.4.1 Information Valuation

Using the model defined in Section 4.1.2, the most valuable temporal nodes are shown in Figure 4.13, while the spatially valuable nodes are shown in Figure 4.14. In Figure 4.13 the map is generated using the temporal information model shown in Section 4.1.2.1 using a length for  $m=1$  and the value of  $\mathbf{a}=1$ . As the temporal samples are sporadic in the dataset, but this would not be the case for the data stream on the nodes, the data was first interpolated for each node and then valuation was performed. In Figure 4.14 the map was generated using the model shown in Section 4.1.2.2 with the values of  $\mathbf{d}$  being equal to 0.5.

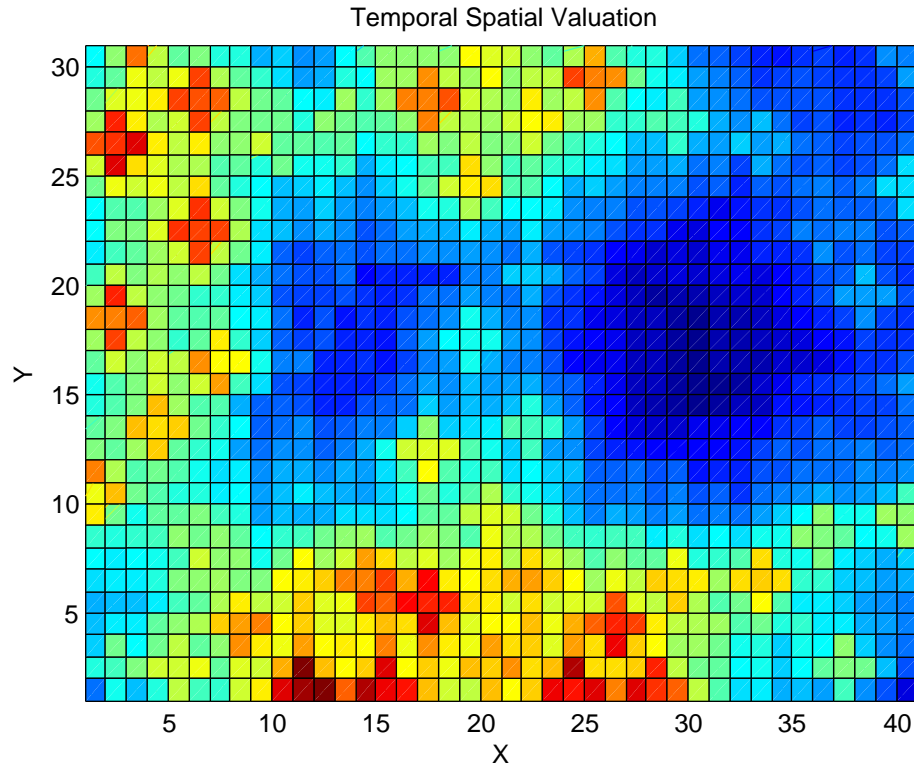


Figure 4.13: Temporally Valuable Nodes over total time

It is clear from these maps, in comparison to Figure 4.9a, that the most data is not obtained from the nodes which provide the most *valuable* data in terms of the models defined in Section 4.1.2. There are three specific areas of value in the dataset being the top central area, the top left hand corner and the bottom central area. To optimise the network to provide the most valuable data, these nodes should have the most data received as they represent the areas which provide the highest change in information over time and space.

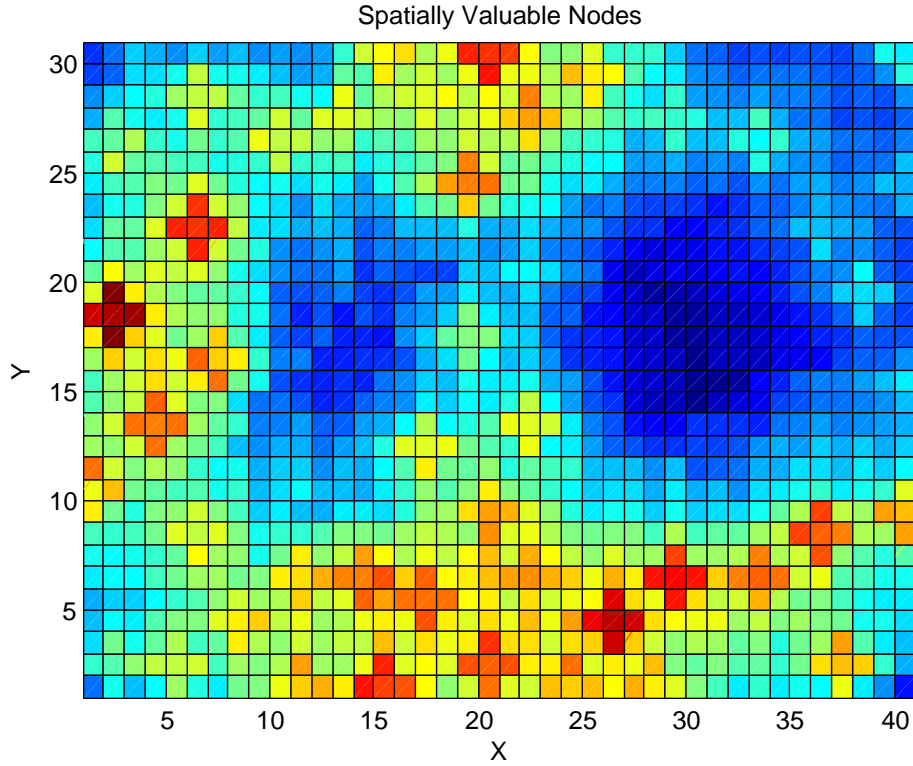


Figure 4.14: Spatially Valuable Nodes over total time

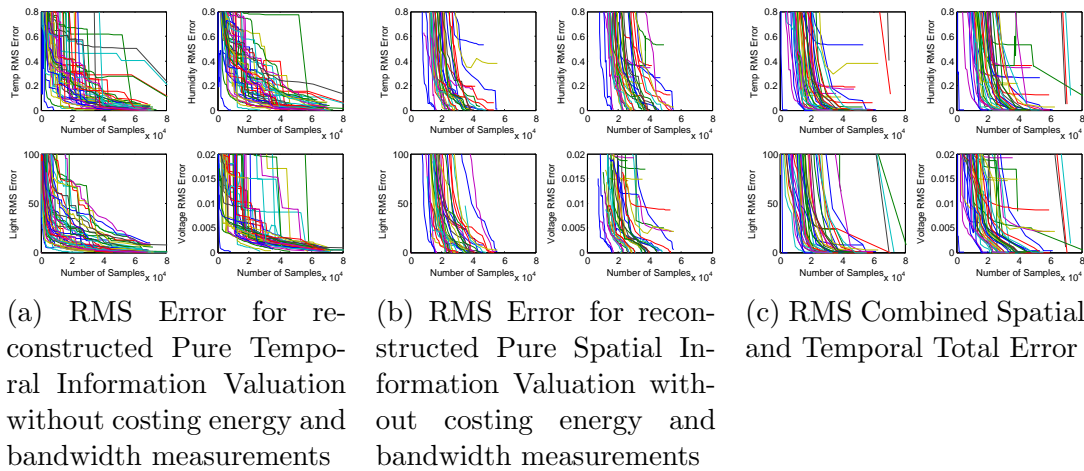


Figure 4.15: RMS Errors of Temporally, Spatially and Total signals

If one were to reconstruct the signals from only the most temporally valuable signals using a similar thresholding technique as in Figure 4.3, one can evaluate how much RMS error is associated with the reconstructed signals. The RMS errors in comparison to the number of samples used is shown in Figure 4.15a. In this figure and figures 4.15b and 4.15c the RMS error contains persistent errors when comparing to Figure 4.12b due to the small accumulated difference of the final values between the reconstructed signal and original signal.

The temporal information value signals have been reconstructed from the received signals, while the lost signals which have been transmitted and not received cannot be recovered from this dataset. Therefore some high frequency signal components associated with these signals is also missing. To construct the graphs made in Figure 4.15a, we interpolated the temporal data from the received data, and assumed that each of the nodes knew their own previous data values.

The Spatial RMS error for the pure spatial valuation signals is shown in Figure 4.15b. In this graph, we only used signals which were received at the base station node, and thus guaranteed that the nodes data was received. To ensure enough data points were available for each epoch it was assumed that the nodes were all co-located and thus data from all nodes could be overheard. It is noted that later in this section multi-hopping is assumed, but this does not affect the overall system greatly as the signals are highly correlated as can be seen in Figure 4.10.

Looking at the RMS error of the signals as we adjust the information valuation metrics in terms of packets, allows us to view the changes in quality of signal as we change the amount of information value we allow through the network. This system of valuation shows that the system does generally work by using a threshold to increase the temporal and spatial value results in decreasing the amount of error in the signal. It should be noted that the signals are not nearly as well bounded as the FFT signals, showing that signals which have few data points which are very similar to each other might not achieve good results. We can also see that the compression is not nearly as good as an FFT/iFFT at low numbers of samples, but does improve and produce similar error rates when the packet count is around 20000 samples, or half of the average of samples sent by each node.

Although the information valuation is useful to decrease data redundancy and allow the node to precompute a value describing its data thus decreasing its energy usage, the spatial position of the node and its associated connectivity affect the energy usage associated with successfully transmitting the data to the sink. It is also clear that the network connectivity and placement of the nodes affects the data throughput, as a poorly placed node would need several retransmissions to successfully send a single packet. Nodes transmitting at the same time within the Interference Range further affect the ability of the node to successfully transmit its data. In an attempt to compensate for

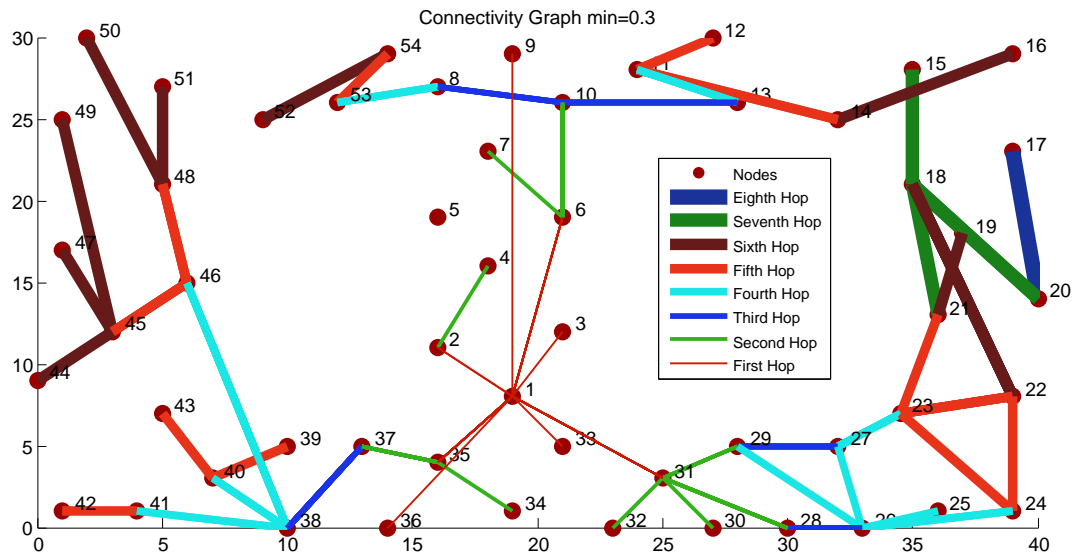
these dispositions of spatial placement and dense nodes within the interference range, the following section considers the additional impact of including the costing elements of the model described in Section 4.1.3.

#### 4.1.4.2 Information Costing

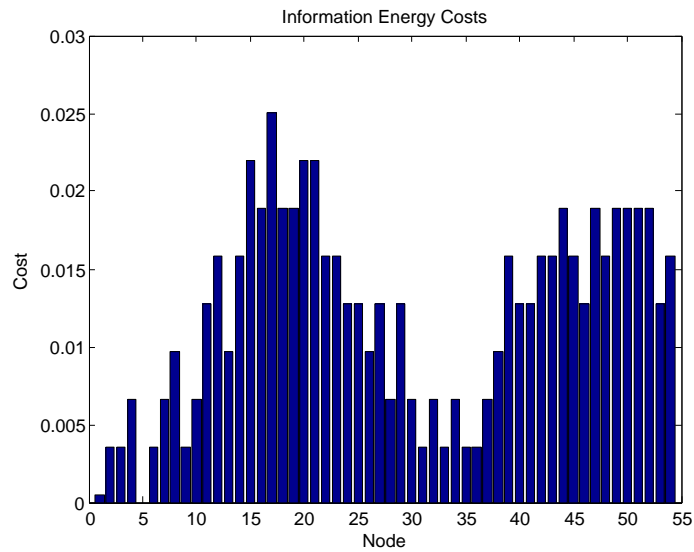
Costing the information is relative to the physical energy cost of capturing and transmitting the data, and the energy cost of transmitting the data within a noisy and congested environment. The first costing metric is generally constant with relation to the packet length, hardware and number of hops, while the noisy and congested channel changes with each epoch as the node is uncertain of how many nodes within the Interference Range (IR) are concurrently attempting to transmit. The node therefore needs to keep record of how much information is being transmitted within its environment to estimate the bandwidth congestion. If a node fails to transmit a packet to the destination, it is required to assess whether the value of the packet is valuable enough to re-transmit it in the network.

As we do not have the RSSI or Power Transmitted information in this dataset, we infer this information from the connectivity data provided by [105]. From the data, we fit an approximate exponential curve to the data and estimate the bandwidth congestion relative to the distance.

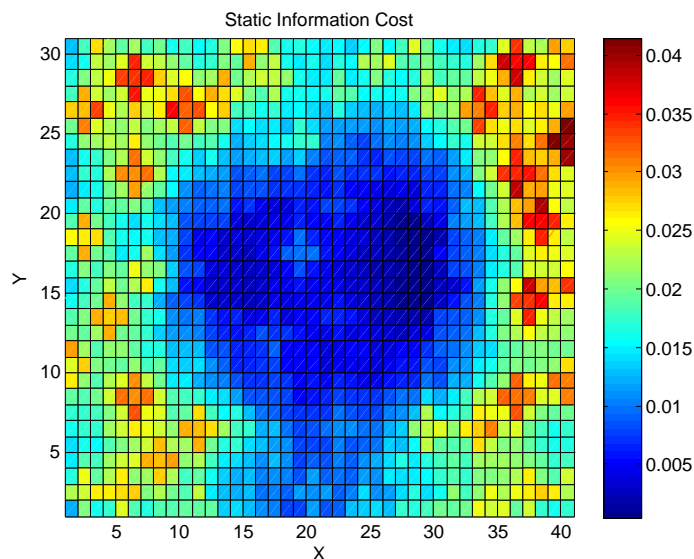
To calculate the information costs, an initial value for  $c_0$ , the collision factor, is selected as 0.5 as approximately half of the epochs are not successfully received for each epoch. Number of nodes,  $n$  is selected as the number of nodes participating in the network: 54.  $R_{max}$ , the wireless network bandwidth, is selected as the Mica2Dot maximum data throughput of 38.4Kbaud or 4800 bytes per second.  $k$  the data transmitted at each time step is calculated as two values, namely an upper and lower bound to test the data throughput. The upper bound of  $k_{data}$  is computed using Table 4.6 to give the maximum size of 37 bytes and the minimum size of 12 bytes. This gives us a maximum data throughput of  $129 \rightarrow 400$  packets per second which is clearly well above the maximum transmit rate which we successfully achieve as shown in the dataset. The difference between the known data collection rate of 54 packets for every epoch of 31 seconds and the total dataset received is attributed to packet loss in the network. This packet loss can be attributed to several different causes including signal degradation, multipath fading and channel congestion. It is assumed in this work that the loss of packets in the dataset is attributed to channel congestion [107, 108], as nodes are placed close enough together to have significant connectivity values; multihop packet forwarding is assumed with a polynomial increasing packet overhead as described in Section 4.1; and all nodes are attempting to transmit synchronously without any channel congestion avoidance.



(a) Approximated number of hops for the dataset. Hops are defined between node pairs with highest connectivity data and are thus not equidistant



(b) Minimum static node cost per node



(c) Information Cost Map

Figure 4.16: Information Costs



Table 4.6: Costing Parameters

	Upper Bound (bits)	Lower Bound (bits)
<b>Header</b>	<b>72</b>	<b>24</b>
Source Address	16	8
Destination Address	16	-
Type	8	-
Group	8	-
Length	8	-
CRC	16	16
<b>Data</b>	<b>224</b>	<b>72</b>
Date and Time	64	-
Epoch	16	16
Mote ID	16	8
Temperature	32	16
Humidity	32	16
Light	32	8
Voltage	32	8
<b>Total Bits</b>	296	96
<b>Total Bytes</b>	37 bytes	12 bytes

To estimate  $d_{max}$  we use the connectivity data along with the distance map and calculate that at our maximum distance for connectivities around 10% is around 25 meters. As we assume fixed packet lengths,  $I_{ce}$  does not change between packet transmissions and is thus a stored, pre-calculated value onboard on the node.

To estimate how much energy is used from each node over the entire time period, one can investigate the dataset provided average voltage data shown in Figure 4.6 and plot a decreasing linear line fit to that data. This fit gives us the co-efficients for the  $y = ax + b$  graph as  $a = -5.319e - 6$  and  $b = 2.717$ . This means that the average starting voltage for each sensor node is 2.7 volts and the volt drop after each epoch is  $5\mu V$ . By assuming the value of the batteries used, we can approximate the power lost in each epoch, and thus the total energy cost of each sample captured and transmitted. As the standard power source for the Mica2Dot motes is a CR2354 cell, these were used as the assumed power source. These batteries are generally known to have 560 mAh which equates to the nodes using a constant drain of around 990  $\mu A$ hour for the 23 day period.

Using this value along with the Mica2Dot Datasheet values for transmission current (tx) of 27mA and reception current (rx) of 10mA [109], one can calculate that if the device transmits 12 bytes (which takes 2.5 ms at 38.4Kbaud) for the voltage to drop at the calculate rate, the node listens for approximately 3.062s per epoch. Conversely if it transmits a 37 byte packet (which takes 7.7 ms), it listens for approximately 3.048s per epoch.

Although the energy cost estimation is reasonably static, the bandwidth cost should vary in a system were nodes compete for bandwidth. In this system, nodes theoretically transmit their data at set time intervals of 31 seconds. It is not shown in the data whether they implement a random backoff if the channel is being used, so an analysis of

the incoming data packets was performed to see how the channel is being used in terms of time. In a system such as the one proposed, the channel bandwidth is an important criterion due to the maximum bandwidth being used as a parameter in the estimation of the cost of transmission in the network. If we look back at Figure 4.7 we can see that there are several time periods in the dataset where little or no data was transmitted in the network. These periods would decrease the bandwidth cost, and therefore increase the throughput of data. It should be noted that in the Intel data, these periods of little bandwidth occupancy are probably due to network failure as opposed to little data being provided, but for the sake of example, if these were times when data was not being transmitted then these would decrease the bandwidth cost. Looking at the theoretical maximum bandwidth of the nodes over an hour, and assuming that the nodes are co-located, the maximum data rate of 38400 baud equates to 138.24 Megabits per hour which in turn relates to a range of 467.03 Kilo- to 1.44Mega-packets per hour, a much higher rate than the dataset throughput. This shows that the nodes do not operate close to the maximum of which the hardware is capable.

If the system implements multi-hop routing to a sink node or base-station, the energy cost of the node's information should increase significantly in proportion to the number of hops. This requirement forces each node to have an understanding of the hop-length from source to destination of its own data. In the Intel data set, we are not given data on hop distance of each node, but it can be estimated using the connectivity data and approximating the hops from this data. From the data we have assumed that the base-station node is node one, and nodes transmit their data back to node one. Using the connectivity data and the distances from each node, we can calculate the energy cost of each packet from the node sources to the sink node one. The approximated hopping distances are shown in Figure 4.16a.

Using the hop data along with the packet data calculated, we can graph the static information costs as shown in Figure 4.16. The static information cost is proportional to the number of hops, but has been normalised to be compared with the information value.

To calculate the dynamic information costs, we need to address the issue of how many nodes are simultaneously competing for bandwidth. Using the bandwidth information over a shorter timescale, gives an estimation of the global network congestion for a specific instant in terms of how many nodes are attempting to transmit around the same time. Using this data, as shown in Figure 4.17, we can estimate the network congestion and calculate the dynamic information costs for the network.

Having calculated the network node costs, we can compare these to the values obtained by the sensor nodes, and visualise the signals generated by the nodes.

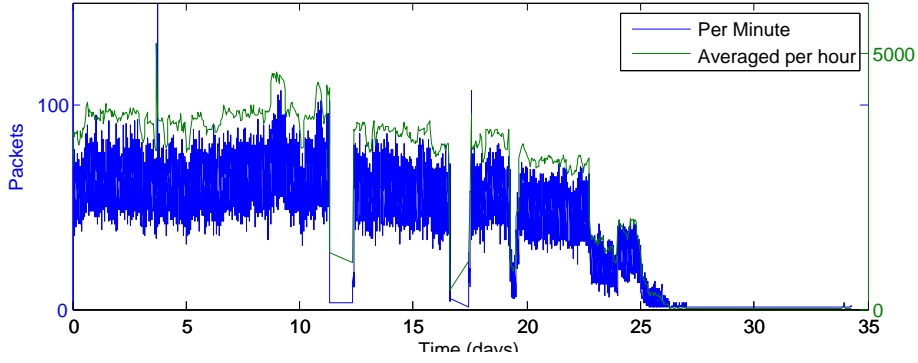


Figure 4.17: Bandwidth in Minutes

#### 4.1.4.3 Data Results

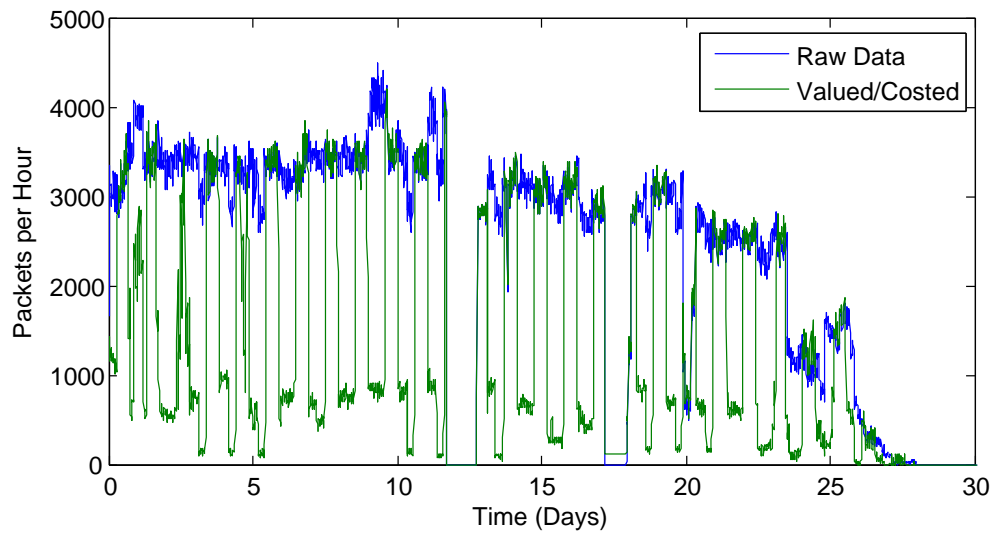
Overall results of running an information valuation and costing is shown in Figure 4.18.

Figure 4.18a shows the number of packets received from the Intel lab dataset in comparison to the information valuation and costing received packet dataset. This figure shows how packets are transmitted over time change as the value of the signals change. Figure 4.18b shows the per node received packets highlighting which nodes transmitted the most packets, as well as which nodes sacrificed the most packets. In Figure 4.18c we see a spatial map of the received packets. It can be seen that the overall packets received is much lower than the previous maps, and most of the packets are received from the inexpensive nodes close to the sink.

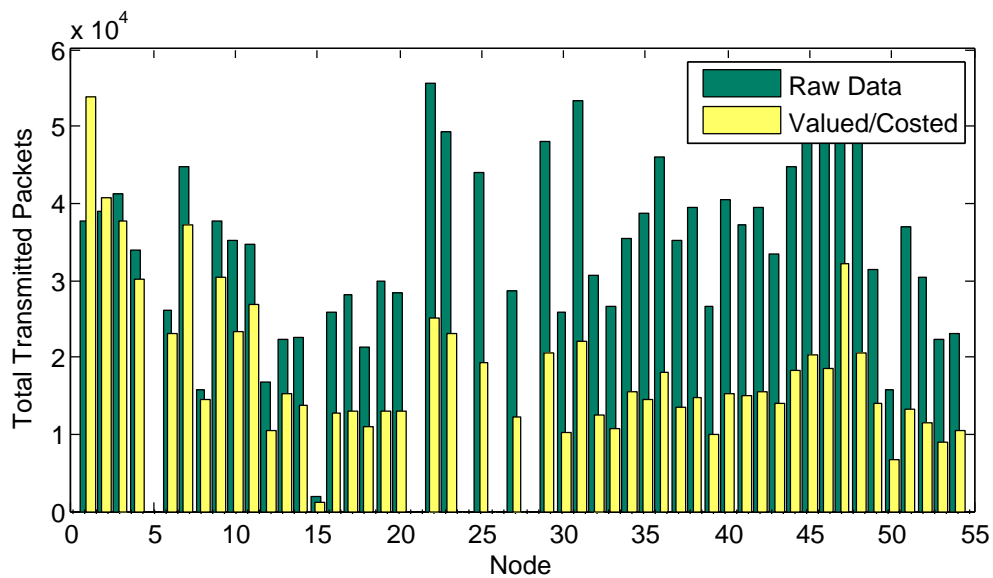
Figure 4.19a shows the RMS error of this valuation/costing system for each node's data. The RMS Errors are divided into four different views showing each of the temperature, humidity, light and voltage signals. It can be seen that there are several specific outliers in the data, which have been caused by failure states of the nodes. This is clearly visualised in Figure 4.20 which shows the failure condition in Node 11 of the humidity signal. In this figure it can be seen that the node's RMS error is consistently low, until the end of the signal when a huge interpolation is required. As the final samples of change are not transmitted by the nodes (due to the cost/value system regarding the information as not important enough), the RMS error is greatly increased.

As the final epochs of the dataset contain very little data, which can be seen in Figure 4.7, removing the last datapoints, and computing the RMS Error over the more restricted 70000 epochs ( $\sim 25$ days) shows a more respectable RMS error in Figure 4.19b. In this dataset, the RMS errors are improved, but there are still outliers due to early failing signals.

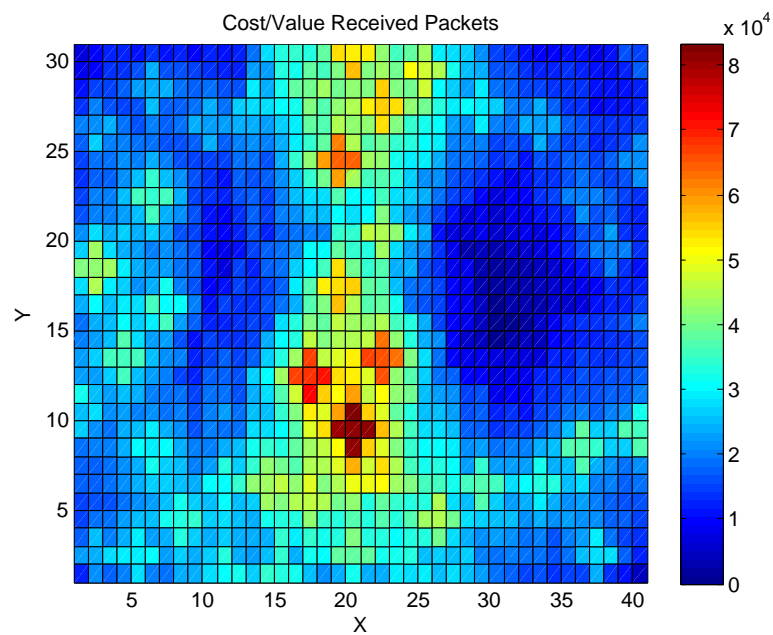
Looking at the RMS error of the data in terms of time in Figure 4.19c shows how the error increases towards the end of the data capture in the temperature and humidity signals. If one compares this plot to Figure 4.6, one can see that the majority of the error appears from the erroneous temperature readings captured by the sensors which



(a) Bandwidth Savings of this dataset

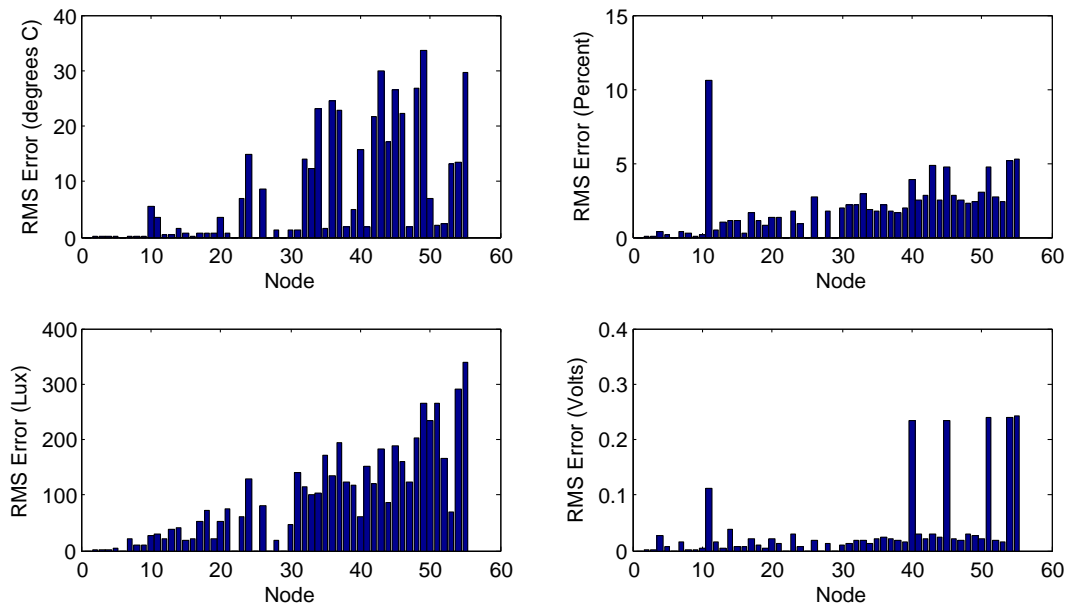


(b) Per Node Packet Savings

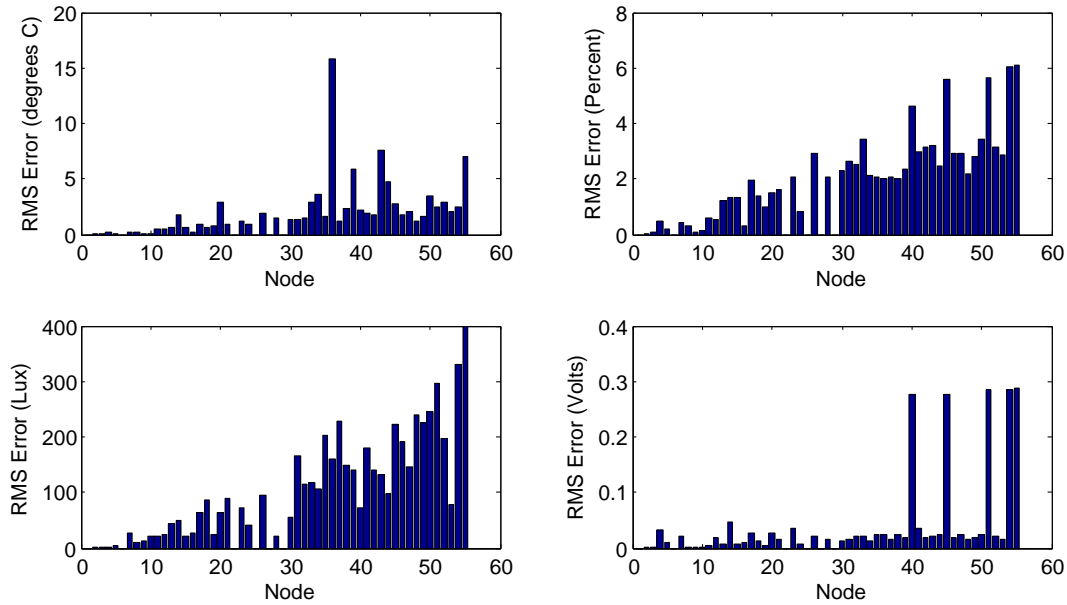


(c) Map of received packets

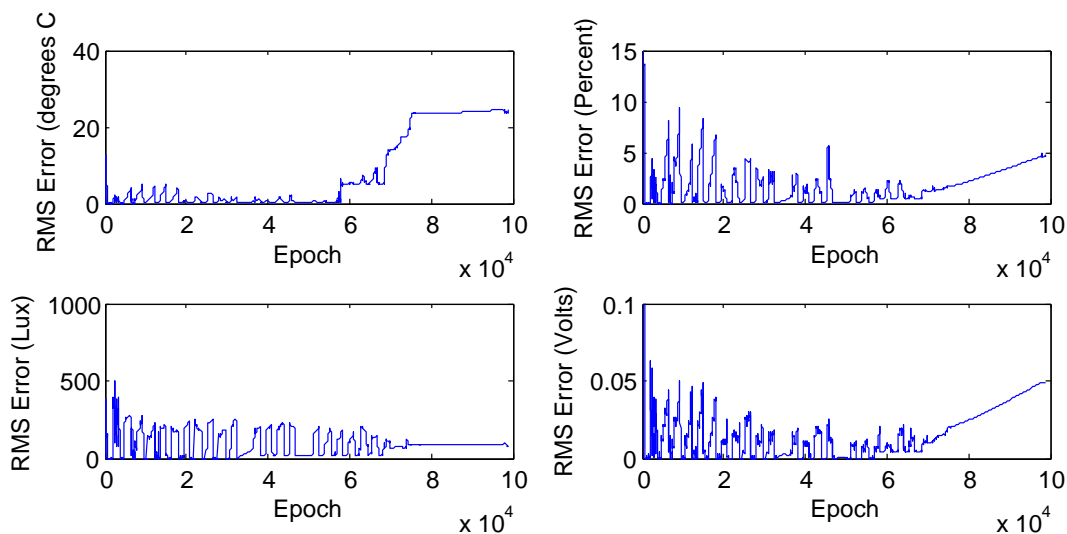
Figure 4.18: Results of Information Valuation/Costing system on all nodes



(a) Mean RMS Error of Node's signals



(b) Mean RMS Error of first 70000 epochs of Node's signals



(c) Overall Average RMS Error per epoch

Figure 4.19: RMS Errors

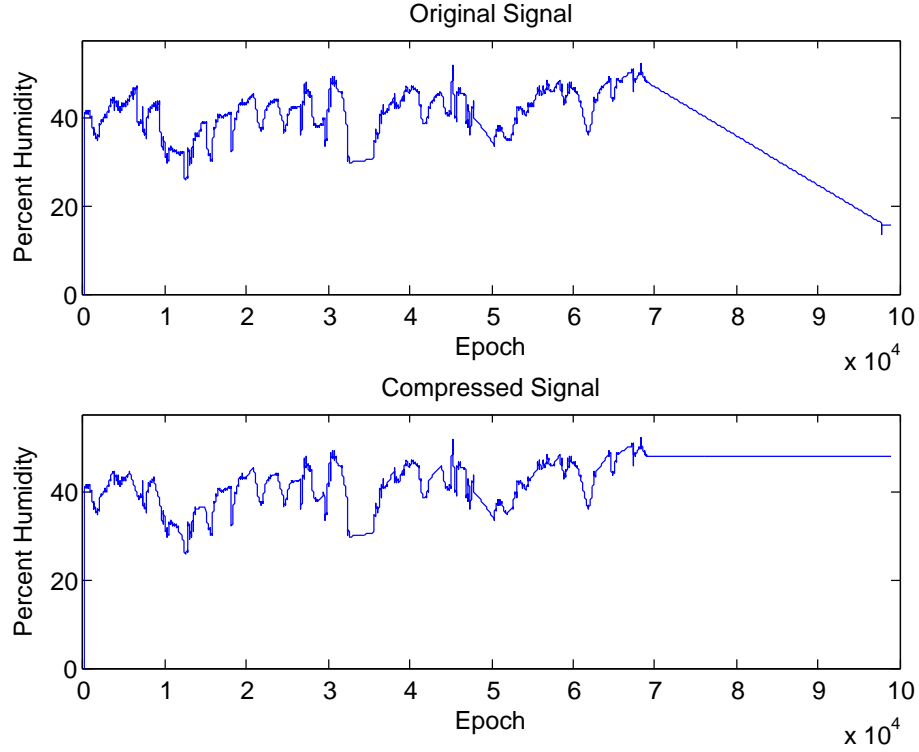


Figure 4.20: Node 11's humidity signal raw dataset (top) and compressed dataset (bottom) showing that the majority of the RMS Error is from a large deviation in the final packets due to assumption differences in the signals

appears to be due to the failing energy source. The RMS error of the light values is proportionally the highest error which occurs continuously across the sensing task. These errors are due to the capped values suggesting that not enough information is being transmitted during the night times, when the signals do not change. The error signals follow a periodic pattern which suggests that these values can be used to improve the model, allowing better valuation and costing models.

#### 4.1.5 Discussion

In this section an information valuation and costing framework has been presented. This system enables a node to simply value and cost the information before transmission and thus allows a node to assess the value of the information relative to its neighbouring nodes.

This system allows a node to compute both value and cost using very simple equations and is thus suitable for computationally limited and low power nodes.

The models for the valuation and costing presented here can be substituted for new models and this should be left for future work. Model improvements could include ensuring that the RMS errors remain low or taking into account relay nodes and thus

keeping the information costs artificially high to prevent low-value information being disseminated in the network.

As the dataset used in this section is not a BSN dataset, this system should be tested against a BSN dataset to see how it performs with this data.

An in-depth study into the dynamics of the system should be performed to analyse the fine-grained behaviour of the information valuation and costing system on nodes in dense node deployments.

## 4.2 Simple Wireless Network Framework (SWNF)

In this section we design, develop, implement and test a software framework called the Simple Wireless Node Framework (SWNF). The reasons for developing the framework are first shown. The architecture of the framework is then described along with a detailed description of each component. The results of comparison testing against the common TinyOS operating system are then shown. By evaluating the energy consumption, lines of code and compiled code size, we show that the architecture is better suited to a WSN as opposed to TinyOS. SWNF is available for download from [67].

The WSN community has concentrated on using TinyOS [46] as a standard operating system for sensor networks. Although TinyOS is well supported, it is specifically intended for a set of supported nodes, and changing individual hardware components requires a large part of the code to be re-written. To write a specific driver for a new hardware interface or sensor, a minimum of six files needs to be created and coded to the TinyOS specification: Modules and Configuration files for the Hardware Independent Layer (HIL), the Hardware Abstraction Layer (HAL) and the Hardware Presentation Layer (HPL). This limits users and application developers of TinyOS to a specified set of hardware, and seriously increases the effort of implementing new hardware.

TinyOS also tries to abstract away the hardware levels, defining everything in pure software. Although this allows code to be extremely generic and independent of hardware, it increases the amount of programming and development required for new hardware a great deal, while still not fully achieving complete control of hardware from software.

To illustrate that the abstraction provided by TinyOS does not achieve complete hardware agnosticism, one just needs to investigate the test applications provided by the TinyOS build (The last version assessed was version 2.1.1). In the `apps/tests/TestLpl` directory, the configuration file `TestLplAppC.nc` shown in Listing 4.1 has a macro definition selecting different HAL drivers depending on the platform selected. This shows that a simple application which changes energy levels requires high level adaptive code, suggesting that the hardware paradigm of TinyOS does not suit a sensor node well as hardware agnosticism can not be achieved in the HIL, HPL and HAL, but only in the highest level Application Code.

TinyOS also requires the implementation of all code in a C dialect called NesC. Although NesC is very similar to C, it does require extra learning and understanding of the terms and concepts associated with it. As most microcontrollers have C compilers, using a different programming language from the manufacturer supported programming language does cause implementation difficulties. This has resulted in several popular hardware platforms not officially being supported as yet (including the patent free 8051 Core and the ARM Cortex-M0 to M4 cores).



---

```

configuration TestLplAppC {}
implementation {
    components MainC, TestLplC as App, LedsC;
    components ActiveMessageC;
    components new TimerMilliC();
    #if defined(PLATFORMMICA2) || defined(PLATFORMMICA2DOT)
    components CC1000CsmaRadioC as LplRadio;
    #elif defined(PLATFORMMICAZ) || defined(PLATFORMTELOSB) || defined(↵
    PLATFORMSHIMMER) || defined(PLATFORMSHIMMER2) || defined(↵
    PLATFORMINTELMOTE2) || defined(PLATFORMEPIC) || defined(PLATFORMZ1)
    components CC2420ActiveMessageC as LplRadio;
    #elif defined(PLATFORMIRIS) || defined(PLATFORMMULLE) || defined(↵
    PLATFORMUCMINI)
    components ActiveMessageC as LplRadio;
    #elif defined(PLATFORMEYESIFXV1) || defined(PLATFORMEYESIFXV2)
    components LplC as LplRadio;
    #else
    #error "LPL testing not supported on this platform"
    #endif

    App.Boot -> MainC.Boot;

    App.Receive -> ActiveMessageC.Receive[240];
    App.AMSend -> ActiveMessageC.AMSend[240];
    App.SplitControl -> ActiveMessageC;
    App.Leds -> LedsC;
    App.MilliTimer -> TimerMilliC;
    App.LowPowerListening -> LplRadio;
}

```

---

Listing 4.1: Configuration File TestLplAppC.nc shown without header information  
Version 5948

As modern microprocessors and microcontrollers rapidly increase in both speed (Moore's Law [110]), and decrease in power (Gene's Law [111]), it is clear that modern microcontrollers will be developed. These microcontrollers which show diversity in conceptual design, physical hardware and continual release provide limited backward compatibility which will require a large re-write of code for any system not using standardised code. A non-standard C dialect would thus require rewriting code for every new microcontroller supported by TinyOS. This shows that using a non-standard C dialect limits current applications use in future applications as well as increasing the effort required by developers to use the operating system for future hardware.

Applications developed for TinyOS are also not compatible through versions 1 to 2. Many applications described in academic papers have been written using TinyOS version 1 of the operating system only rendering them unusable with current versions of TinyOS. Several academic papers have been written specifically on porting TinyOS applications from version 1 to 2, indicating the level of complexity in porting these applications [112].

Several other operating systems developed for WSNs exist, but many are tightly coupled to the hardware on which they are implemented, while others provide a similar operating system concept, but written in C.

Contiki [113] is a C-based event-driven kernel providing an innovative preemptive multi-threading approach and over the air programming. Power saving services are thought to be dependent on the application, and therefore no explicit power savings abstractions are implemented. Currently Contiki supports MSP430, Atmel AVR, Windows and Linux cores, and several older computer platforms including the ZX Spectrum. As Contiki is designed to be a full-scale operating system, it is overly complex and the code base is relatively large.

LiteOS [114] gives the application developer a Unix shell-like environment for accessing and operating WSNs, but has limited hardware support, and is currently only accessible for MicaZ and IRIS nodes. It is not an event driven system, and is tightly coupled with the AVR toolchain, making it complex to move to different toolchains.

By using a specified application framework such as The Unified Framework shown in Figure 4.21 and described in Section 4.2.1, the requirement of the system supporting multiple threads is removed, which simplifies the requirements of the system as a whole. By focusing on only cooperative networks (that is systems which share a common communication protocol), the system can run at a low-level in a single thread - which allows for simpler codebase and less chance of concurrent errors, allowing for a more robust system.

By changing the paradigm of an operating system to an application framework, we limit the scope of the application to be developed, but encapsulate the usage model of WSNs to fit more robustly to the framework. The next section presents the framework used for WSNs.

#### 4.2.1 Simple Wireless Network Framework

The Unified Framework [66] is a framework for developing WSNs. An overview of this framework can be seen in Figure 4.21. Drawing from this framework, the concept of three system stacks is used. These are the: Sensor, Communication and Energy stacks which are independent stacks required in a sensor network node. These three stacks encapsulate the majority of tasks required to fulfill the role of a wireless sensor node.

Taking the Unified Framework, a specified set manner in which these individual stacks operate relating to each other is proposed. This is defined as a *framework* as opposed to an Operating System. This framework pattern of operation is similar to the commonly used Model-View-Controller (MVC) framework popular with web applications and modern GUI toolkits.

The SWNF is based on the Unified Framework with the extension of an asynchronous message passing system and library functions to support further sensor functionality and debugging information. It is available for download from [67].

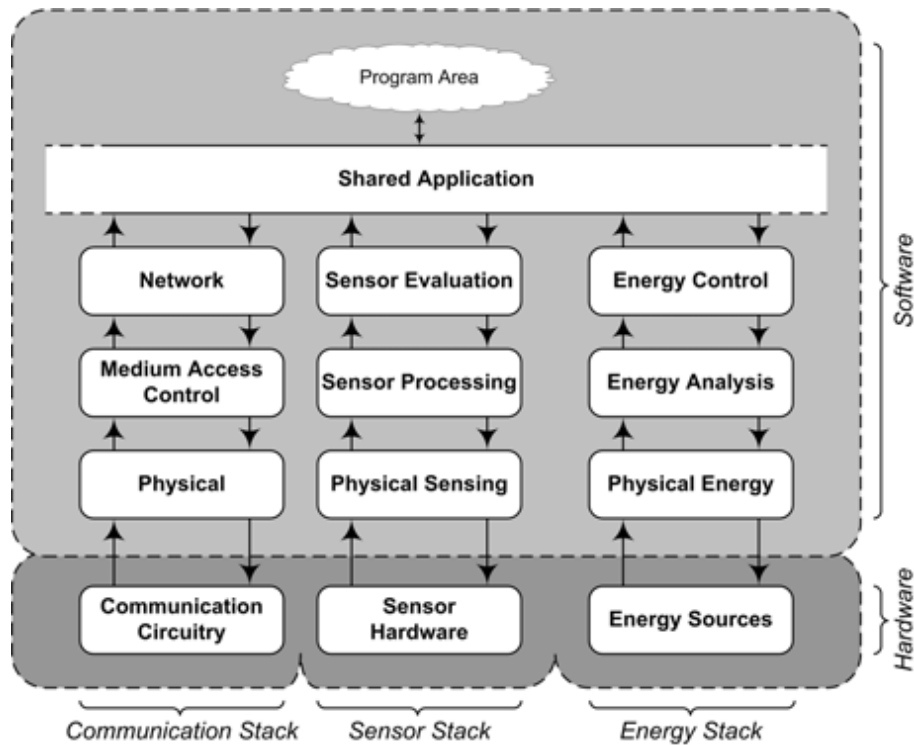


Figure 4.21: Overview of the Unified Framework [66]

Using this framework allows the components to work together independently of the underlying hardware, while still utilising the hardware as much as possible. The different components of SWNF are shown in Figure 4.22.

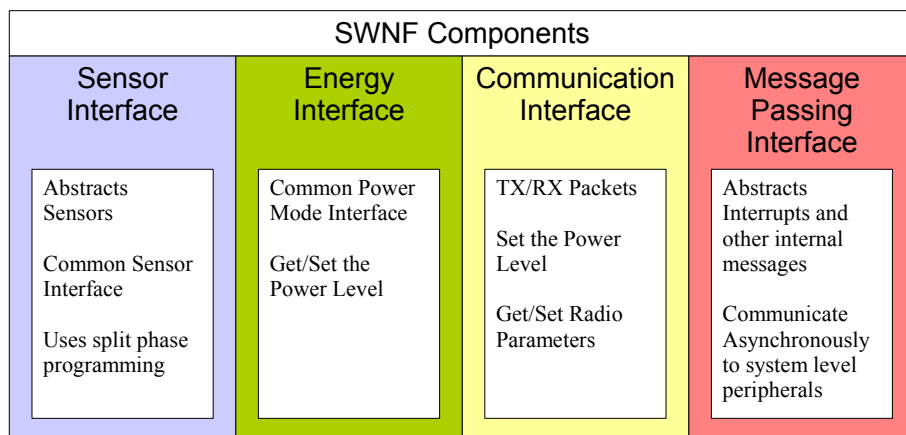


Figure 4.22: The core SWNF Components

A central messaging system core to the SWNF framework is defined which takes into account interrupt driven messages and converts them to a SWNF messages. All system-level messages are thus converted to these defined messages and processed in the service message block shown in Figure 4.23. This central messaging system thus abstracts out the interrupt systems developed in differing hardware implementations, and allows for

abstraction of independent interrupts, while still capturing the functionality of these interrupts.

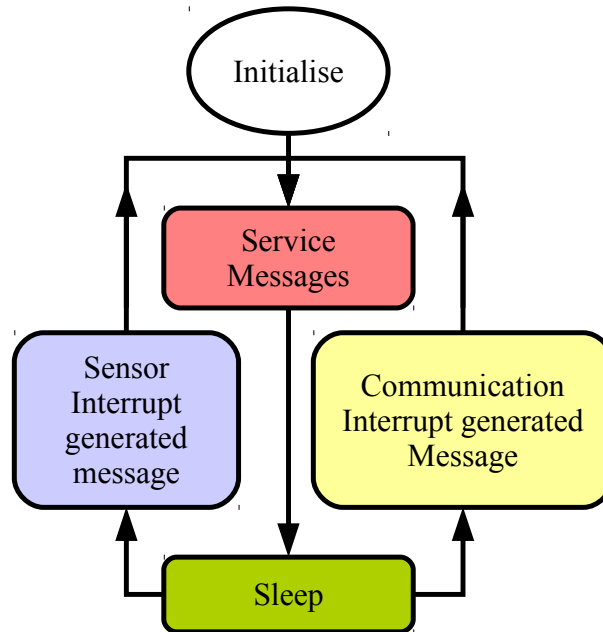


Figure 4.23: Simple Wireless Network Framework Messaging Operation

The system flow described in Figure 4.23 shows how the application flow of an individual node operates. It is a simple single super loop orientated design with limited architecture, attempting to describe all conditions which should arise in a sensor node. By keeping the system structure extremely simple and understandable it allows developers to rapidly implement code which can be used on different hardware implementations. Each hardware interrupt generates a SWNF message which is serviced sequentially by the service messages function, only when the device is ready to service functions. This simple C framework allows for libraries and utilities to be used independently and merged into the code, while simply separating out higher level communication protocols or energy monitoring systems by encapsulating these within separate components. By keeping the codebase footprint small (See Section 4.2.4 and Table 4.8) it allows developers to rapidly gain a full understanding of the whole system, and thus develop applications both robustly and rapidly.

Although timers and timing can be regarded as a sensor components to be developed under the sensing stack, the message passing interface supports simple timed messages allowing tasks to be run after certain time periods. Any advanced timing usage should be further developed as a sensor presenting itself to SWNF as a sensor.

As opposed to the HIL, HAL, HPL device driver layers of TinyOS, SWNF provides just two layers in each stack: a hardware driver layer, and a processing layer. The hardware driver layer is responsible for access to the hardware in a simpler and more understandable way by accessing the hardware directly in the hardware driver layer,

while the processing layer is responsible for higher-level management of that specific resource via function calls which specify the action or activity to be performed. In terms of the communication stack, the processing layer would manage access to the communication layers. By keeping the two layers simple and close to specifications and datasheets, it allows users and developers to take full advantage of hardware, without too much abstraction from system hardware. Although this limits the portability of the code to numerous different WSN platforms, it does provide some portability which more closely resembles the different hardware abilities provided by different manufacturers and devices. Using only two levels of abstraction reduces the amount of code required and the number of files required for each driver interface.

Two abstraction levels opposed to three abstraction levels requires that slightly larger drivers are required for each specific hardware component and hardware drivers. Although this might appear to increase the number of files required, the three-layer approach does not decrease the numbers of drivers required as an HPL driver is still required for each hardware component. The three abstraction level approach used by TinyOS intends to enable the Application developer to not program to the hardware, but rather to the Software stack, but this again has been shown in Section 4.2 to not sufficiently encapsulate the full required hardware abstraction.

#### 4.2.1.1 Communication Stack

The communication stack handles incoming and outgoing packets defined in the configuration header `swnf_cfg.h`. The communication Stack exposes three interfaces to the main application: Transmit, Receive and Initialise. These functions are directly mapped to the hardware interfaces differing only in the parameters passed to the application level representing a complete packet, while parameters passed to the hardware level are raw byte streams. Control of the power levels of the communication stack is not required, as this is controlled by the energy stack. The communication stack deals solely with communication and is therefore clear and concise. The code for the communication stack fixed packet length message handler is shown in Listing 4.2.

---

```
//Commands sent to Radio IC to receive packet p
comms_result rx_packet(swnf_packet_t* rx_pkt) {
    uint8_t length = sizeof(swnf_packet_t);
    hw_rx_pkt((uint8_t*) rx_pkt, &length);
    if (length == sizeof(swnf_packet_t)) {
        return C_COMPLETE;
    } else {
        return C_ERROR;
    }
}
```

---

Listing 4.2: Communication Stack Receive packet callback function for fixed packet length packets showing only simple translation of SWNF packets to bytes

The default communication stack implementation allows for simple sending and receiving of messages. Using this stack approach, more complex implementations of the communication stack can be written allowing the application developer a rapid understanding of what the complex implementation would achieve. Ideas for more complex implementations include security over the wireless communication channel, tailored power control of transmit messages, software acknowledge of messages, and priority queuing of messages.

#### 4.2.1.2 Sensor Stack

To develop code for the sensor stack, a sensor needs to be registered to the device and enabled. Each sensor has an associated memory section detailing what the sensed values should contain similar to the IEEE 1452 TEDS standards. This meta-sensor information allows the sensor node to establish knowledge or understanding about the information relative to the other nodes in the network. This information is not required to be set on the node for non-cooperative networks, but for networks monitoring a single physical process it is highly recommended.

Each sensor driver requires three functions associated with it: `initialise`, `read` and `power_ctrl`. The `initialise` function allows for code to be run at startup to ensure that the sensor is operating correctly. The `read` function is an operation which is run to start the sensor reading process. The `read` function can return either a value immediately, or if required, trigger a result to be delivered to the appropriate code using the Message Passing Interface. The third function `power_ctrl` allows the energy stack to switch the sensor on and off using the energy stack provided by the framework. The `initialise` function for the sensor should register the `power_ctrl` function with the energy stack.

#### 4.2.1.3 Energy Stack

Managing the energy levels of the device is a key component to ensuring the node operates in the lowest possible power state continuously. Without any energy management, wireless sensor nodes can be extremely inefficient and suffer from reduced lifetime.

By utilising a framework approach as opposed to an operating system approach, a more holistic view can be taken on the implementation of an energy management policy. As a default setting which could be overridden, there are four energy states for the framework which describe the functioning of a sensor node: Active, Sensing, Listening, Sleeping. These four modes define the overall state of the sensor node which defines the only functioning states of the sensor node in that mode (ie. sensing implies the radio is off, while listening implies all sensors are off). By using these four states, instead of only focusing on the microcontroller's low power modes, allows a developer to understand what the node is capable of doing in these states, as well as allowing the developer to change and update the states more in-line to the application.

An application developer can define further energy modes by adding a new state in the `swnf_cfg.h` configuration file to define a specialised power saving mode. The power control functions defined in that mode are then needed to be registered with the energy stack to define that modes sleep state.

#### 4.2.1.4 Message Passing Interface

The Message Passing Interface uses a message queue mailbox which occupies 5 bytes of data per message for 16-bit architectures (7 for 32-bit architectures). Each message, defined in a struct as shown in Listing 4.3, contains one byte describing the message type, two bytes of data and a two byte identifier for the callback function to run once the message has been allocated processing time. The callback function is a pointer to a normal C function with no parameters passed to it. Any data which needs to be accessed in these functions should be accessed via global variables. The Message Passing Interface abstracts a nodes interrupt system and allows for asynchronous processing of events. Messages can be passed bi-directionally to different stack components. SWNF comes with a set of generic message types and allows for a set of custom message types. Messages can thus represent specific tasks to be run, to pass data between different functions, or to inform components that certain activities (such as an ADC value being ready) have been completed.

---

```
typedef struct {
    uint8_t class:2;
    uint8_t type :6;
    uint16_t data;
    void (*function)();
} swnf_message_t;

// MESSAGE CLASSES
#define DELAYED_MESSAGE 0x00
#define ENERGY_MESSAGE 0x01
#define COMMS_MESSAGE 0x02
#define SENSOR_MESSAGE 0x03

//ENERGY MESSAGES
#define MSG_ENRGY_SETSTATE_ACTIVE { ENERGY_MESSAGE,TYPE_ENRGY_STATE_ACTIVE,0,0 }
#define MSG_ENRGY_SETSTATE_LISTENING { ENERGY_MESSAGE,TYPE_ENRGY_STATE_LISTENING,0,0 }
#define MSG_ENRGY_SETSTATE_SENSING { ENERGY_MESSAGE,TYPE_ENRGY_STATE_SENSING,0,0 }
#define MSG_ENRGY_SETSTATE_DEEPSLEEP { ENERGY_MESSAGE,TYPE_ENRGY_STATE_DEEPSLEEP,0,0 }

//SENSOR MESSAGES
#define MSG_SENSOR_DATA_MESSAGE { SENSOR_MESSAGE,MSG_TYPE_SENSOR_DATA_READY,0,0 }

//COMMS MESSAGES
#define MSG_COMMS_RX { COMMS_MESSAGE,MSG_TYPE_COMMS_RX,0,0 }

//TIMING MESSAGES
#define MSG_DELAYED_FUNCTION { DELAYED_MESSAGE,0,0,0 }
```

---

Listing 4.3: Definition of Message struct and common message types

The Message Passing Interface contains a mailbox, allowing messages to be queued in turn, which is customisable in length, but is configured to 16 slots by default. This requires a total RAM footprint of 80 bytes for the messaging passing interface. When the MCU wakes from an interrupt, the service messages function is run, which runs

through the gathered messages in the mailbox and processes them in turn. Once all the messages are run, the device returns to sleep.

Messages are posted to the mailbox by running the `swnf_post(swnf_message)` function. Posted messages contain two bytes of data, and a pointer to a function callback. This allows both data to be passed and functions to be called. Examples of data would be sensor data, or sending parameters to devices connected over external peripherals, while examples of both are delayed messages which require data and a function callback. An example of posting different messages to the mailbox is given in Listing 4.4 and further example applications can be seen in Appendix A.

---

```

swnf_message_t periodic_msg = MSG_DELAYED_FUNCTION;
swnf_message_t adc_message = MSG_SENSOR_DATA_MESSAGE;

//In ADC Sensor Implementation
void adc_result(){
    ...
    adc_message.data = ADC10MEM;
    swnf_post(adc_message);
}

//Function to run every 1000 system counts
static void polled_msg(){
    ...
    get_adc();
    periodic_msg.data=swnf_get_ticks() + 1000; //Run Polled function in 1000 ↔
    counts
    swnf_post(periodic_msg);
}

//In main.c
int main(){
    ...
    periodic_msg.function=polled_msg;
    periodic_msg.data=1000; //Start in 1000 counts
    swnf_post(periodic_msg);
}

```

---

Listing 4.4: Example usage of messages in ADC callback and running polled function within application code

#### 4.2.1.5 External and Additional Libraries

Library functions are included and defined in the `libs` directory of the source code. A library is a directory within the `libs` directory containing a header file exposing the functions used and a number of `c` files containing the hardware specific library code.

Accompanying the default installation are several libraries including:

- **LEDS** Allowing the user to get and set different light emitting diodes for the node to show states of the leds.



Table 4.7: Framework Implementation Comparison of Template Applications without hardware implementation

	RF2500T	EFM2500	PC
<hr/>			
Source			
Directories	9	9	9
Source Files	15	15	15
Lines of Code*	445	736	474
<hr/>			
Compiled			
Compiler	CCS	Codesourcery	MinGW
RAM Usage (bytes)	16	232	1180
ROM Usage (bytes)	1248	2248	5980

\*The differences in lines of code represent the different architecture's implementations of `stdint.h`

- **UART** Allowing the user to get data out on the serial port normally connected to a sensor node.
- **RTC** Allowing the user to get Real-Time Clock information about the node activity.
- **ADC** Exposing the Analog-to-digital converter of the sensor node.

#### 4.2.2 Hardware Implementations

Due to the simplicity of the framework, it can be easily and rapidly ported to numerous different hardware platforms. To demonstrate this usability, the stack was implemented on three different hardware platforms/toolchains: A Texas Instruments RF2500T MSP430 Sensor Node, a custom Energy Micro Cortex-M3 Sensor Node which uses a EFM32-G210F128 microcontroller along with a CC2500 radio, and a GCC PC version for emulation.

Implementing the stack on different hardware requires implementation of eight different functions specified by a file called `hw_declarations.h`. The required functions are divided into the different components described in Section 4.2. The hardware implementation should also implement code to convert all interrupt messages to `swnf_messages` and post them to the mailbox. The hardware developer is expected to have an understanding of implementing these functions on the specified hardware.

The actual hardware required to implement the framework is a microcontroller, a sensor of some form, a radio and a timing mechanism or internal clock. Table 4.7 shows the compiled code space of the implementations on the different hardware platforms.

The next section details the importance of using a framework for sensor network development as opposed to a straight operating system.

### 4.2.3 Framework vs Operating System

WSNs tend to fulfill specific sensing tasks which are regarded as highly application specific [115]. Using an operating system to abstract away the hardware and improve software reuse does not fulfill the requirements of both the hardware developer and application developer as it requires the hardware developer to write more code to interface with the operating system (a driver for the operating system) and requires the application developer to learn the intricacies of the operating system used. An application framework on the other hand, handles the functioning of the application according to a set design pattern allowing the application developer to understand precisely how the application functions and concentrate on the application, without concern over the communication, energy management and sensing, but can modify those settings as desired.

Changing the paradigm specifically for sensor networks, allows both the application developer closer working to the hardware, while still maintaining a set pattern of coding the application, allowing code reuse.

### 4.2.4 Results

The reason for using a framework or operating system should address four concerns: simplifying the application developer's access to the hardware; ensuring code is readable, understandable, and generic enough to make it reusable for different sensor and communication interfaces; ensuring the developers code style is standardised within the framework constraints to make code light and efficient; and ensuring a programming style to enable robust applications. To test these concepts, we compared three applications written against TinyOS and SWNF, which are functionally the same but programmed against the two different platforms.

Three applications selected to be tested were used from the TinyOS examples directories. As these applications are examples which attempt to show the simplicity, best case and ease of use of TinyOS, these were chosen as a fair comparison to SWNF. These applications were then re-written as SWNF applications, and attempts were made to write them as functionally similar as possible. The example applications written for SWNF can be found in Appendix A.

1. **Null** This is a basic template application testing to see whether the operating system is installed and working.

2. **Low Power Listen** This application uses the node's low power listening state to attempt to ensure an energy efficient mode.
3. **Oscilloscope** A simple wireless oscilloscope application using a sensor to transmit information to a base station. The application was compiled in TinyOS with the MTS300 sensorboard for the Mica nodes as no sensor is available for them, while an ADC monitoring the internal system voltage was used for the SWNF applications.

The first test attempts to compare the code complexity of the applications. To do this, the number of files used to code the application, the total number of lines of code within those files used, and the number of directories containing those files were compared.

The second test compares the compiled size of the programs on the different platforms, on different hardware platforms. This attempts to investigate the code efficiencies of the systems evaluated.

The third test compares the energy usage of the written applications on the different hardware platforms. As energy usage is an important concept in WSNs, it is important to compare the amount of energy used by each application. As the different hardware platforms require different amounts of power for the devices, it would not be a good metric to evaluate the actual power used, but rather the cyclical changes of the energy levels showing how the framework and operating systems perform.

To test the code complexity for TinyOS, we investigated the number of files required in the build of the different applications. This test was performed as it demonstrates the numerous levels of abstraction performed by TinyOS. The directory count of files used also is an indicator of how many different directories an application developer would be required to navigate, were they to modify a certain part of the codebase, change an implementation of some hardware, or redefine a specific communication stack.

The total code length of all the files is then also assessed to ascertain how large the code base is required for the developer to read through for a clear understanding of the application. Lines of Code was generated through the use of the open source utility called cloc [116]. Cloc generates a report on the number of lines of code while removing comment lines. The number of files used is a metric of the source files used to generate the final binary image, and was generated through `gcc -MM` for the SWNF C source and header files, and `make <platform> verbose` command for TinyOS. As Code Composer was used for the RF2500T, which does not support the MM command, the files were manually counted for the RF2500T.

Figure 4.24 compares the lines of code required for the different hardware platforms. It is clear from the figure, that the lines of code for the EFM2500T is much larger in comparison to the other platforms. This is due to it being a new platform, and thus the hardware implementation for this device is not efficient, both in terms of coding,

Table 4.8: Code Complexity

System	Program	Platform	Number of Files	Lines of Code	Number of Directories
Tiny OS	Null	Mica2	34	1258	12
	Null	MicaZ	28	1039	9
	Null	TelosB	34	1378	8
Tiny OS	LowPowerListen	Mica2	138	5326	16
	LowPowerListen	MicaZ	193	7872	26
	LowPowerListen	TelosB	204	9882	16
Tiny OS	Oscilloscope	Mica2	146	5587	18
	Oscilloscope	MicaZ	201	8132	28
	Oscilloscope	TelosB	204	9895	16
SWNF	Null	RF2500T	21	2419	10
	Null	EFM2500T	44	13855	12
	Null	PC	31	3006	10
SWNF	LowPowerListen	RF2500T	23	2541	11
	LowPowerListen	EFM2500T	46	13957	13
	LowPowerListen	PC	33	3123	12
SWNF	Oscilloscope	RF2500T	26	2596	12
	Oscilloscope	EFM2500T	48	13964	14
	Oscilloscope	PC	33	3105	13

and energy (See Figure 4.28). The differences in the lines of code used differs greatly for the TinyOS platforms, while changes in only small amounts are seen in the SWNF applications. This shows that by having built in extra code to represent the separate communications, energy and sensor stacks into the codebase initially increases the amount of used code, but further code developments to fulfil normal wireless sensor node activities and actions are easily and rapidly achieved with minimal extra code required. This suggests that the framework encapsulates activities and functionality of a sensor node more succinctly. Overall the lines of code for each application is much lower, and thus simpler for both the PC and RF2500T platforms.

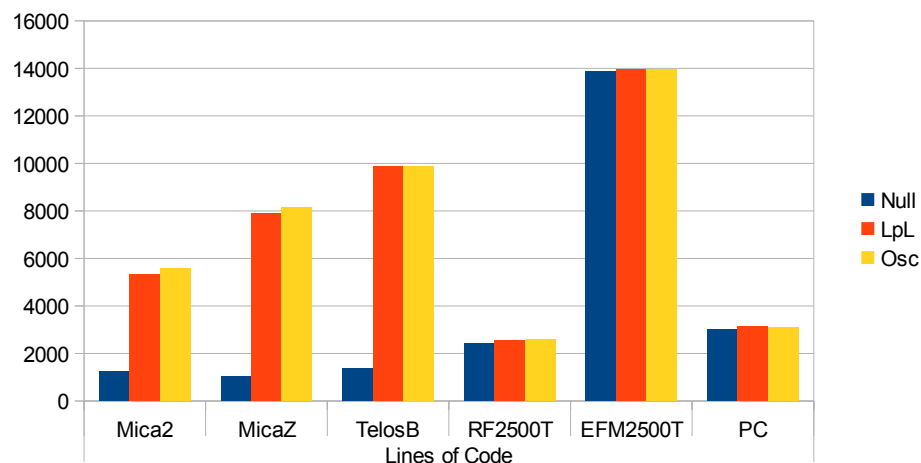


Figure 4.24: Comparison of Lines of Code

A comparison of the number of files is given in Figure 4.25. Due to the three layer driver architecture of TinyOS, the number of files used in each application for TinyOS requires a large number of source files to compile the applications. The increase in files decreases the accessibility of the code base, and although it attempts to simplify the code it increases its complexity by requiring an application developer to investigate over 100 source files in order to understand every aspect of the workings of a simple application.

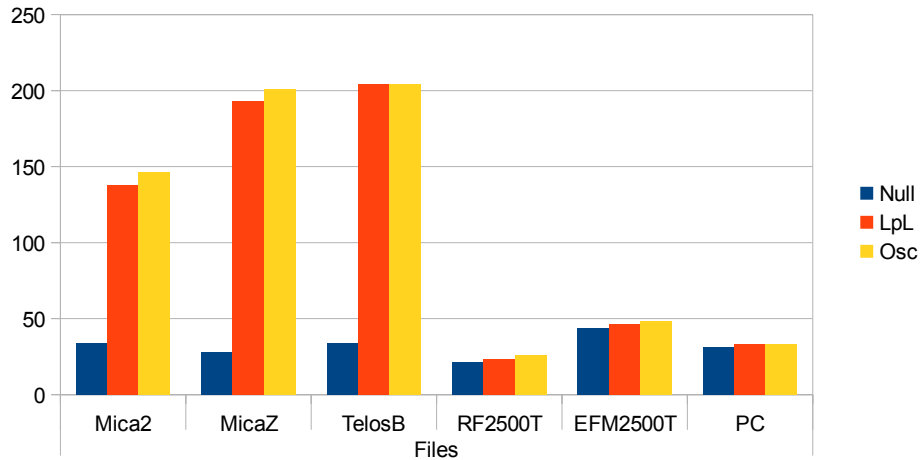


Figure 4.25: Comparison of the Number of Files used in Build

To analyse the complexity further, the number of directories into which the codebase is separated into was investigated. Figure 4.26 shows the number of source directories used in compilation of the final application binaries. Although the SWNF applications tend to once again use more directories initially for template applications, the TinyOS applications use many more directories for larger applications indicating the complexity of the code base again.

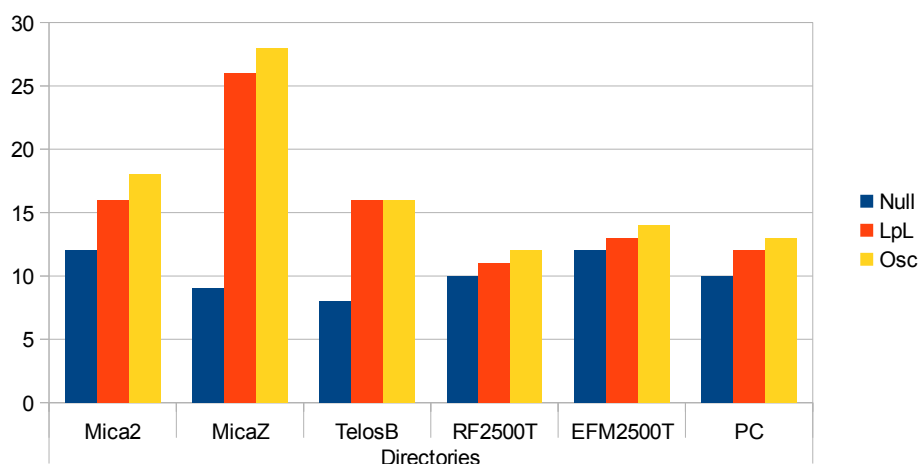


Figure 4.26: Number of Directories

Complexity of TinyOS and SWNF is shown to vary greatly. TinyOS, due to its nature as an embedded operating system, is thus useful for many differing embedded devices,

and does not fit the model of a Wireless Sensing Node operating system well in terms of complexity. By limiting the scope of the software to a wireless sensing node framework, allows for less complexity of code in terms of lines of code and number of files required, as long as the hardware implementation is done well. By not having a lean hardware implementation, the total codebase can be bloated, but the simplicity of the application is still provided for.

The applications were compiled using different compilation toolchains depending on the hardware used. All the TinyOS applications were compiled using the default `make <platform>` commands within the demonstration applications directory. The RF2500T compilation toolchain was Code Composer Studio v4.2.4. The EFM2500T used the Codesourcery GCC port [117]. The PC toolchain was MINGW [118] running on Windows.

Table 4.9: Compiled Application Results

	Null		LPL		Oscilloscope	
	RAM	ROM	RAM	ROM	RAM	ROM
Mica2	5	758	240	9750	273	10552
MicaZ	4	616	306	11300	351	12816
TelosB	6	1372	331	10546	457	16684
RF2500T	158	2986	188	3456	170	3502
EFM2500T	208	17168	296	18064	296	18192
PC	1596	7504	1684	8008	1724	7872

Figure 4.27 shows that the size of TinyOS applications grow close to linear as extra software modules are included, while the SWNF applications stay about the same size. This is due to the framework defining the workings of the network and energy management from the onset, while the operating system concept only includes the required components. The figure also shows that in terms of code size, TinyOS does not provide any particular advantage in terms of compiled code size for matching WSN applications. The figure also shows that the PC application requires a larger footprint in comparison with the embedded processors, but this is due to the added complexity of the artificial network layer.

To measure the current, the nodes were connected to a high-side current monitor implemented using a INA126 instrumentation amplifier connected across a  $1\Omega$  resistor. The input voltage for all nodes was 3V. An experiment was run to evaluate the average current draw for each application. A large  $2200\mu\text{F}$  capacitor was connected in parallel to the load node to reduce the instantaneous current draw spikes. The overall current draw for each application can be seen in Figure 4.28. Current draws for both the PC and TelosB nodes have been excluded, as hardware was not available for the TelosB mote, and the PC implementation was assumed to not be physical hardware.

By looking at the dynamic changes of the current draw in the different applications it shows that the hardware implementations of the frameworks is highly dependent on

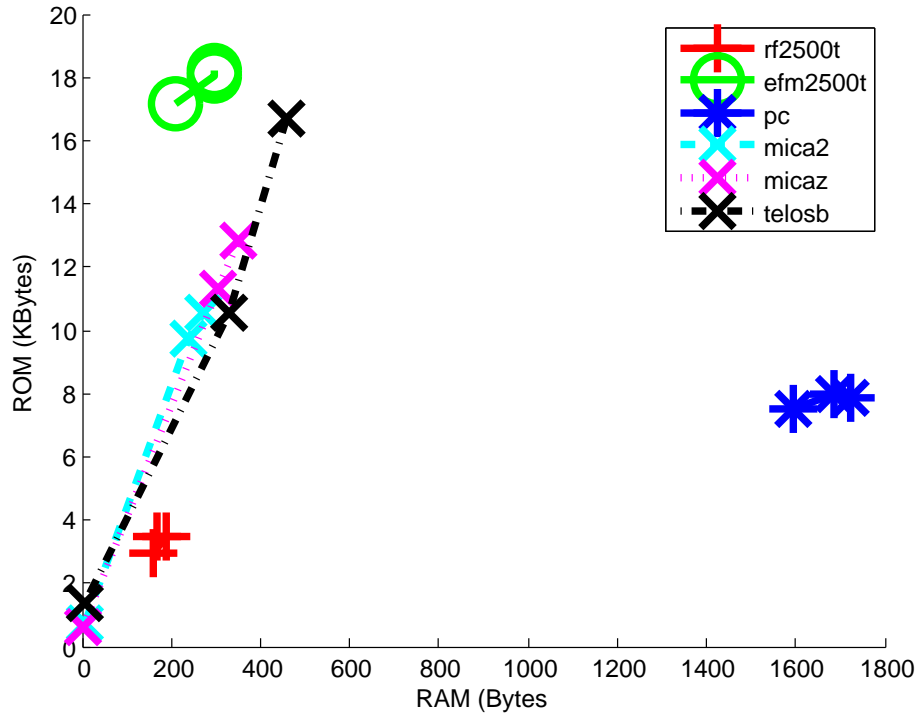


Figure 4.27: Compiled Application Sizes

the current draw. In the Low Power Listening application, the one TinyOS MicaZ implementation does not implement any form of dynamic power control, while the other TinyOS Mica2 implementation does. Similarly, in the Oscilloscope application, the EFM2500T implementation causes aggressive current changes while the RF2500T does not.

In comparing the energy usage of these two frameworks, it is clear that each of the two demonstrate superior performance in some cases. Although specific hardware differences are a major contributing cause of this, further improvements in code can achieve decreased current usage. In comparing the hardware driver implementations of each of the nodes, TinyOS has a clear advantage in terms of development time, while the SWNF drivers are relatively new - suggesting that there is relatively more chance of performance gains for the SWNF implementations.

As shown by the Application code Listing in 4.1, TinyOS requires hardware specific energy management code in the application level codebase to achieve effective power management, whereas the SWNF framework allows a generic interface to achieve the power control. It is thus easier for the application developer to write generic applications to the SWNF framework, although well-written TinyOS applications with a clear understanding of the TinyOS framework can achieve similar energy usage results.

#### 4.2.5 Conclusions

In this section we have presented a simple framework for rapidly developing sensor network applications called the Simple Wireless Network Framework. It is shown that the framework competes well with TinyOS in terms of hardware resource usage and power efficiency, but offers a reduced complexity and a simpler approach to programming for wireless sensor networks. The framework thus fulfills the four concerns related to frameworks and operating systems as shown in section 4.2.4, and provides an architecture which allows heterogenous hardware to run the same code energy efficiently while still preserving a simple and understandable code base.

TinyOS has shown to be an excellent tool for developing embedded wireless sensor network systems. It does however suffer from several implementation details, and by abstracting itself too far into software, creates new problems of portability to new hardware platforms, backwards compatibility of revisions and marginalisation due to different dialect usage. Applications developed for TinyOS thus do not offer guarantees of maximum energy efficiency and implementing the energy efficient code is not trivial. This makes code written for TinyOS good for research and testing, but requires a large amount of extra work for any long-term, energy-efficient or robust deployments. The SWNF Framework presented in this work has shown that using a different approach to coding wireless sensor network devices, an application developer can program an application for wireless sensor nodes without the overheads of using a different C dialect, understanding a three level driver model and learning a larger code base. The SWNF application uses less RAM for similar applications requiring sensing and communication, while providing similar energy usage to a TinyOS node using maximum energy efficiency.



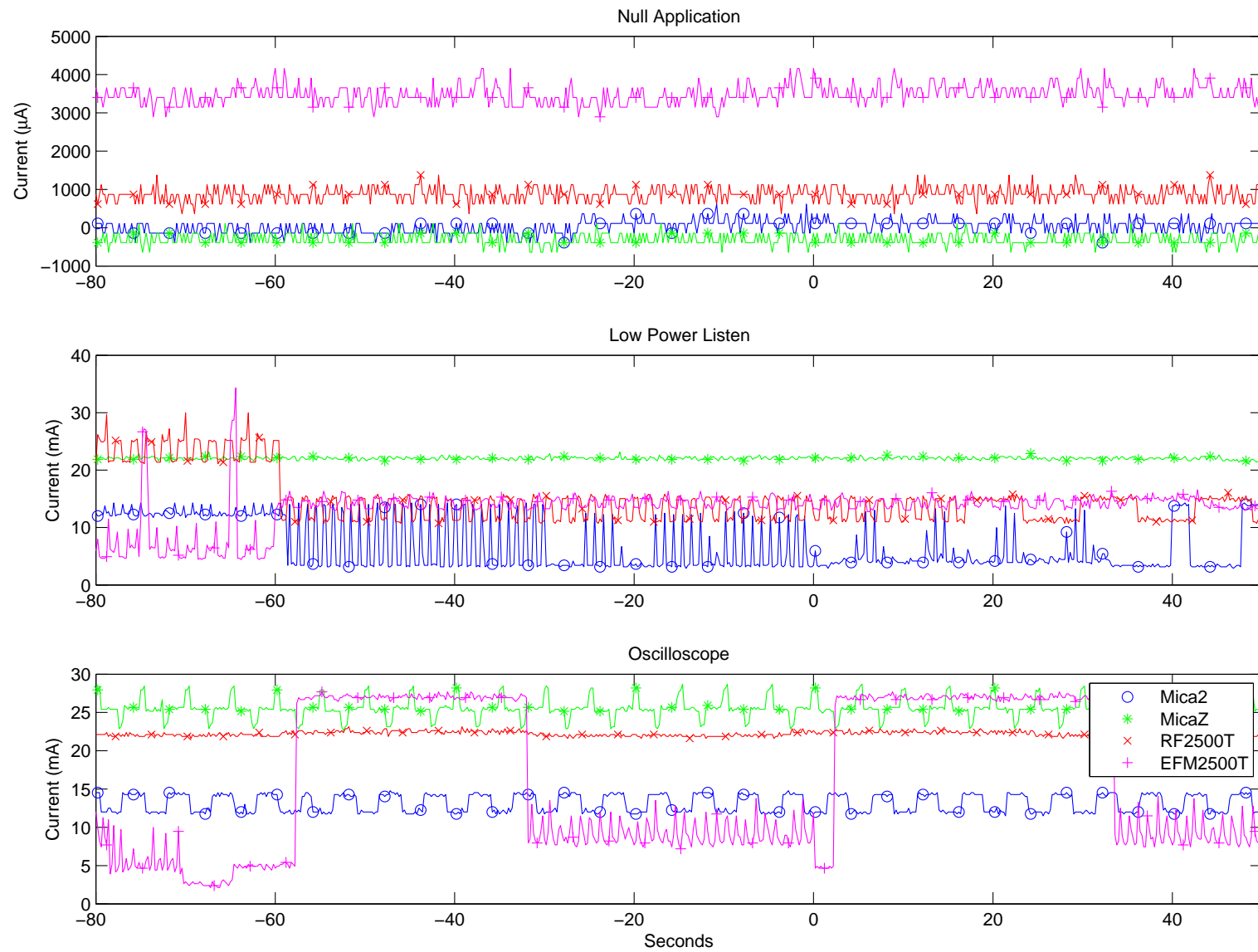


Figure 4.28: Energy usage comparison for Different Applications showing the hardware implementation differences

### 4.3 Tiered Vertical Distribution Framework

In this section an architecture to enable distributed processing among nodes in a vertical network is designed. BSN information can be diverse in terms of quantity as can be seen by the difference in signal outputs shown in Chapter 3. Different signal quantities inherently have different processing requirements. To enable the different processing requirements, an architecture and data protocol are developed through which different amounts of data processing can be achieved on different nodes of different complexity.

In the conversion of measured physical parameters to electronic information a compromise must often be reached between processing ability, energy, connectivity and size. Data collected close to the physical parameter would ideally be a small but powerful processing node with low energy usage and always accessible internet connectivity. This is currently unachievable, as more energy is required for more processing, and size increases with more energy required. Ensuring connectivity requires distribution of some sort be it a redundant connection, fall-back node, or different communication medium. As this is currently not feasible, we present an architecture to attempt to manage this compromise in an effective manner, balancing energy, processing, size and connectivity.

To enable this an architecture is proposed whereby encapsulated data is transported through the network following the diagram as shown in Figure 4.29. In this figure, data collected by the sensing node(s) is transported to a service application via a network gateway, and then further transported to a processing cluster where the data can be processed.

Node A initiates the transfer and transmits the data along with information it has used to capture the data. This metadata describes the sensor, accuracy and related information about the capture, thereby allowing further processing to understand contextual information regarding the data capture. The network gateway can append further contextual information to the data or append encapsulated information from other nodes. This encapsulated data is then transported to the Service Application. The Service Application is responsible for collection and processing of the data, but is not required to process the information itself, instead it creates a set of requests of what actions should be performed with the data and sends out the appropriate requests to the processing cluster or other servers with relevant actions. Once the action responses have been collected by the service application, an action is performed by the service application.

At the network gateway, the data is encapsulated into an XML file called an Information Shard. XML is used as it is a common text format making it easy to both read and parse with any computer; easy to embed data in other file formats and compatible with numerous other standards such as RDF. As XML requires a relatively high overhead of data, it is only used upwards from the network gateway, and the network gateway is responsible for translating the data into a more efficient, yet less understandable format

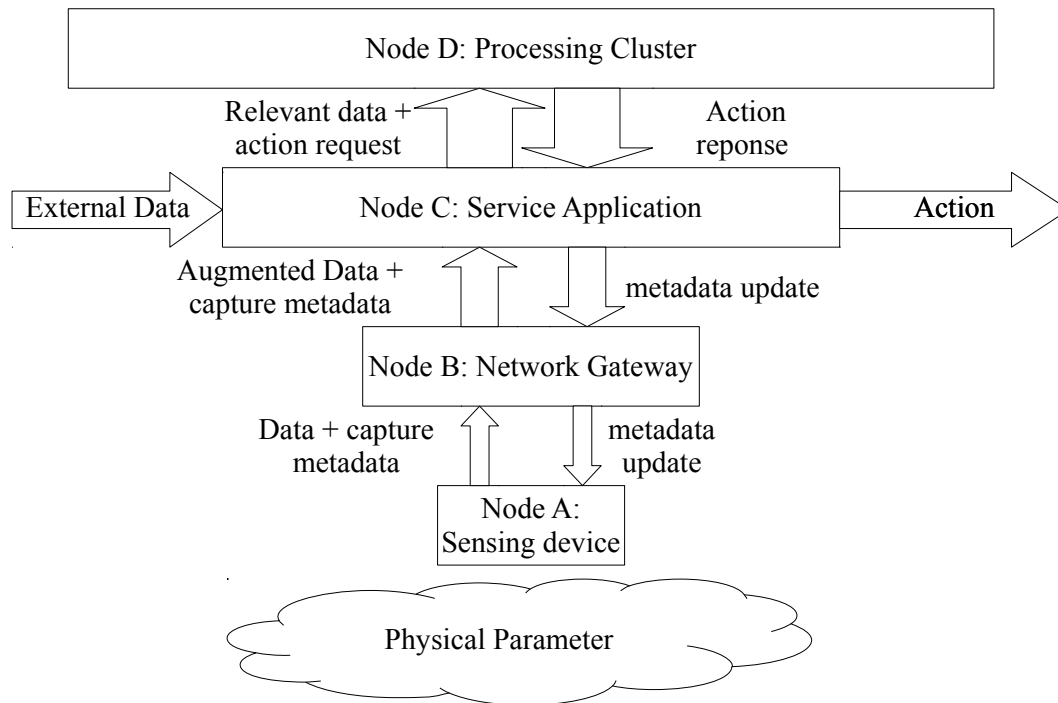


Figure 4.29: Movement of data through proposed system

to the sensing device. The shard consists of two parts, the first contains all the data, metadata and information regarding the captured information, and the second contains information regarding the changes required to the network to improve the data capturing process.

Within the XML shard, the capture information sections contains metadata information regarding the procedure used to capture the data. Although it does not contain the full information used to capture the specific data, it is intended that the nodes through which the data is passed do have a common understanding of how the data was captured.

The update information section contains information generated by higher network layers to update the lower layers of the network with new and updated information on capturing data. This mechanism provides feedback between the higher layers and lower layers of the network.

### 4.3.1 System Overview

The overall architecture of the system can be seen in Figure 4.30. The pattern of message exchange is given in Figure 4.31. Data from the Sensing device is processed and classified on an energy efficient, lightweight processor which assesses the value of the information. Important and valuable information is transmitted to the Network Gateway as classified by a set of rules known as the “capture” rules. The network gateway then further classifies and processes the information, appends further contextual data,

and uploads relevant information to the service application for further data analysis, processing, storage and offline review. The information online is compared to the user's online Internet presence, and further queries can be performed on other processing clusters. Information extracted and evaluated by this process is then compared to a table specifying rules of which to contact the user (Notification rules), and the user is then informed as required by these rules. Updated and processed information is then also transmitted back to the network gateway for communication to the patient, along with rule updates for the rules on the sensing device and network gateway.

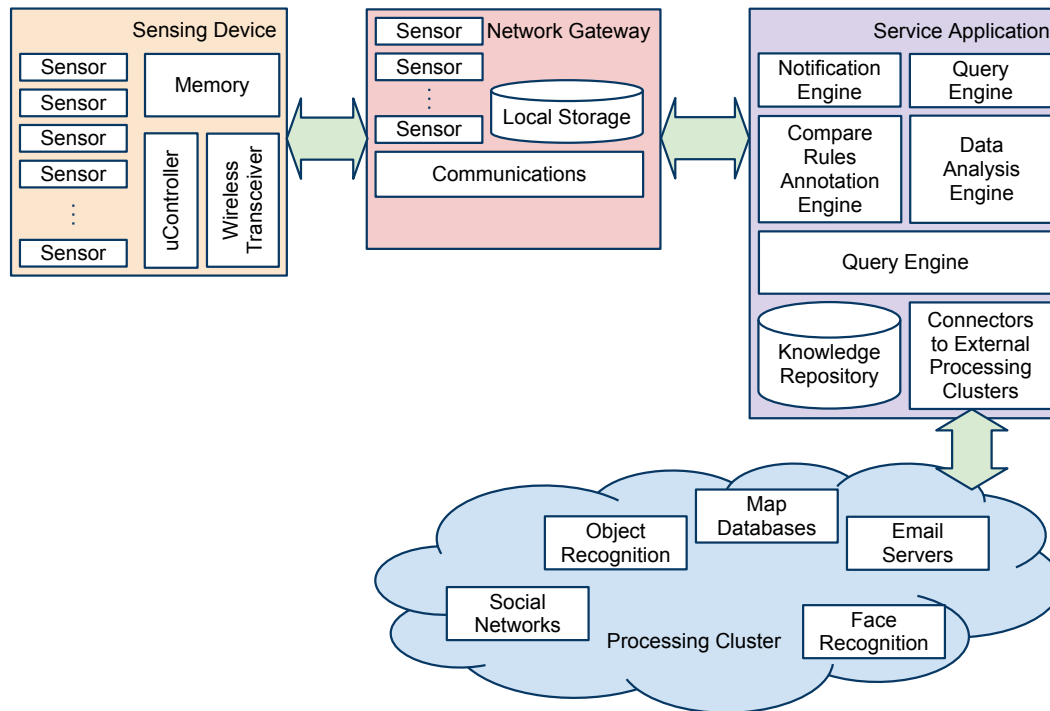


Figure 4.30: Implementation of the Tiered Vertical Distribution Architecture

The data communicated using this system is encapsulated into three different objects describing their function. A data object carrying information from the sensor or network gateway to the service application is known as an “Information Shard”. Information objects fed back from the Service Application to the gateway and the sensing device are known as the “review rule update” and “capture rule update” respectively.

### 4.3.2 Component Description

Each component has a specific role, with clear inputs and outputs to perform its function. The role of the Sensing device is to sense data autonomously and energy efficiently, and to alert the other system components of the important information. The network gateway has two roles: a) to communicate the important information to the Service Application b) to provide an interface with which the patient can interact. The Service application is the front-end interface of data processing centre and provides access to the processing

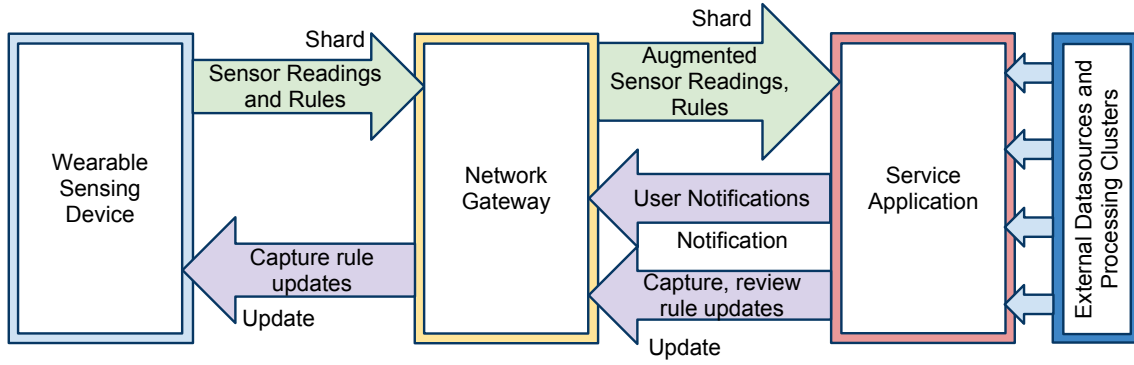


Figure 4.31: Data Message Transfer

clusters. It provides all the tools necessary to analyse data, and allows an interface to the processing clusters where data can be processed intensively and rapidly. The service application also provides the patient with notifications and can take further actions if required. It also processes the information supplied by the other components to evaluate whether rules need updating.

### 4.3.3 Sensing Device

The Sensing Device software is built upon a version of the SWNF shown in 4.2 to clearly separate out the energy management from communications and sensing. The application level software running on node processes the sensor data against an in-memory table of rules to assess the value of the sensor data. When the sensed data is valued to pass an in memory rules, further more energy expensive data capture can be triggered.

Once data capture is triggered, the sensed information, along with the rules which triggered the capture are compiled into a data package and transmitted to the network gateway. The process is illustrated in Figure 4.32.

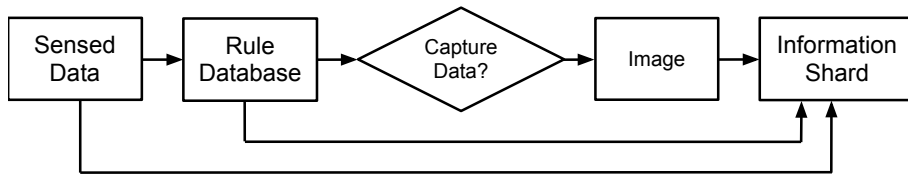


Figure 4.32: Data Capture Process

The sensing device continuously captures sensor data according to the capture rule database until the rules are changed. Capture rules can be adaptively changed through feedback from the network gateway and the service application. The rule database is structured as can be seen in Figure 4.33.

Capture rules can contain variables, and a small set of pre-defined functions such as sum, magnitude, and time-delayed values. The time-delayed values can be viewed as *taps* in

ID	Sensor	Rule	Result if True	Result if False	Lifetime
----	--------	------	----------------	-----------------	----------

Figure 4.33: Rule Structure

a Finite Impulse Response (FIR) filter to create reasonably complex filters in the rule database. The pre-defined functions are simple, often used functions which allow rules to be created using simple structures. A fixed number of variables, due to the limitation of memory size, can be used within the rules database to allow more complex sets of rules based on a set of individual rules.

The rule lifetime allows rules to last for either a specified short period of time, or a longer more permanent time period. This allows the Service Application to enable the Sensing Device to tune its sensitivity to capture more/less data when it regards certain time periods to be more relevant for data capture as opposed to other periods of time which it regards the patients activity to be less important for capturing data.

#### 4.3.4 Network Gateway

The Network gateway comprises of two main components: an interface component allowing the user to simply and easily interact with the pervasive system; and a communication component allowing the information gathered by the sensing devices to be further processed, classified and forwarded onto the service application.

The interface component provides the patient with a system to value previously captured information, and change current priorities or rules using a simple interface. It also acts as an interface to the service application allowing the user to link other data and add or connect other processing clusters to the pervasive system.

The communication component provides communication from the Sensing Device to the Service Application. It appends further sensed information to the sensed data from the sensing devices and is responsible for ensuring capture rules are returned to the correct sensing devices.

On receiving information from the Sensing Device, the gateway analyses the incoming data shard according to a set of rules called the *refine* rules database. Refine rules are written in a similar format to the *capture* rules, but comprise more complex functions and rule types allowing for basic signal processing functions such feature detection. Once a shard has been processed using the network gateway's *refine* rules, the shard is then pushed on to the Service Application for further processing by the service application and processing clusters.



Figure 4.34: Information Shard

Figure 4.34 details the different information components of an Information Shard. The information contained in the shard is intended to describe a self-contained data package of the sensors data. From leaving the gateway, an Information Shard should contain high resolution information; why the information is important (Detected Features); and the rules it used to make the decision (Rules used). Although the information sent in this shard might appear to be redundant, it is required to ensure that the further stages in the pervasive system can process what rules were used to trigger the data capture, and thus make changes to those rules, should it decide to do so.

#### 4.3.5 Service Application

The Service provides a mechanism for long-term backed up data storage, powerful offline data analysis and annotation, a gateway to external processing clusters and online services, a service to update the rules databases on both the Network Gateway and Sensing Device, and a Notification Engine which describes which further actions should be taken. The service application waits for new Information Shards to be received, and on reception, processes the data on a rule set for the patient, known as the compare rules. The compare rules, which supersede the capture and refine rules, describe the required comparison of the current Information Shard to internal and external data sets, as well as running intensive processing via the processing clusters on the data to extract information which would prove useful to the patient in the future. Compare rules can contain any subset of capture or refine rules.

#### 4.3.6 Conclusions

In this section we have presented an architecture for use in BSNs. This architecture is a vertical architecture for distributing processing among a group of heterogenous devices.

This architecture defines the standards used to transport data around the network and the activities required to be performed by each tier in the network. Using this framework

allows relatively computationally weak nodes to move their data to computationally powerful servers and processing clusters to analyse the data in a structured and standardised manner.



## 4.4 Discussion

In this section we have shown three software frameworks for enabling BSNs. The first framework is an information dissemination framework to support energy-efficient data dissemination for different adaptable data types through a BSN. The second framework enables lightweight hardware agnostic coding. The third final framework supports vertical processing on nodes enabling computationally powerful nodes to perform more processing than computationally weak nodes.

This section has demonstrated how software frameworks are required to enable distributed BSNs. Specifically, this section has highlighted and addressed two sections of the complete system, task distribution and data distribution, and has highlighted important factors required in realising these sections:

**Task Distribution** Task Distribution and processing cannot always be performed on a single node, and is required to be distributed among computationally powerful enough nodes, and nodes which do not need to concern themselves about efficiency

**Data Distribution** BSN data is extremely diverse in quality, value and quantifiable information. To ensure efficiency of the network in terms of energy and time, the network requires a means of valuation of information/data to ensure that valuable information is being gathered.

**Hardware Agnosticism** To simplify tasks running completely and seamlessly across different platforms, a unified programming model is required for heterogeneous nodes. This allows different nodes to be programmed against the same codebase, but still operate on different hardware platforms.

The next chapter details two implementations of these concepts, exploring them in terms of horizontal and vertical networks. The first network, a horizontal network, uses the capacitive sensor presented in 3.1, and the frameworks presented in Section 4.1, 4.2 while the second network uses the Camera Interface presented in 3.2 and the architecture presented in 4.3.

## Chapter 5

# Prototype Horizontal and Vertical Body Sensor Network Platforms

In this chapter two distributed network platform prototypes are designed and developed. These two platforms provide proof of concept of the work done in previous chapters and show the application of the frameworks developed in Chapter 4.

The first platform is called miNet. miNet is a horizontally distributed BSN platform for healthcare which collects data on several physiological signals of a patient for long term use. This platform makes use of the novel respiratory sensor developed in Section 3.1, the information valuation and costing framework developed in Section 4.1 and the node framework shown in Section 4.2.

The second vertically distributed platform is called DejaView and makes use of the camera interface designed in Section 3.2, the node framework in Section 4.2 and the Tiered Vertical Distribution Framework in Section 4.3.

All hardware presented in this section was designed and developed by the author. The miNet platform presented in section 5.1 is a project run and managed by the author, while the Dejaviw system presented in Section 5.2 is a project within ECS [13], but all the software and hardware presented in this section is work of the author alone.

### 5.1 Distributed Personal Body Sensor Network (miNet)

In this section we develop miNet, a distributed network comprising of a set of individual sensor boards connected to a set of different sensor boards. Because each node requires a different amount of pre-processing, the sensor boards can be connected to either a Texas Instruments RF2500T mote or a EFM2500T (developed in this section), thus showing how it is a heterogenous platform. The nodes evaluate their information before

transmission, communicate using a protocol called the Simple Helping Hand Protocol (SHHP) and the software developed for each node is written using SWNF (Section 4.2).

SHHP is a protocol developed for horizontal BSN platforms which provides a standardised, low power, multicast orientated communication protocol among heterogenous nodes. It has been developed by the author for the miNet platform.

The following section describes the platform which enables distribution.

### **5.1.1 System Overview**

miNet is a heterogenous, horizontally distributed physiological Body Sensor Network. This network allows for processing and distribution of sensing through a BSN. This platform demonstrates the ideas presented in the previous sections and shows how the system operates as a whole.

The system comprises of a set of processing nodes connected to different sensing boards and a set of sink nodes or data gathering nodes presented in this section. The system is heterogenous in that two different processing boards are used in the network: The Texas Instruments RF2500T and the EFM2500T. Sensing is performed by numerous nodes around the network, and data is gathered or transmitted further by other nodes.

miNet attempts to enable a long term distributed sensor network for detection and prevention of physiological changing parameters of an individual, thus enabling long term monitoring models such as the Modified Early Warning Score [119] to the patients body.

The next section describes the hardware devices developed for the miNet hardware platform.

### **5.1.2 Hardware**

Several nodes were developed for miNet to enable the heterogenous distributed horizontal network. The circuit diagrams for the nodes and processing boards are shown in Appendix B and the PCB Layouts are shown in Appendix C.

The first node, the EFM2500T detailed in section 5.1.2.2, is an enhanced processing node with much improved processing capabilities. The BTNode shown in section 5.1.2.3 board links together with a normal RF2500T or EFM2500T board to enable the miNet network to interface with a Bluetooth network.

A set of physiological sensor boards were developed to test several of the concepts presented in this work. These boards are intended to measure physiological body parameters using as little power as possible.

The following sections detail the sensor boards developed for this work.

#### 5.1.2.1 Sensing Nodes

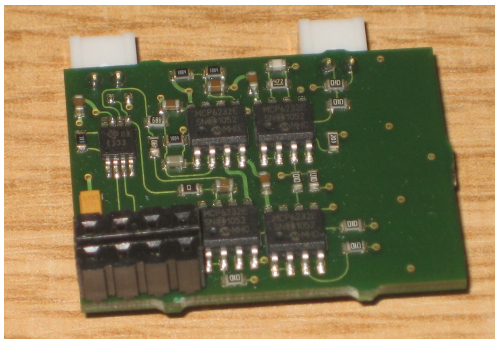
The cardionode sensor board measures a patients ECG signal. Its circuit diagram is given in Appendix B.4. This circuit diagram was designed from the four lead low power ECG Circuit provided in the INA333 Datasheet [120] and used due to its low power. Figure 5.1a shows an image of the Cardionode sensor board.

The respinode sensor board measures a patients respiratory signal using a guarded capacitive sensor system as shown in section 3.1. The respinode circuit diagram is given in Appendix B.5. The Respinode allows for four sensor attachments to ensure respiratory rate signal can be observed from numerous noisy signals. An image of the device is shown in Figure 5.1b.

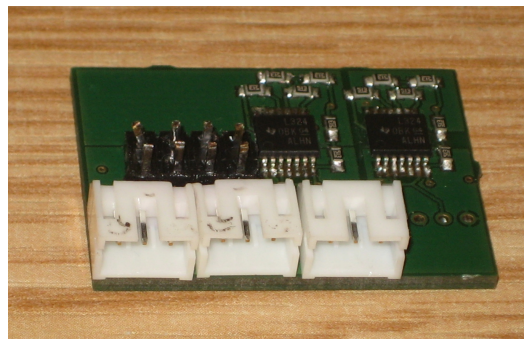
The spoxnode sensor board measures a patients pulse oxygen level and heart rate via the Saturation of Peripheral Oxygen (SPO2) method [121]. Two versions of the spoxnode were developed. Version 1 uses a BCI 31392B1 Digital Micro Power Oximeter board for the SPO2 calculation. Version 2 uses the circuit shown in Appendix B.6 and is visualised in Figure 5.1c.

The accelnode, which directly measures acceleration via an accelerometer, can be used for numerous different tasks but predominantly activity recognition. The accelnode, shown in Figure 5.1d also contains a USB connection with power regulator and an FRAM memory storage for saving data in low power. The USB connection provides power only, and allows several nodes to be chained on a single standardised cable for long term use. Power can also be provided by an onboard battery, with the device being able to switch between power sources seamlessly. The FRAM storage provides fast and low power data storage for high-sampling rate data. This node also stores information gathered by the network onboard and allows for previous information stored in the network to be retrieved at a later stage of the sensing process. The circuit diagram for this node is shown in Appendix B.7.

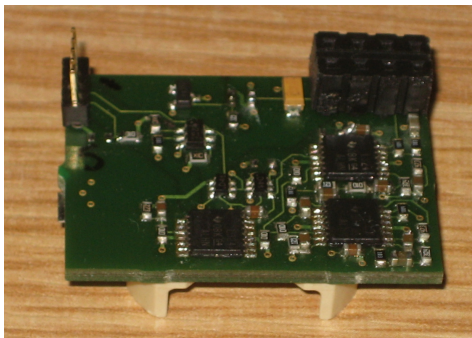
Two other sensors were connected directly to the EFM2500T boards. These included a skin temperature sensor and a Commercial off the Shelf (COTS) Heart Rate Monitor. For an accurate measurement of skin temperature, a DS18B20  $0.5^{\circ}\text{C}$  accurate temperature sensor was used attached to the skin surface, while a Polar Heart Rate Module (HRNode) and band to detect a patients heart rate was connected to a processing board to measure heart rate.



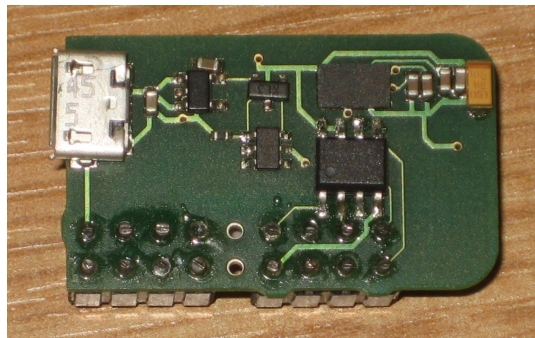
(a) Cardionode Sensor Board



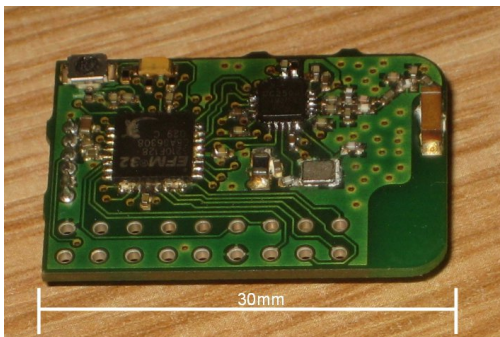
(b) Respinode Sensor Board shown with fourth connector removed



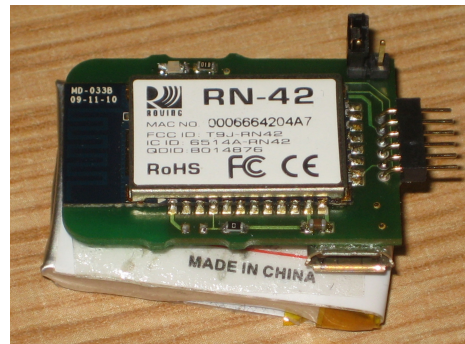
(c) Spoxnode2 Sensor Board



(d) Accelnode Sensor Board



(e) EFM2500T Processor Board



(f) BTNode Communication Board

Figure 5.1: Sensor boards developed for the miNet system

### 5.1.2.2 EFM2500T Processing Board

The EFM2500T is a processing board similar to the RF2500T, except that instead of using an MSP430F2274 microcontroller on board, it uses a 32-bit Energy Micro EFM32G210F128 microcontroller containing an ARM Cortex-M3 core capable of running up to 32MHz. This allows the device to operate an order of magnitude faster while still maintaining the same form factor, size, similar energy consumption and communication platform. The EFM2500T has a slightly different pinout configuration, as it has a separate USART attached to the last few pins of the output connector allowing it to communicate with both the radio and external device simultaneously.

Figure B.8 shows a circuit diagram of the EFM2500T.

### 5.1.2.3 BTNode Communication Board

To enable the nodes to communicate with further different hardware devices, a Bluetooth enabling communication board was developed. This board connects the main USART Serial driver of the EFM2500T and RF2500T boards to a bluetooth communication module. The communication board includes a lithium polymer battery charger and voltage regulator to ensure a stable voltage required for the Bluetooth radio. The device is shown in Figure 5.1f.

### 5.1.3 Software Framework

The software architecture designed for miNet is illustrated in Figure 5.2. In this diagram, SWNF handles the low-level radio, sensing and energy stacks, while the miNet layer handles the common code between devices. The Info/Value Costing described in Section 4.1.2 is implemented and added as a layer which is used before transmission using SHHP (see subsection 5.1.3.1) as the application level protocol to disseminate information among the nodes participating in the network.

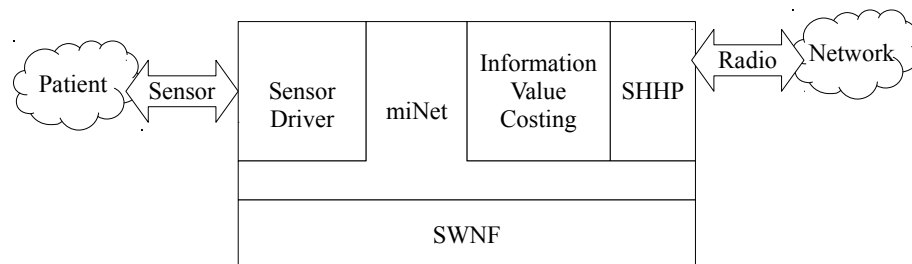


Figure 5.2: Software Overview of miNet Node

In Figure 5.2 sensor drivers are written and accessed via SWNF through an intermediary layer built on top of SWNF called miNet. SHHP packets are encapsulated within default SWNF packets to allow the protocol to be implemented on different hardware platforms. SHHP allows the heterogeneous sensed parameters to be communicated on the network with little network overhead. The miNet layer is a shared library which is used across all nodes, whereas each type of node has a separate configuration and application code file to describe its own sensors and abilities.

The miNet software layer describes how the sensors are to operate and how the information is to be transmitted across the network. The intention of miNet is to monitor changing physiological parameters of a person. The latest and most up to date physiological parameters are captured and stored on any nodes wishing to capture the information. Nodes are required to use SHHP instructions to request information from

the participating nodes regarding certain topics being monitored. miNet allows for the different nodes to provide different data depending on what is required. Table 5.1 shows the different topics and nodes which provide the information about the topics.

Table 5.1: Topics and Nodes providing the data

	ECG	Temperature	Motion	Respiratory Rate	Heart Rate
Cardionode	✓				✓
HRnode					✓
SpoxNode				✓	✓
AccelNode			✓		
RespiNode				✓	
DS18B20		✓			

SHHP supports a common BSN level data paradigm, each individual node requires a sensor driver to be programmed into that specific node to provide the correct information which each node will make available to the network. Along with the driver, a model is associated with the sensed parameter to ensure the data is valid and correct. This information is then passed on to the information value costing system, and if classified as valuable, passed on to the network. This process is illustrated in Figure 5.3. Each node thus requires individual code to be programmed onto the nodes; a process made rapidly possible using the SWNF framework.

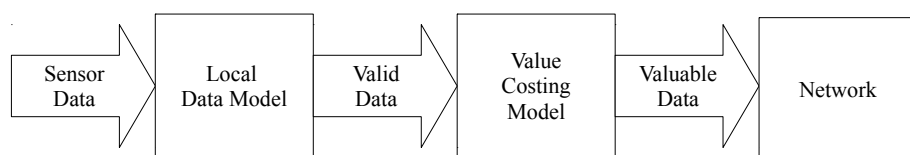


Figure 5.3: Flow of data captured from miNet sensors

Each miNet node has the same codebase except for three separate individualised sections detailed in Figure 5.4. These sections are independent to ensure that code and memory for each device is minimised by not having to store drivers not used. Sensor model code is used to describe and program the individual sensors of each device. The Information Value Model parameters define the valuation/costing model parameters for the specific sensors used. The packet data parameters are also parameters to address certain issues with the specific sensor data such as the conversion of data values to data streams.

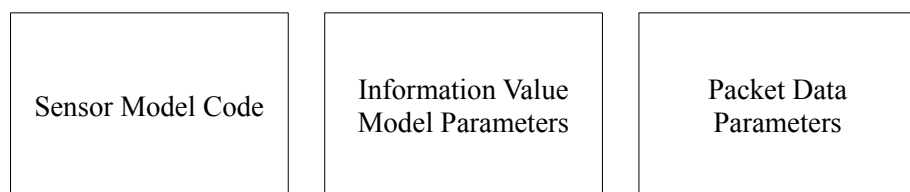


Figure 5.4: Overview of Data to Information transfer in miNet

The miNet layer also provides node-level features which are able to be leveraged on all nodes by using the SWNF framework. Sensor values ascertained by the sensors on the

different sensing boards are fed into the miNet framework to enable node level processing functionality such as entering low power modes.

#### 5.1.3.1 Simple Helping Hand protocol (SHHP)

Simple Helping Hand protocol (SHHP) is an application level protocol for body sensor networks to allow physiological parameters to be transmitted across a short-range wireless network. It ensures information is transmitted with the minimal amount of energy acceptable to the network. This allows for a smaller battery, energy harvesting and energy constrained devices to be used, which in turn allow the patient more freedom of movement.

SHHP is an enabling protocol which encapsulates the data provided by the Information Valuation and costing system shown in Section 4.1.2 and ensures different data is transmitted correctly among the nodes.

SHHP, inspired by the CodeBlue publish/subscribe routing layer [15] and ADMR [47], differs from other body sensor network communication protocols as it is a multicast orientated protocol. The intention of SHHP is to support a node transmitting to a set of nodes which use the information provided. This paradigm thus enables nodes to communicate in a peer to group fashion.

As SHHP is an application level protocol, it is compatible with many current WSN network level protocols. An overview of the generic packet structure can be seen in Table 5.2. As SHHP has a data size of 49 - 105 bytes per packet, this allows it to integrate into Industry Standard IEEE 802.15.4 Messages, TinyOS AM Messages [122] and Texas Instruments SimplicTI messages [40], which are all protocols used in current WSN deployments.

Table 5.2: Application Level Packet Structure

Version	Priority	TTL	Message Type	Group	Data
4 bits	4 bits	4 bits	4 bits	1 byte	49 - 105 bytes

SHHP is a backward compatible protocol allowing for continuous development. To address concerns with development, it uses 4 bits to encode the version of the protocol, thereby allowing sixteen concurrent revisions of the protocol. Four bits are assigned to prioritise packets destined across multi-hop networks, while the Time-to-Live field (TTL) ensures that messages are not continuously forwarded, and messages can be forwarded up to a maximum of fifteen times. The Message Type field indicates what type of message is being sent. Currently there are two message types: Data Messages and Instruction messages. Data messages are simple messages relaying information about a patient or user of the network, while Instruction messages are messages which are used to redefine and reprogram the on-node scoring table and other modifiable system



parameters. The next field, Group, indicates about which group/patient/user the node is providing information. The final field Data, is a dynamic sized field providing the information relative to the packet such as data points, streamed signals or Instructions with which to reprogram the node. Other fields such as destination, source, sequence number, CRC check and others are all gathered from the lower network level of the OSI stack, and are assumed to be known in this application level protocol, so therefore not required to be re-included at this level.

After two nodes have been paired using a handshake provided by an underlying network level, SHHP stores the address of each node in its memory. A group is then defined for a user's data, allowing different nodes to communicate information about the same user to and from different nodes and devices. This allows for a virtual connection to be established between the different nodes in the network.

Information stored in the Datafield of the generic packet descriptor, can be of two types, data and instruction packets. Data packets are packets which transport data about the node or the user. It takes the form of key-value pairs, with the first pair always a 32-bit time value describing the present time from midnight accurate to 20 micro-seconds. 20  $\mu$  seconds has been chosen as it allows for a sampling frequency of 50kHz, which is far higher than the highest required sampling frequency of intended physiological measurements, while still small enough to allow only a single 32-bit value to be sent on the start message transmission.

Table 5.3 defines how a data value is communicated in a key-value pair. The ID of the key describes precisely what information is defined in the following bits, while the size tells the node that the information is either 8-, 12-, 16-, 32-bits in length or 8-, 12-, 16-, 32-bit stream. The next key-value pair is located after the value, and therefore each packet is required to be read in order. If the information is a stream, no other key-value pair is included in the packet, while different values can be sent if single values are used. If the information is not a stream, and multiple values are given for different sensors, a date value key-pair should be placed in between the data to define the time change between the initial packet time and the new time. 16 Keys are used as this describes all the data provided in this network sufficiently, while keeping the transmitted data overhead low.

Table 5.3: Data Packet Key-Value Pair

Key		Value
ID	Size	Number
4-bit	4-bit	8-,12-,16-,32-bit values 8-,12-,16-,32-bit stream

Instruction packets fulfill the role of network management and node management.

Instruction packets are packets which are used to request data from the user's nodes, define user details related to the user which the node is monitoring, update the node's valuation/costing models, or upgrade the software which is currently on the node. These packets are more complex than normal data packets, and therefore require different descriptors for each instruction type.

As SHHP allows for a set of topics to be described in the network protocol, the following section describes the SHHP data used in this section. Table 5.4 summarises the associated SHHP data messages and related functions. These are described in a header file used by SHHP called `shhp_config.h`.

Table 5.4: SHHP Keys and associated Data Values

Key	Abbreviation	Types of Data	Default Bit Size
Motion	MOT	Stationary, Slight, Active	8 Enum
ECG	ECG	10-bit unsigned integer	16
Plethysmography	PLE	10-bit unsigned integer	16
Temperature	TEM	10-bit unsigned integer	16
Heart Rate	H_R	8-bit unsigned integer	8
Respiratory Rate	R_R	8-bit unsigned integer	8

### 5.1.3.2 Sensor data Models

Each sensor requires a model to process the sensor data before valuation and costing. The output of the sensor model is a detection of whether the information is valid, and if so the data value. The next section gives an overview of the individual sensors and their models.

To convert the acceleration information into a useful descriptor, categorised into stationary, slight motion and active categories, the node ran a 10 tap moving average window over the sum of absolute of the three axes running at 10Hz, and used this information as an indicator of activity. The calculations for this function are illustrated in Figure 5.5.

In Figure 5.5 the red output result is an indicator of motion associated with the patient. Motion has been classified into three states: stationary, slight and active. This is a basic model of activity to indicate the user's current motion. It is intended to be worn on the wrist of the patient.

Using the ECG obtained by the cardionode, two SHHP parameters can be derived: ECG and heart rate. To obtain the heart rate, a simple beat detector was used and implemented as shown in Figure 5.6. In this figure, a beat is detected by a rapid instantaneous change rising to over 750 and falling below 200. These values were selected from single patient trials. Signal lock is detected by a large rapid decreasing change

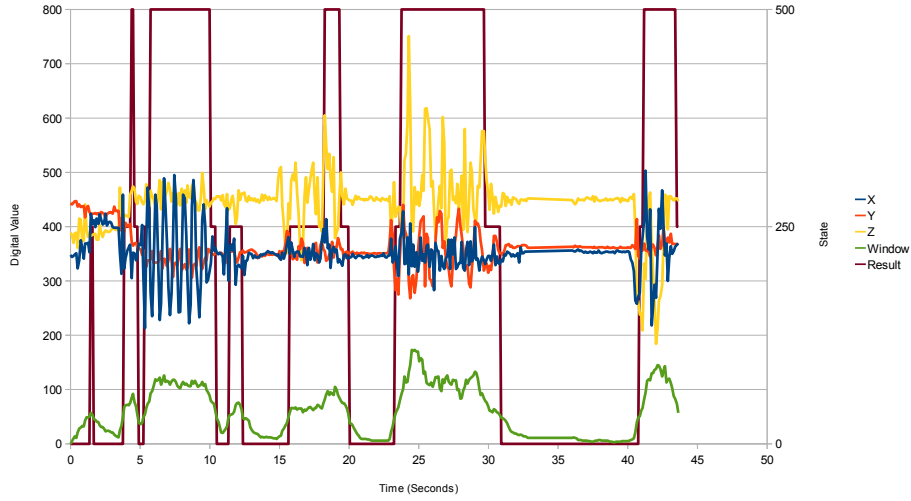


Figure 5.5: Accelnode Activity Classification sensor model

detected over a short sample rate triggering a signal lock. If a rapid decline does not have a peak value of over 750, a signal lock is not detected.

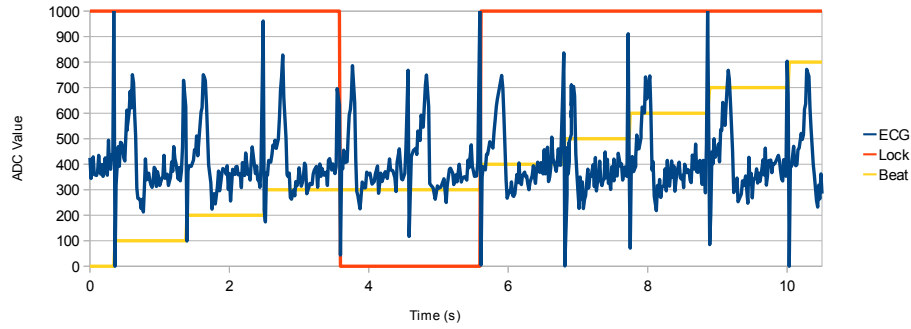


Figure 5.6: ECG Sensor model

#### 5.1.4 Information Valuation and Costing Model Implementation

The Valuation Model consists of a struct containing the valuation coefficients and related values as shown in Listing 5.1.

In the `ivc_value_parameters` struct, there are two distinct sections relating to the temporal and spatial models; a descriptor on the coefficient size and an extension allowing historical information to be included. The historical extension allows for value calculations to be normalised spatially and temporally giving a good relative approximation of the information value between 0 and 255.

The Costing models data is contained in the struct shown in Listing 5.2.

The valuation and costing models are modified by the functions shown in Listing 5.3. The functions return a value representing the value and cost of information and then

---

```

typedef enum {
    BIT_32 = 3, BIT_16 = 2, BIT_8 = 1
} iv_data_size;

typedef struct {
    uint16_t time_taken;
    uint8_t node_id;
    uint32_t data;
} ivc_net_data_val;

typedef struct {
    uint8_t it_number;
    uint8_t it_coeffs[IVC_MAX_VALUE_COEFFS]; /*< Temporal Model*/
    ivc_net_data_val is_vals[IVC_MAX_NET_SIZE]; /*< Spatial Model*/
    iv_data_size data_type; /*< Bit size of coefficients:8,16,32 */
    /* Historic Information */
    float historic_max;
    uint16_t historic_counter;
    uint16_t historic_timeout;
} ivc_value_parameters;

```

---

Listing 5.1: Valuation Model Structs

---

```

typedef struct {
    /*Energy Values
    uint8_t E_elec; /*< Per Byte RX/TX Circuitry Energy*/
    uint8_t txPower;
    uint8_t e_mcu; /*< MCU Energy for sampling and processing */
    uint8_t n_hops; /*< Number of hops */
    uint8_t dist_max; /*< Maximum transmission distance*/
    //Bandwidth Values
    uint8_t b_tx_time; /*< Packet Transmission Time */
    uint8_t b_pcol; /*< Probability tx collision */
    //Historic Values
    float historic_max;
    uint8_t historic_counter;
} ivc_cost_parameters;

```

---

Listing 5.2: Cost Model Struct

third function calculates the decision based on the decision parameters. Separating out the value/cost and decision parameters, allows the system to be flexible to both new valuation and costing calculations without affecting the framework, as well as allowing different decisions to be taken using the same code.

The final function `ivc_update_is_vm()` is called when new data is received from the network and updates the spatial valuation model for the value parameters. Information models need to be maintained to ensure spatially aged data does not affect the valuation of data. When data is received from the network, both the value and costing models are updated: the value model to update the spatial information, and the cost model to update the probability of collision.

The value and cost models use 82 and 12 bytes of RAM respectively in this implementation, which is thus not too taxing for a microcontroller with 1KB of RAM. The

---

```

uint8_t ivc_calc_value(uint8_t *data, uint8_t len, ivc_value_parameters *vp);
uint8_t ivc_calc_cost(ivc_cost_parameters *cp, uint8_t len);
ivc_decision ivc_compare_vc(uint8_t val, uint8_t cost, ivc_decision_param *dp);
void ivc_update_is_vm(uint8_t node_id, uint32_t val, ivc_value_parameters *vp);

```

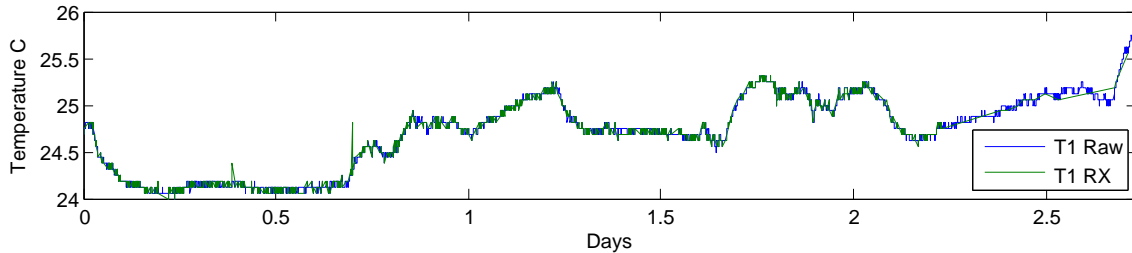
---

Listing 5.3: Information Valuation and Costing Functions

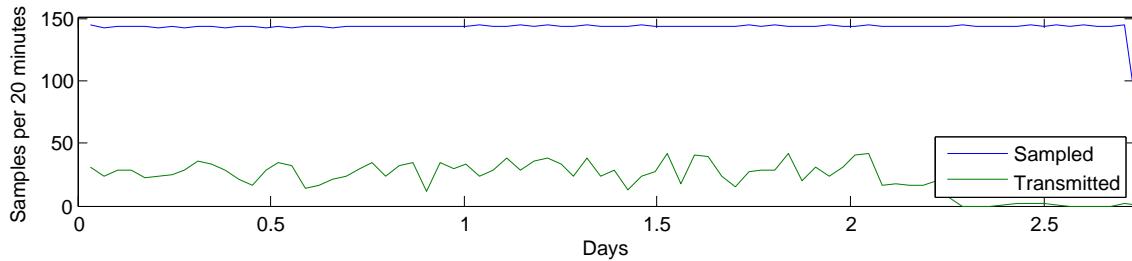
implementation does however make use of a reasonable amount of heap space, meaning that memory is still at a premium and the mcu cannot make full use of its 1KB implementation as a large portion of it is used as heap space.

### 5.1.5 Results

Testing of the miNet platform was done through functional and experimental tests. To functionally test the valuation and costing system running on the miNet nodes, we ran an experiment over three days to value temperature information of three colocated nodes. Sampled temperature was recorded by the three nodes connected to a pc, as well as recorded by a collection node recording the shared information signal between the devices. The signal was sampled every 20 seconds, and the information was valued and costed before transmission.



(a) Temperature Model Sampled Signal vs Recieved data



(b) Number of samples transmitted vs sampled over 20 minute intervals

Figure 5.7: Functional Test showing miNet working over three colocated nodes measuring temperature

Figure 5.7a shows a comparison of the signal recorded on the node with the signal received. In this figure, there is a spike on the received signal at about day 0.7, representing an erroneous packet. It is clear from Figure 5.7 that the signal has very little loss in signal quality while decreasing the number of samples transmitted from on average 148

samples to on average 44 samples per 20 minute interval. This shows a saving of around 70%.

It can be seen in Figure 5.7a that this signal still appears to be above the required sampling rate as no aliasing is present. This is due to the value model having not experienced extremely valuable data from the sensor input.

Around day 2.25 that the signal has experienced a larger change value which then through the normalisation of the value of the information thus decreased the overall data rate.

The temporal value models of each sensor signal can be seen in Figure 5.8. In this figure a comparison of different BSN signals and their associated temporal information valuation using the models shown previously in this section. These temporal models are combined with the dynamically changing spatial value models and the cost models of each node.

Figure 5.9 shows an ECG signal taken from a patient along with the received signal at a sink node. The information valuation and costing system continuously evaluates the signals to show that the several cases where the R-wave is smaller than usual: representing a large dynamic change, the signal is transmitted to the receiver, whereas the other ECG signals which are not valued as important, are not transmitted. The valuation of the data on the cardionode includes the beat detector shown in Section 5.1.3.2 and heart rate information. It is thus calculated by the information valuation and costing system that when the beat or heart rate information changes, a large information value is created and the information is disseminated into the network. The cardionode transmits a stream of datapoints showing the behaviour visualised in Figure 5.9. It should be noted that the node has been running for a long time in this example, so as to have already transmitted samples of the previous signal.

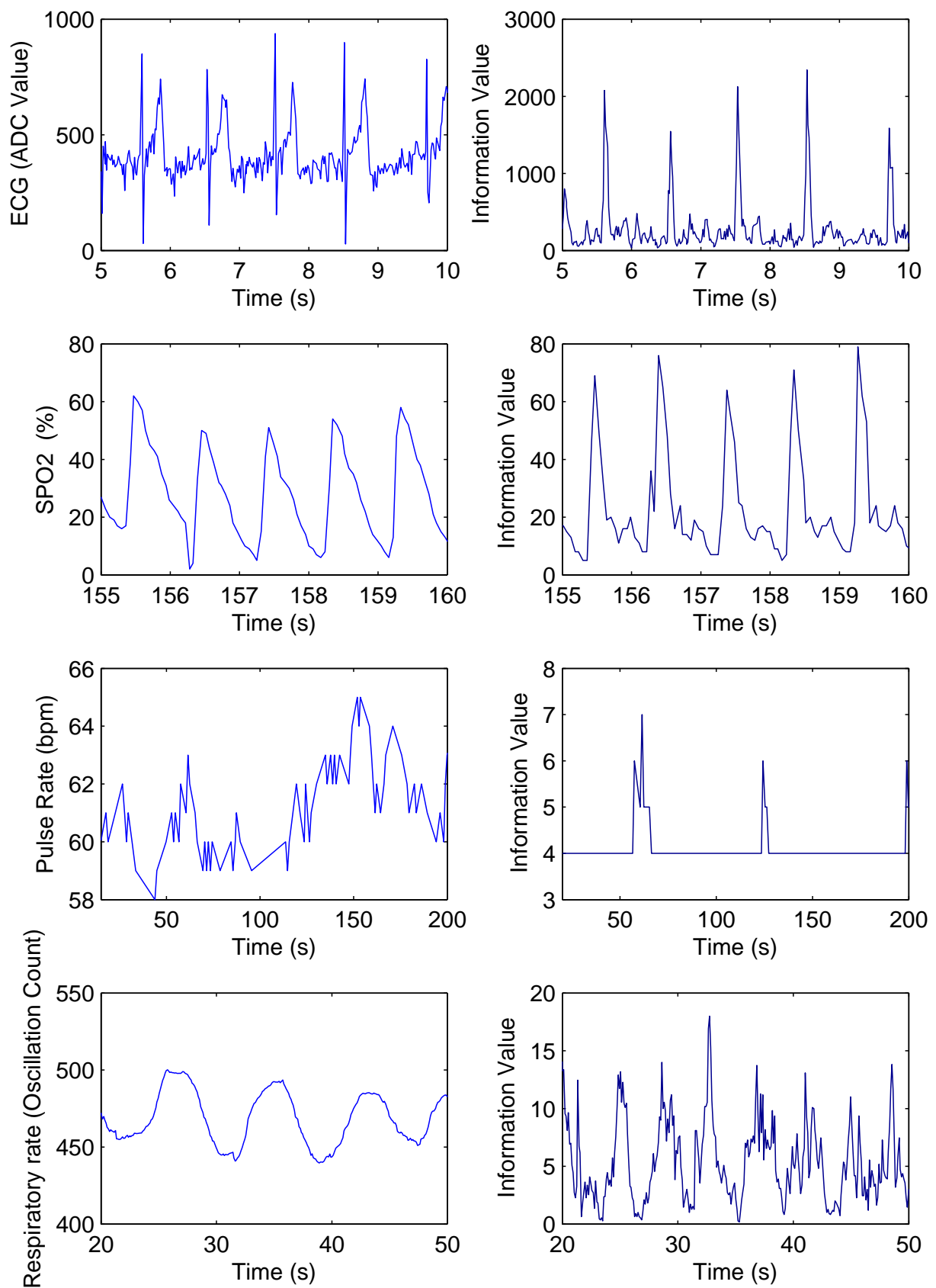


Figure 5.8: Temporal Information Valuation on Real miNet data

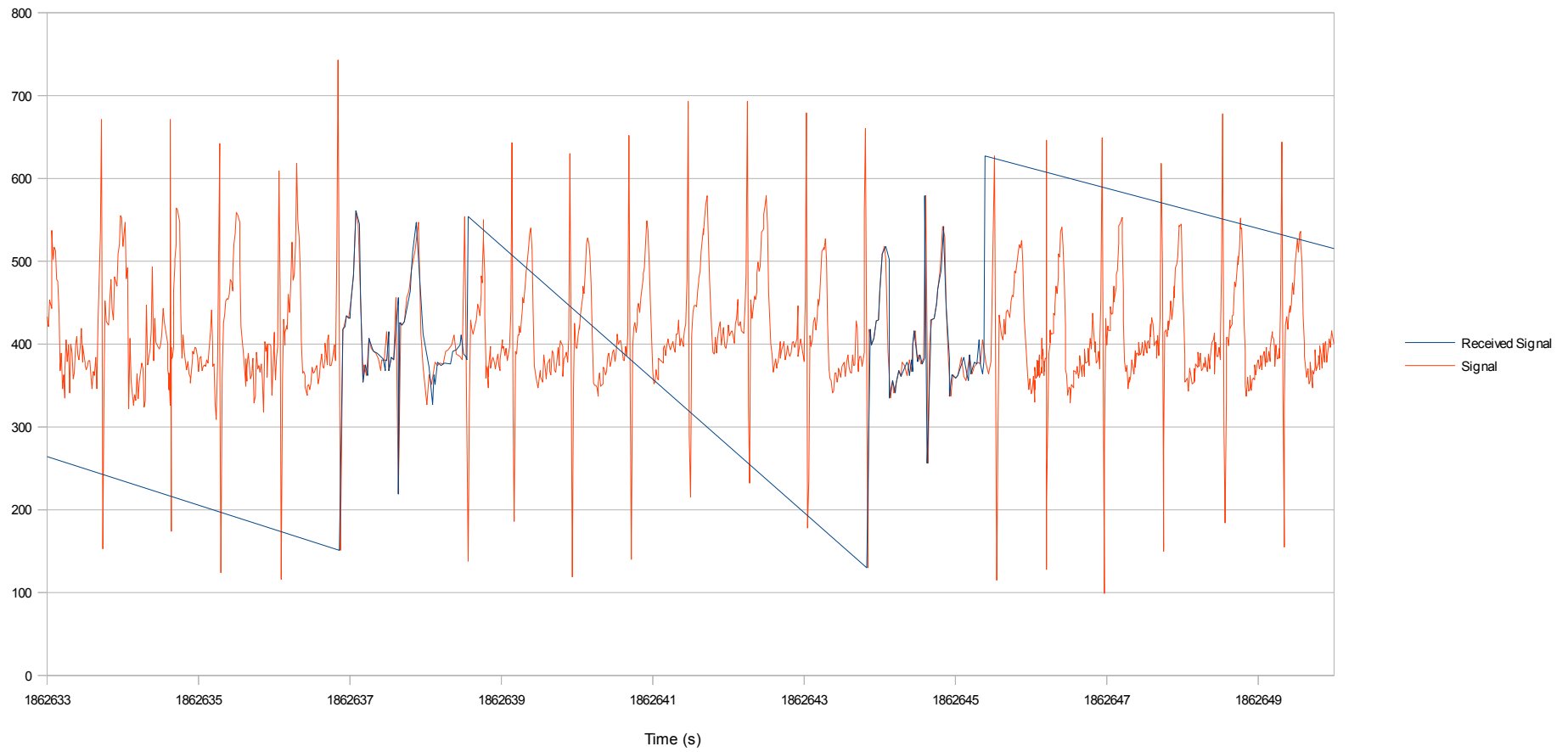


Figure 5.9: Signal vs Received Signal of Information Valuation and Costing system active. Notice that only the signals which are not similar to the other signals are transmitted



### **5.1.6 Conclusions**

In this section we have designed and developed a prototype horizontally distributed body sensor network platform for healthcare. This platform builds on the Information valuation and costing framework designed in Section 4.1 and the Heterogenous wireless node framework in Section 4.2.

Looking at the results, it can be seen that the system operates as expected, but no energy consumption information is provided and should be done in future work to see whether the tools used enable long term body sensor networks.

## 5.2 Memory Assistance Device: DejaView

The DejaView system follows the Tiered Vertical Distribution framework designed in Section 4.3. Figure 5.10 shows an image of the second generation DejaView device. The third generation DejaView device, not shown in this work, incorporates the camera interface developed in Section 3.2.



Figure 5.10: The DejaView Device Version 2

The following section describes the implementation of the system and results of its functionality.

### 5.2.1 Conceptual use and Scenario

The conceptual design for this system is a portable, unobtrusive and continuous use prospective memory aid for an early stage dementia patient. A prospective memory aid aims to assist a patient in remembering future events, actions or activities which the patient should perform, or would like to perform, but has not remembered to do so. The system attempts to remind users of important factors in their current environment classified into cues relating to people, places, objects and actions. The concept of the system has a number of significant advantages over existing devices [4], such as the Microsoft Sensecam [123], representing a step change in the capability of technology-based memory aids. These include:





- A low power, wearable, intelligent device which autonomously captures images and sensed information to cue autobiographical memories.
- A web-enabled, wireless system that integrates with the users mobile handset to add contextual information, to provide feedback and to link the DejaView device to the Internet.
- The automatic annotation and analysis of images, using multiple distributed databases, allowing the system to effectively present the user with the contextual cues of relevant images, and allow further effective querying from a knowledge repository.
- Supporting prospective memory in a periodic planning process, and also as a real-time prospective memory aid.

To illustrate the functioning of the device, a walkthrough example is presented below and Table 5.5 shows a sample description of images, the related actions and valuation of information associated.

*Mr Jefferies wakes up in the morning by his alarm clock on his mobile handset. The reminder message alerts him to remember to put on his DejaView Device. Having put on his clothing, and DejaView Device, he walks to the kitchen and turns on the kettle. As he is preparing his cup of tea, his doorbell rings, which he goes to answer. It is his neighbour, asking him whether she can borrow a cup of sugar. He cannot remember her name, so as he fetches the sugar, he glances down at his mobile phone, and sees that the DejaView Device has taken several images already, uploaded them, analysed them, and is informing him that he was talking to Mrs Jones who lives next door. Having fetched the sugar and giving it to Mrs Jones, he asks her how Mr Jones is doing. They converse for a short while, and afterwards he heads back indoors and sits down on his chair. As he looks down at his mobile handset, he sees that he was making a cup of tea, and heads back into the kitchen to continue making his cup of tea. Having had his breakfast, Mr Jefferies starts walking to the corner shop for his weekly groceries. As he leaves the house, and starts walking away from his car, his phone vibrates, and a message is displayed on his phone reminding him that he has a doctors appointment in twenty minutes for which he will be late for if does not drive there soon. Mr Jefferies decides to postpone his shopping and go to the doctor instead. That evening, whilst at home, Mr Jefferies reviews his calendar and labels the images related to his doctor. He adds the date of his next appointment.*

Table 5.5 illustrates examples of how this system can support the user's memory. The first column shows an image which the system has captured as it believes this image is valuable. The image is broadly classified into a type by the sensors and rules which activated the image capture. Column two describes the output of the data analysis engine, showing the patterns which have been recognised by the system. Column three shows examples of the messages which could be output to the user and a description of

Table 5.5: Sample Images and discussion of the assistance the memory can provide

Image & Type	Patterns Recognised	User Message, memory assisting function	Valuation
People 	Face:Dave, Face:Karim	<b>Talking to Dave and Karim. Spoke to Dave last week Tuesday afternoon.</b> Remind the user of the peoples names. Remind the user of when last they talked to these individuals	<b>Important situation,</b> Increase the image capture rate.
Place 		<b>Walking down Hallway at 34 Destiny Road. Appointment at 4:30pm. Room 23, 34 Destiny Road.</b> Reminding user of an appointment and where they are.	<b>Unimportant situation.</b> Very little image data captured. Decrease the image capture rate.
Object 	Object:Fan, Object:Hand	<b>Turning fan on. Fan must be turned off before leaving office</b> Remind user of operating requirements for equipment interacted with	<b>Reasonably Important situation</b> Fan is now on. Remember to turn it off.
Action 	Object:Hand, Object:Cup, Object:Milk	<b>Pouring a cup of tea. Have you added sugar?</b> Has the fridge been closed and the kettle turned off?	<b>Important situation,</b> An action is being performed. Remember to close fridge and turn the kettle off.

memory assisting function the message is providing. The fourth column shows how the information is valued, and the resulting actions which need to be taken.

### 5.2.2 DejaView Device

The DejaView device is intended as a small, lightweight peripheral camera to be worn at eye-level and capture continuous daily activity of one's life. To ensure the camera does not have an energy expensive imaging sensor on continuously, it uses an array of low power sensors to capture events which will only enable the high power camera sensor to capture an image when it is regarded as valuable.

To ensure the device size and power requirements are met, we need to ensure that the most important sensors are used, to attain the most relevant data for the least amount of overall energy.

To select the most relevant sensors, a list has been compiled of what is available to sense the relative value of the readings, and how the sensor could be used to sense.

In the paper by Lee and Dey [124], memory recall is achieved by caregivers through the use of memory cues. Memory cues are used to remind a patient of the events which are deemed to be able to be recalled by the patient. Good memory cues were categorised into Person, Object, Place and action cues. From the paper, the order of importance of the different cues (first being most important) are: Person, Action, Object and Place.

To record these events, we wish to use a camera to record images associated with these cues. We will attempt to sense events related to these cues, using sensors which can actively add value related to these cues. To evaluate which sensors to use, we developed a scoring criteria to assess different currently available sensors.

Table 5.6: DejaView Sensor Selection

Sensor	Manufac- turer	Sample Device	E/samp (nJ)	Vol (mm <sup>3</sup> )	Focus	Relative Value	Relative Size
<b>Used</b>							
Microphone	Knowles	SPM0208HD5	84.38	22.18	Person	5	4
Three axis compass	Honeywell	HMC5843	450	21.28	Action	4	4
Light level	Avago	APDS-9008	672	1.32	Place	3	5
Accelerometer	Freescale	MMA7455L	$39.36 \times 10^3$	15	Action	3	4
Passive Infrared	Panasonic	AMN32111	$24.3 \times 10^6$	1621.46	Person	2	2
<b>Not Used</b>							
Humidity	Honeywell	HIH4030	$56 \times 10^3$	95.64	Place	2	4
Temperature	Maxim	DS1820B	$2.25 \times 10^6$	36	Place	2	4
Heart Rate Monitor	Polar	RMCM01	$10.8 \times 10^3$	1416	Health	3	2
GPS	RF Solns	GPS-310FS	$742.5 \times 10^6$	1142.4	Place	2	2
Blood Pressure	Tatung	TMD-36AB	$120 \times 10^9$	131040	Health	1	1

Table 5.6 shows a compiled table of these sensors and their scores. The evaluation criteria used is the relative value the sensor provided summed with the relative size

score. The relative value score is defined as a score between 1 and 5 (5 being energy efficient, 1 being energy inefficient) related to the order of the energy cost of each sample while the relative size is defined by rank 1 to 5 (5 being small, 1 being large) of the order of the volume the sensor occupies. It should be noted that although the Passive Infrared Sensor has a lower score than the sensors not selected, it was deemed important to use as it could detect “Person” types which were deemed highly important in the work by Lee and Dey [124].

It is clear from the table that no sensors can ascertain with which objects the person is interacting, and it is assumed that an RFID based system is currently too expensive and unrealistic to implement in a real-world case scenario, as all devices would require to be tagged, and the relative energy expense of a greater than 10cm RFID reader is an order or two of magnitude larger than any sensor used in this system. Object type cues could only be ascertained from the environment with which the person is interacting, through the use of camera images.

As this system relies on a mobile handset, limitations on the communication hardware were imposed. Mobile handsets providers have been implementing low power Bluetooth communications protocols for almost ten years now, while other lower power radio protocols, such as Zigbee, have not yet penetrated the market. Bluetooth was thus selected as the radio technology.

Bluetooth, although not the lowest power radio technology, does have a reasonably high data-rate which is required to transmit large images over the wireless link. To enable low power operation of this radio link, a suitable radio module was required which allowed for enabling low power modes, disabling connections which were not in use and low power listening modes.

The newly released Energy Micro MCU was selected as the preferred MCU. Although it does not have a camera interface, it was selected as a microcontroller as it was not only powerful, cost effective and extremely low power, but it had a considerably higher memory capacity in comparison with similar energy efficient MCUs such as the PIC and MSP430. High power microcontrollers containing a camera interface (such as the Blackfin and OMAP), were considered too computationally powerful and energy expensive to run this continuous system, and an alternative solution to the camera interface was sought.

The interface presented in Section 3.2 was designed for this, but to test the system in its entirety, a simple serial camera interface was used in this version of the system.

Figure 5.11 shows an overview of the architectural components of the device. The device prototype can be seen in Figure 5.10 and circuit diagrams for the device are given in Appendix B.

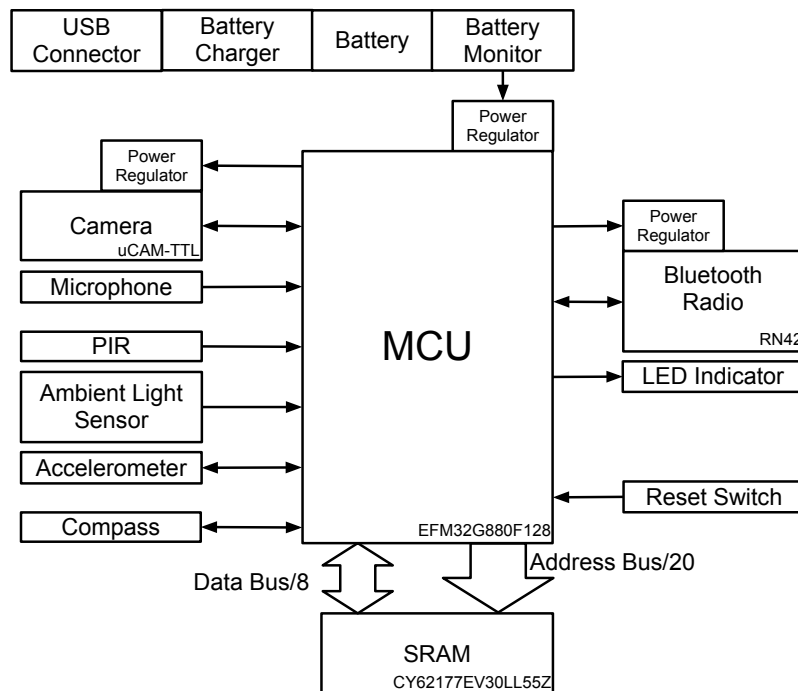


Figure 5.11: The DejaView device overview

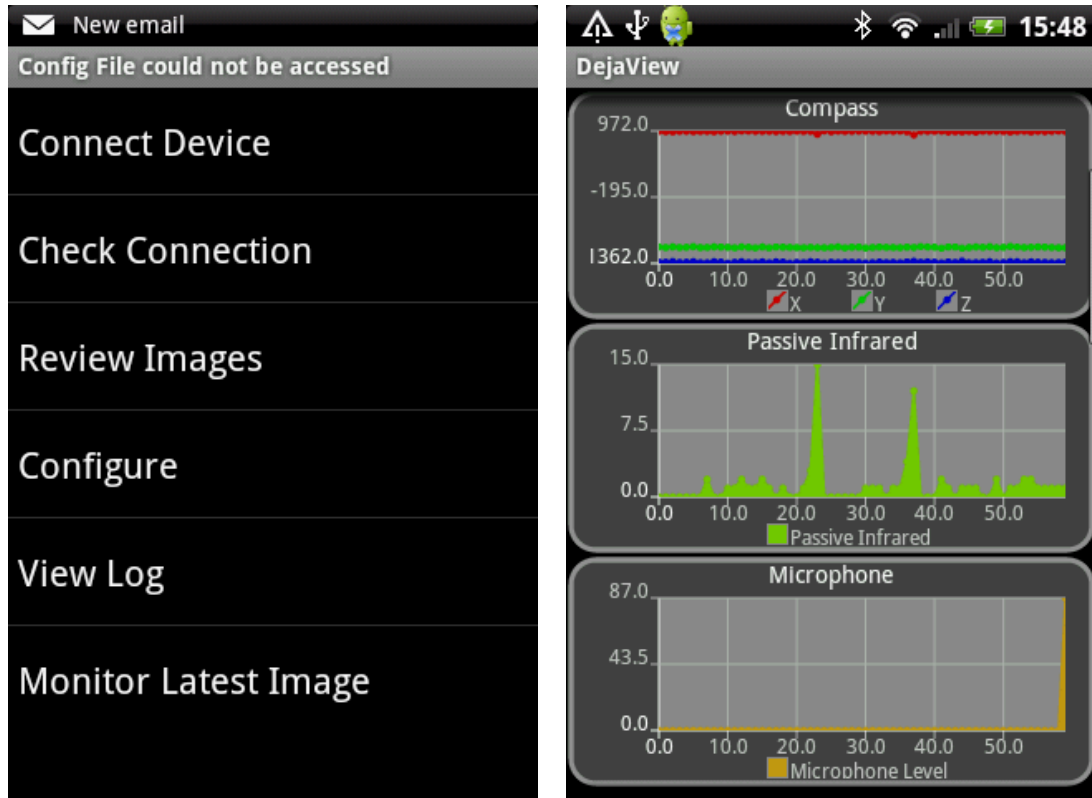
Due to the relatively high power and network neighbour constrained Bluetooth radio, the Information Value and Costing system described in Section 4.1.2 is not used. Similarly, due to the Dejaview Device having to only communicate with a single other device, the Android application, the abstraction and tools provided by SWNF in Section 4.2 does not provide any remarkable benefit and is therefore not used in its entirety, as the sensor drivers provided by the SWNF implementation are used.

### 5.2.3 Android Application

The mobile phone application was developed by the author as an Android application consisting of two separate components: a Background Service to manage the bluetooth connection, and an Activity for interaction with the user.

To use the system in as low power a mode as possible for both the DejaView Device and the mobile handset, the mobile handset was selected to run as a server, waiting in low power for a connection request to initiate. This allowed the DejaView Device to operate with the Bluetooth radio off, until it was required to start communication.

Figure 5.12 shows some snapshots of the android application.



(a) Android Application Main Menu

(b) Data collected from device

Figure 5.12: Data captured by the Dejaview System

#### 5.2.4 Internet Service Application

A server application was developed in PHP to allow access to the data and images gathered by the system. The system integrates with Google Maps as a demonstration of appending external database information to the data gathered by the DejaView system. A snapshot of the system is demonstrated in Figure 5.13.

#### 5.2.5 Results

Initial testing of the system was performed through energy tests of the device to see how much energy was expended on information capture. Figure 5.14 shows the device running normally in sensing mode. It can be seen that the current is nominally high for the device running at around 5mA. The majority of this current is keeping the Bluetooth radio in constant listening mode, and this will be decreased in future implementations. The Bluetooth current is composed of both a constant 2-3mA on current, and a 40mA current transmit spike seen operating every ~2 seconds. The smaller spikes in the figure are the sensors turning on and capturing data.



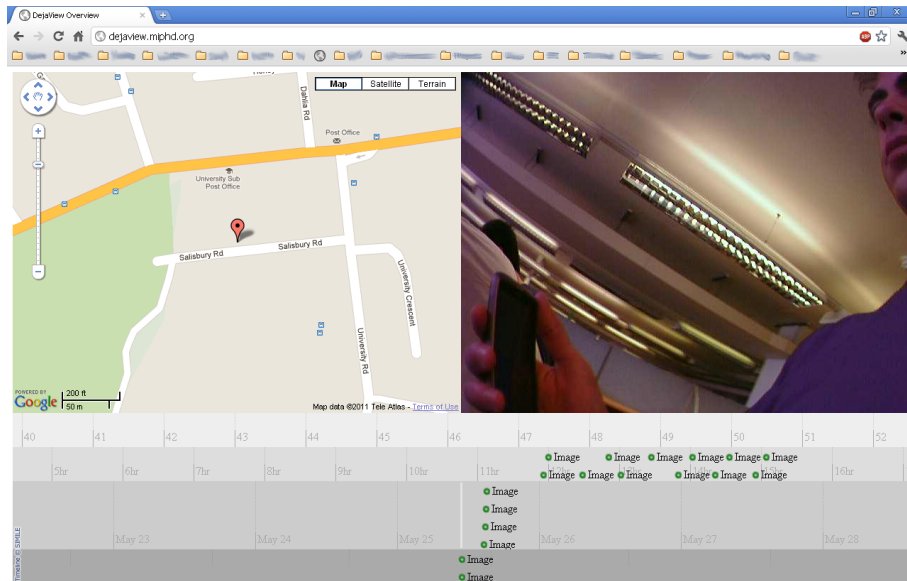


Figure 5.13: A Screenshot of the Internet Service Application

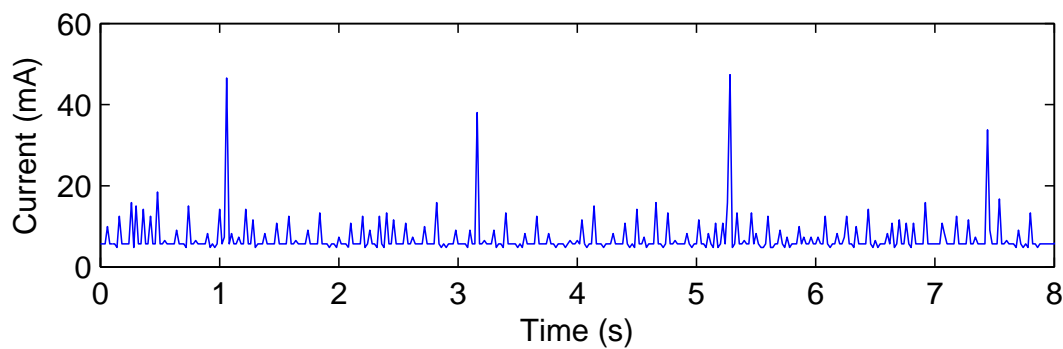


Figure 5.14: DejaView Device Sensing Current Draw

Figure 5.15 shows the current profile and timing of an image capture of the device. In this figure, the sensors activate the camera, enabling the image capture shown in the Capture section. The different stages of capture and processing are then demonstrated in terms of current.

Table 5.7 shows the calculated timings and energy consumption of the device during image capture. From this table it can be seen that the energy used on setting up and transferring data over the radio contributes a large part of the energy consumption, while the capture of the information only contributes  $\sim 45\%$  of the total energy used during capture and transmission.

As an initial evaluation of the system, some of the images captured by the system are shown in Figure 5.16 along with the data signals which triggered rules used to capture the images.

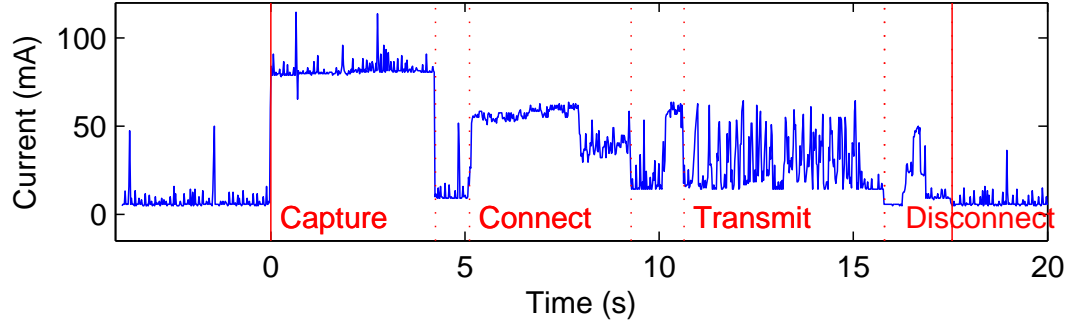


Figure 5.15: DejaView Device Capture Image Current Draw

Table 5.7: Timing and Energy comparison of information capture

	Time (s)	Energy (mJ)
<b>Camera</b>		
Image Capture	4.24	1216
Append Sensor Data	0.88	38
<b>Bluetooth</b>		
Connect	4.16	743
Enter High Speed Mode	1.36	155
Data Transmission	5.16	510
Disconnect	1.74	95
<b>Total</b>	<b>17.54</b>	<b>2757</b>

In these examples, the individual axes of the accelerometer and compass information have been combined and averaged for visual purposes. Capture was initiated at time zero, but further information was appended to the signal due to the delay between capture completion and transmission of the image.

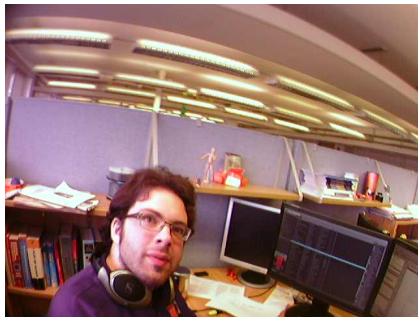
### 5.2.6 Conclusions

In this section a prototype vertically distributed body sensor network for supporting memory is presented. This system shows an implementation of the tiered vertical framework presented in Section 4.3 and a use-case for the Camera Interface presented in Section 3.2.

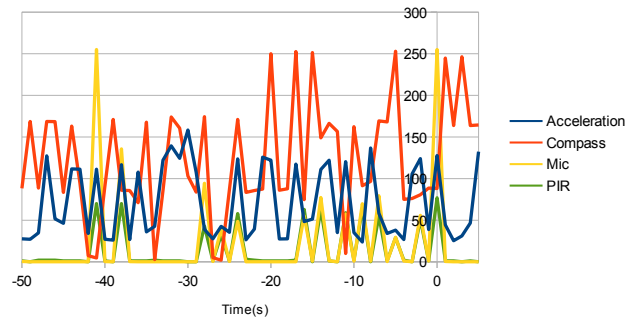
The system has been demonstrated to work through initial testing and the energy consumption of the low power DejaView device shown to operate in a low power manner.

Using the Tiered Vertical Framework provided in Section 4.3, this system can be seen to be able to support complex processing of data captured by the system on computationally power processing clusters, although use thereof has not been demonstrated.

By utilising the XML Information shards for data transport in the system allows for a rapid development time as these can be easily integrated into Android and Web Applications.



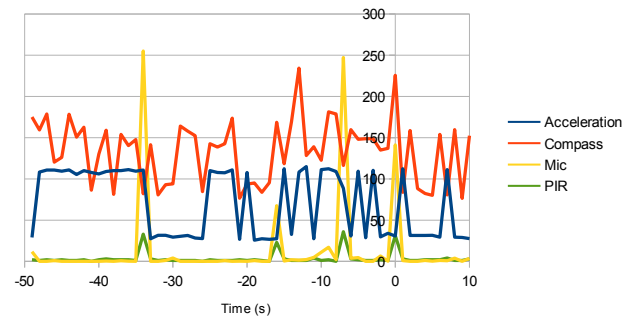
(a) Person Example 1



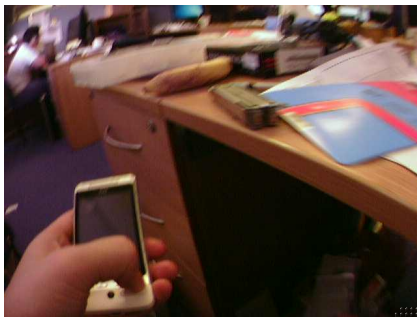
(b) Person Example 1 Data



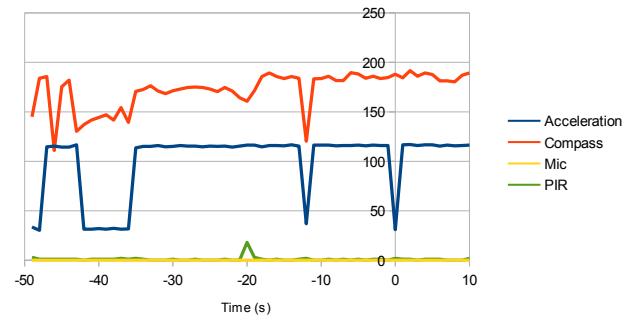
(c) Person Example 2



(d) Person Example 2 Data



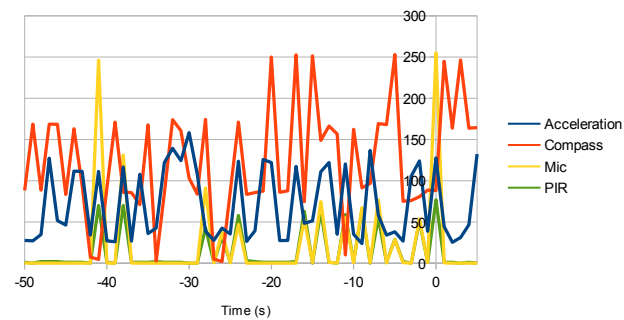
(e) Object



(f) Object Data



(g) Place



(h) Place Data

Figure 5.16: Data captured by the Dejaview System showing the associated signals

### 5.3 Discussion

One of the problem areas this section has brought up, is the need for application developers to have a good understanding of many different technological tools for heterogeneous devices. It is important that future Body Sensor Networks work on numerous different devices and at different levels, but a stringent requirement is for the devices to co-operate and interwork seamlessly.

In this chapter we have demonstrated two prototype body sensor network platforms using the technologies and systems developed in Chapters 3 and 4. In the first prototype platform we have demonstrated the use of the frameworks presented in Section 4.1 and 4.2 and made use of the novel sensor designed in Section 3.1. The second vertical prototype network platform demonstrates how the camera interface developed in Section 3.2 can be used along with the frameworks developed in Section 4.2 and Section 4.3.

The horizontal network presented in Section 5.1 shows that a system of nodes using the information valuation and costing system can work cooperatively in assessing the information and thus together reduce the network traffic. A vertical network is presented in Section 5.2 showing how information valuation is still important for these kinds of networks, but this valuation can be performed by a group of sensors using rules on the low power device.

Standardisation of signals communicated between nodes is also of high importance. It is important that the standard conforms to the requirements of the application.

It is clear that the two networks presented in this Chapter can work together cooperatively, but further work is required to enable this.



## Chapter 6

# Conclusions and Future Work

In this work, we set out to develop a set of enabling technologies to realise distributed BSN platforms. The aims were to understand the pros and cons of distributed BSNs in order to achieve a universal BSN for pervasive, self-managed healthcare. In doing so, we developed numerous different components for BSNs, gaining insight into the complexities and requirements of such systems. Many conclusions can be drawn from this research, which have been categorised into the different sections below.

### 6.1 Developing Low Power Sensors and Sensor Interfaces

A novel respiratory sensor which uses changes in permittivity near the skin surface to measure the respiratory rate of a person was designed, developed and analysed. This sensor could replace RIP as it does not require a surrounding chest strap; only a patch. The sensor can also be duplicated and the signals correlated for higher quality signals.

In Section 3.2 a camera interface for low power applications implemented using a CPLD, SRAM memory and Microcontrolling unit was developed. Using these components allowed us to capture high resolution still images using only 522mJ. This can allow for new high resolution images to be periodically captured for long term use.

Continuous body worn sensors need to be extremely low power. Signals generated from on-body (Respiratory Sensor) and off-body (Camera Interface) both require a certain amount of processing to decrease the quantity of the data to acceptable levels. It is not viable to transmit the data before evaluation as the quantity of data vs the quantity of information in the sensed signal is generally an order of magnitude different.

## 6.2 Software Frameworks supporting Distributed Sensor Networks

Current frameworks for BSN research are not adequate for enabling distributed BSNs. We have developed three frameworks to enable a next generation of distributed BSNs. These frameworks enable key requirements for distributed BSNs:

- **Data/Information distribution:** Information may be required to be pre-evaluated before expensive transmission. Using an information valuation and costing system presented in Section 4.1.2, allows the nodes in the network to assign a value to the information in terms of its spatial and temporal structure, thus allowing it to control the network in a distributed fashion.
- **Heterogeneity with minimum power usage:** Using the SWNF framework presented in Section 4.2 allows developers to use a common framework across numerous node platforms which fits the model of a wireless sensor node better than an operating system.
- **Task distribution:** Using the Tiered Vertical Architecture shown in Section 4.3 allows high processing computing clusters to process information sensed by low power nodes.
- **Versatility** These frameworks have been designed and implemented to attempt to be as versatile as possible and can thus be used on numerous different hardware platforms and are independent of the underlying technologies.

## 6.3 Prototype Horizontal and Vertical Body Sensor Network Platforms

The two platform implementations demonstrated show how distributed BSNs can distribute both data and tasks horizontally and vertically. These implementations show how a BSN could achieve multiple different sensing tasks simultaneously. By demonstrating the independent sensing tasks, we can see that it is feasible to achieve a universal distributed BSN.

The tools developed in this work have been shown to enable distributed BSNs. This work suggests that for best energy usage, sensing should be distributed horizontally, while processing should be distributed vertically.

## 6.4 Discussion

These two platforms enable a new generation of distributed BSN research. Multiple parameter sensing systems such as miNet are feasible, but the communication bandwidth of the nodes is required to be an order of magnitude higher than the output of the sensed parameters on a single node, as dense distributed nodes require more bandwidth to ensure communication overhead does not overburden the network and thus spend a large amount of energy on network traffic management. Using the system defined in Section 4.1.2 does alleviate some of the network burden, but further work is required to ensure this is done well.

Endless supplies of data can be gathered from BSNs, but it is clear that there needs to be some form of unified understanding of what the information is for and what it means. Gathering the data is clearly an achievable task, and as technology improves, more data may be available on any person. Current understandings of Electronic Patient Records [125] do not support a means of filtering data, and it is clear from this research that a stronger method of encapsulation of data obtained from a patient is required. An Electronic Patient *Model* might abstract the notion of a patient health record more coherently and understandably providing future physicians and healthcare providers with a rapid understanding of a patient's captured data. Allowing for functional code to be run on a patients electronic patient data would allow healthcare providers to gain a rapid understanding of the patient model, which in turn enables the provider the ability to address the patients conditions within an environment which the provider understands. This assists a healthcare provider in delivering effective healthcare. These new forms of healthcare can now be pursued as a result of developing these enabling technologies.

With these enabling tools we can dynamically change the network's sensing tasks to what we believe to be important in the network. Although this is now feasible, the question of what information is *important* to be captured still remains. If we prioritise certain sensing tasks as opposed to others, why are they more important even if they contain less information? These tasks are related to the Semantic Information related to the patient under observation. Although the vertical distribution system allows for dynamic changing of the Semantic Information, we still do not have an understanding of what the semantic information regarding the patients is.

## 6.5 Future Work

In this work several key issues relating to BSNs have been investigated and addressed and in some cases assessed how suitable the solution to the issue is. It is clear that due



to the nature of BSNs, developing platforms for them requires in-depth knowledge of several skill-sets in developing such systems.

Although this work is preliminary in addressing several of the issues related to distributed BSNs, it is by no means complete and a great deal of further work is required to fully realise and enable distributed Body Sensor Networks.

In Chapter 3 a sensor and sensor interface were developed. The respiration sensor, although novel, has not been fully tested in terms of its energy cost or long term use. Further testing of the sensor on numerous patients in numerous settings before the sensor can be realised beyond an initial prototype. The results presented in this section are not exhaustive and can be improved with more numerous trials.

The sensor interface presented in the same chapter allows a low power MCU access to camera images. This interface is useful for numerous applications beyond BSNs due to its low power consumption and reasonable small size. Unfortunately, due to the nature of the low power MCU, it is also very limiting in terms of what the MCU can do with the image data once captured. This interface should merely be regarded as a store and forward system of the image data as any processing of the image data with the computationally low power MCU can be more energy expensive than comparable computationally high power MCUs due to the amounts of time involved. Modern computationally high power MCUs appear to be addressing the concerns of the high sleep currents and are thus becoming more suitable for these kinds of applications. The only remaining advantage of using this interface is that of the costs of the interface components relative to the high cost of the computationally high power MCUs.

Developing sensors for BSNs takes time and care to ensure small size and low energy usage while still providing useful information. It is thus important to thoroughly research, investigate and assess these three criteria when developing new sensor and sensor interfaces for BSNs.

Chapter 4 presents three software frameworks which aim to address several concerns related to distributed BSNs.

The first framework, an Information Valuation and Costing framework, uses an approach whereby the implementation limitations of current BSNs are prioritised (such as the processing ability of the low power nodes) to ensure the system is realisable on current hardware. This approach limits the useful lifespan of this system as Moore's and Gene's Laws show smaller, lower power and more computationally power processors will be available in the future, thus allowing more complex and less lossy systems to be used. This does not mean that the framework will be rendered completely useless, as future work to expand on the valuation and costing models in line with the improvements of processors could improve the usefulness of the system.

The value and costing models used in section 4.1.2 are merely initial models showing how they can be used in the context of these distributed BSNs. An in-depth design, development and analysis of different and more complete models is required to gain a better understanding of the dynamics of the distribution among the sensor nodes which these models generate.

In reflection, the dataset which this section is tested on is not ideally suited to demonstrate the usefulness of distributed BSNs, but due to the limited availability of BSN datasets, it is thought that this dataset is sufficient. In future, the system presented in this section should be run on datasets orientated more to BSNs.

The second framework presented in section 4.2 provides a software platform to rapidly develop BSN applications. This framework provides a solution beyond BSNs and can be used for WSNs as well. It is a simple and small framework and due to its small nature, is not necessarily suitable for computationally powerful processors requiring real-time behaviour. The framework has been shown to be similar and competitive against TinyOS, but with future work, it is the authors belief that the framework can achieve better results in terms of energy consumption, Flash footprint and RAM overhead than TinyOS. More development time is required to achieve these results. Due to the comparison of the software systems on different hardware, it is difficult to make direct comparisons, and thus it is important to realise the SWNF framework on hardware which is currently supported by TinyOS. This would allow for a more direct comparison and is thus left for future work.

The tiered vertical distribution framework shown in section 4.3 is retrospectively similar to the SAPHE architecture differing only in the ambient environment of the SAPHE architecture and the processing clusters of the tiered vertical distribution framework. As this architecture is just proposed, further development, implementation and testing of this architecture can be done beyond that shown in section 5.2.

The prototypes developed in Chapter 5 are simply proof of concept platforms, and further investigation and testing is required for these prototypes. It is difficult to assess the value created by these prototypes without a good understanding of the value generated by the collected information. It is thus imperative to find and isolate specific and immediate examples whereby these platforms can assist and support a defined outcome or action. These outcomes or actions such as those in table 5.5 for the DejaView system provide defined goals to assess the vertical system, and similar goals should be isolated for the horizontal platform to be tested against. These high level goals define the overall importance of these forms of distributed BSNs and are thus the most important future work to isolate and develop.

The initial prototype platforms developed in Chapter 5 are not mutually exclusive, and can be used together to enhance the network even further. To enable this, a universal architecture encompassing both horizontal and vertical networks designed around BSNs

for healthcare is required. This architecture should be designed specifically for this kind of network, as design goals for these networks is already diverse and difficult to design for.

## Appendix A

# SWNF Example Applications

**A.1** Null Application

**A.2** Low Power Listen Example Application

**A.3** Oscilloscope Example Application

---

```
/**
 * Application Template File
 *
 */
#include "swnf_cfg.h"
#include "swnf.h"
extern swnf_energy_settings energy;

/** Energy Service Response */
void swnf_energy_service_response(swnf_message_t *message) {
}

/** Sensor Service Response */
void swnf_sensor_service_response(swnf_message_t *message) {
}

/** Communication Service Response */
void swnf_comm_service_response(swnf_packet_t *incoming_packet) {
}

int main() {
    swnf_init(); /** Initialise SWNF */
    energy.sleep_state = ACTIVE;
    swnf_start(); /** Start SWNF */
}
```

---

Listing A.1: Null Application

---

```

/**
 * Low Power Listen
 *
 */
#include "swnf_cfg.h"
#include "swnf.h"
#include "leds/leds.h"

extern swnf_energy_settings energy;
extern uint8_t node_id;
swnf_packet_t outgoing_pkt_buf;

static void Delay(uint16_t milli) {
    uint16_t start = swnf_get_ticks();
    while ((milli + start) > swnf_get_ticks()) { //Do Nothing
    }
}

/** Energy Service Response */
void swnf_energy_service_response(swnf_message_t *message) {
}

/** Sensor Service Response */
void swnf_sensor_service_response(swnf_message_t *message) {
    switch (message->type) {
        case MSG_TYPE_SENSOR_BUTTON_PUSH:
            break;
        default:
            case MSG_TYPE_SENSOR_DATA_READY:
                break;
    }
}

/** Communication Service Response */
void swnf_comm_service_response(swnf_packet_t *incoming_packet) {
    led_toggle(GREEN_LED);
}

uint8_t counter = 0;
uint8_t sendSkip = 0;
int16_t sendInterval = 0;

void nextLp1State() {
    switch (counter >> 5) {
        case 0:
            sendSkip = 0;
            sendInterval = 0;
            energy.sleep_state = ACTIVE;
            break;
        case 1:
            sendInterval = 100;
            energy.sleep_state = ACTIVE;
            break;
        case 2:
            sendInterval = -1;
            energy.sleep_state = LISTENING;
            break;
        case 3:
    }
}

```

---

```

        sendInterval = 0;
        break;
    case 4:
        sendInterval = -1;
        energy.sleep_state = LISTENING;
        break;
    case 5:
        sendSkip = 7;
        energy.sleep_state = LISTENING;
        break;
    }
}

void lpl_timer() {
    counter++;
    if (!(counter & 31))
        nextLplState();
    if ((counter & sendSkip) == sendSkip) {
        if (sendInterval >= 0)
            Delay(sendInterval);
        tx_packet(&outgoing_pkt_buf);
        led_toggle(RED_LED);
    }
    swnf_message_t delayed = MSG_DELAYED_FUNCTION;
    delayed.function = lpl_timer;
    delayed.data = swnf_get_ticks() + 1000; //Every Second
    swnf_post(delayed);
}

int main() {
    swnf_init(); /** Initialise SWNF **/
    outgoing_pkt_buf.id = node_id;
    swnf_message_t delayed = MSG_DELAYED_FUNCTION; //Post Initial Function
    delayed.function = lpl_timer;
    swnf_post(delayed);
    swnf_start(); /** Start SWNF **/
}

```

---

Listing A.2: Low Power Listen Example Application

---

```

#include <stdlib.h>
#include <stdbool.h>
#include "swnf_cfg.h"
#include "swnf.h"
#include "adc/adc.h"
#include "leds/leds.h"

#define BAT      11
#define TEMP     10

extern swnf_energy_settings energy;
uint8_t reading;
extern uint8_t node_id;
swnf_packet_t local;
uint16_t current_interval;
bool suppressCountChange;

/** Energy Service Response */
void swnf_energy_service_response(swnf_message_t *message) {
}

/** Sensor Service Response */
void swnf_sensor_service_response(swnf_message_t *message) {
    switch (message->type) {
        case MSG_TYPE_SENSOR_BUTTON_PUSH:
            break;
        default:
            case MSG_TYPE_SENSOR_DATA_READY:
                //Add the data to the adcbuffer
                local.readings[reading++] = message->data;
            }
    }
}

/** Communication Service Response */
void swnf_comm_service_response(swnf_packet_t *incoming_packet) {
    led_toggle(GREEN_LED);
    if (incoming_packet->version > local.version) {
        local.version = incoming_packet->version;
        local.interval = incoming_packet->interval;
        current_interval = incoming_packet->interval;
        if (incoming_packet->count > local.count) {
            local.count = incoming_packet->count;
            suppressCountChange = true;
        }
    }
}

void osc_function() {
    if (reading == NREADINGS) {
        tx_packet(&local);
        reading = 0;
        if (!suppressCountChange)
            local.count++;
        suppressCountChange = false;
    } else if (reading == (NREADINGS - 1)) {
        energy.sleep_state = ACTIVE;
    } else if (reading == 1) {
        energy.sleep_state = SENSING;
    }
}

```



```
    }
    adc_trigger(BAT);
    swnf_message_t delayed = MSG_DELAYED_FUNCTION;
    delayed.function = osc_function;
    delayed.data = current_interval;
    swnf_post(delayed);
}

int main() {
    swnf_init(); /** Initialise SWNF **/
    adc_init();
    local.interval = DEFAULT_INTERVAL;
    local.id = node_id;
    current_interval = DEFAULT_INTERVAL;
    suppressCountChange = false;
    energy.sleep_state=SENSING;
    //Post Initial Function
    swnf_message_t delayed = MSG_DELAYED_FUNCTION;
    delayed.function = osc_function;
    swnf_post(delayed);
    swnf_start(); /** Start SWNF **/
}
```

---

Listing A.3: Oscilloscope Example Application

## **Appendix B**

# **Circuit Diagrams**

### **Low Power Camera Interface**

**B.1** Camera, CPLD and Memory interconnections

**B.2** Low Power Camera Interface Microcontroller Interface

**B.3** USB and Power connections

### **miNet Boards**

**B.4** The cardionode sensor board circuit diagram

**B.5** The respinode sensor board circuit diagram

**B.6** The spoxnode2 sensor board circuit diagram

**B.7** The Accelnode/Storage node circuit diagram

**B.8** Energy Micro Sensor Board

### **DejaView Device**

**B.9** Power connections

**B.10** Sensor connections

**B.11** Bluetooth Communication connections

**B.12** Microcontroller connections

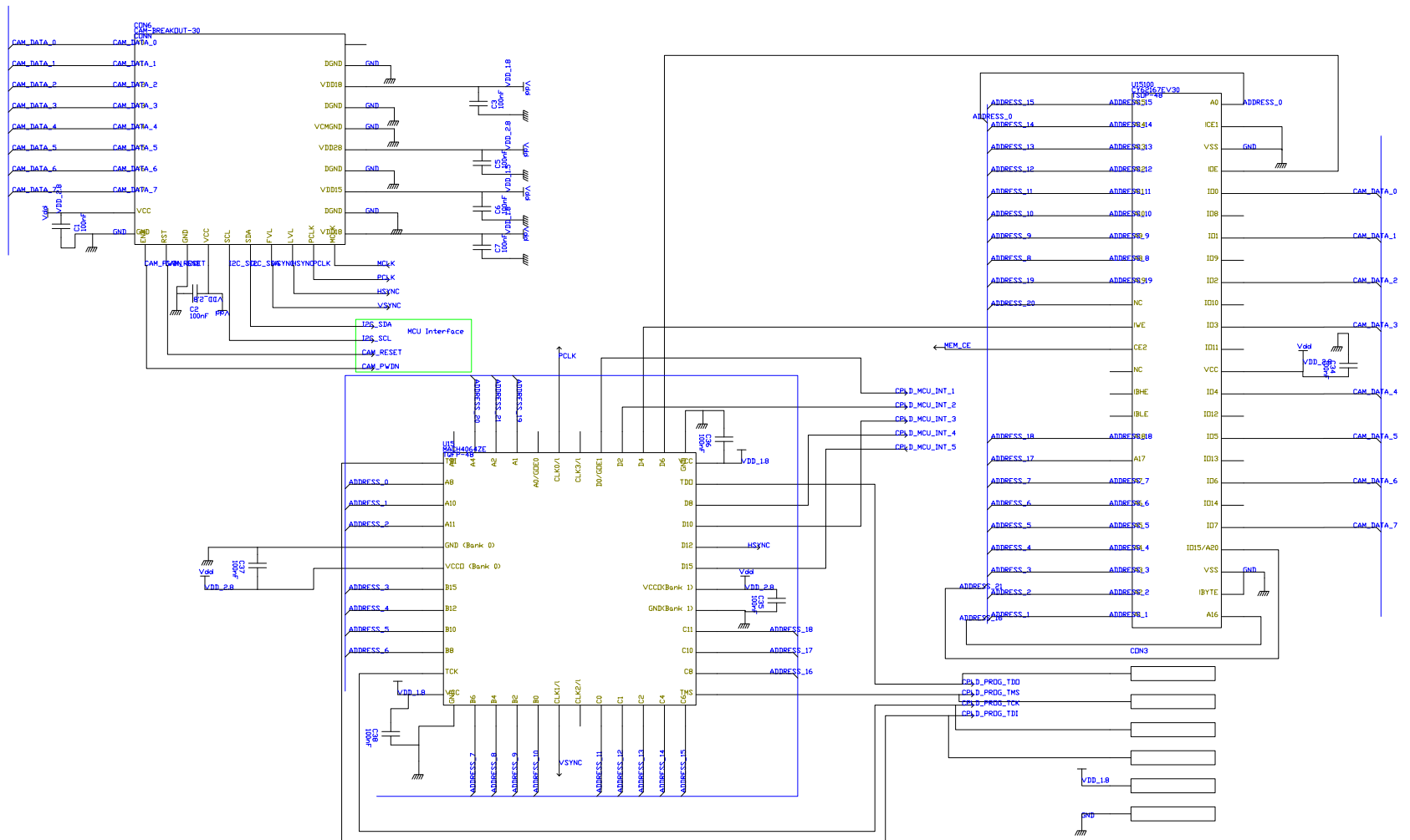


Figure B.1: Low Power Camera Interface Camera, CPLD and Memory

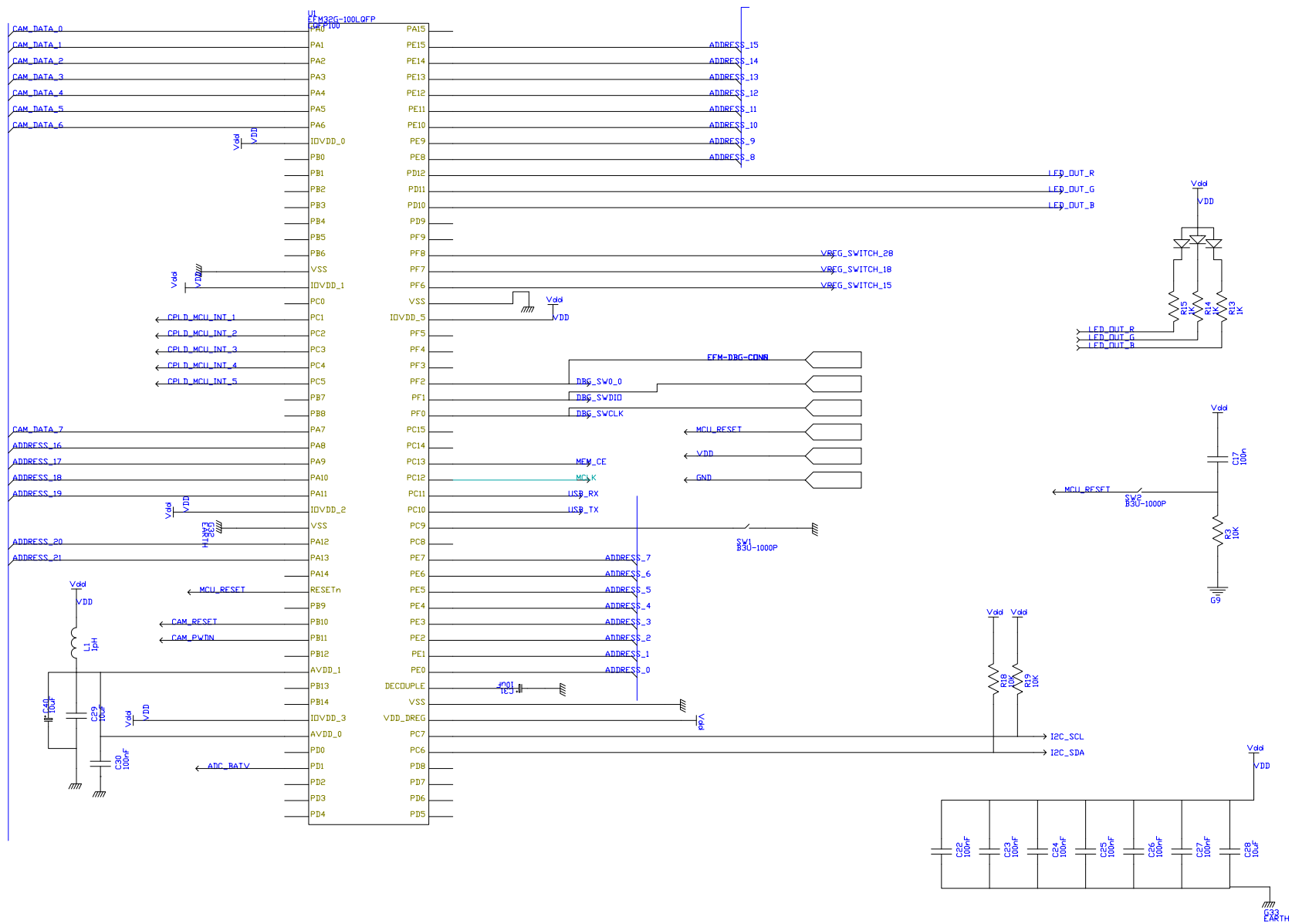


Figure B.2: Low Power Camera Interface Microcontroller Interface

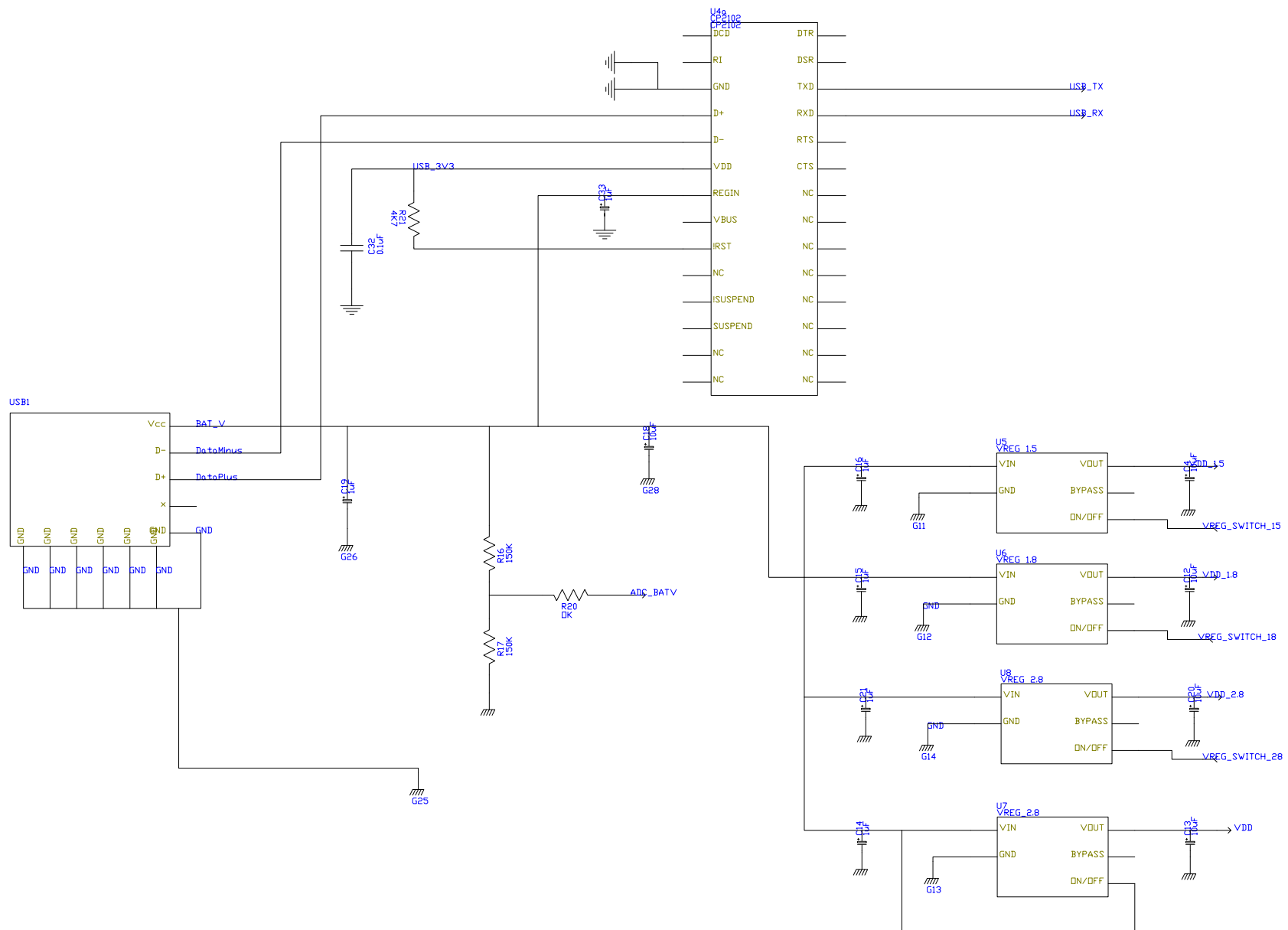


Figure B.3: Low Power Camera Interface USB and Power

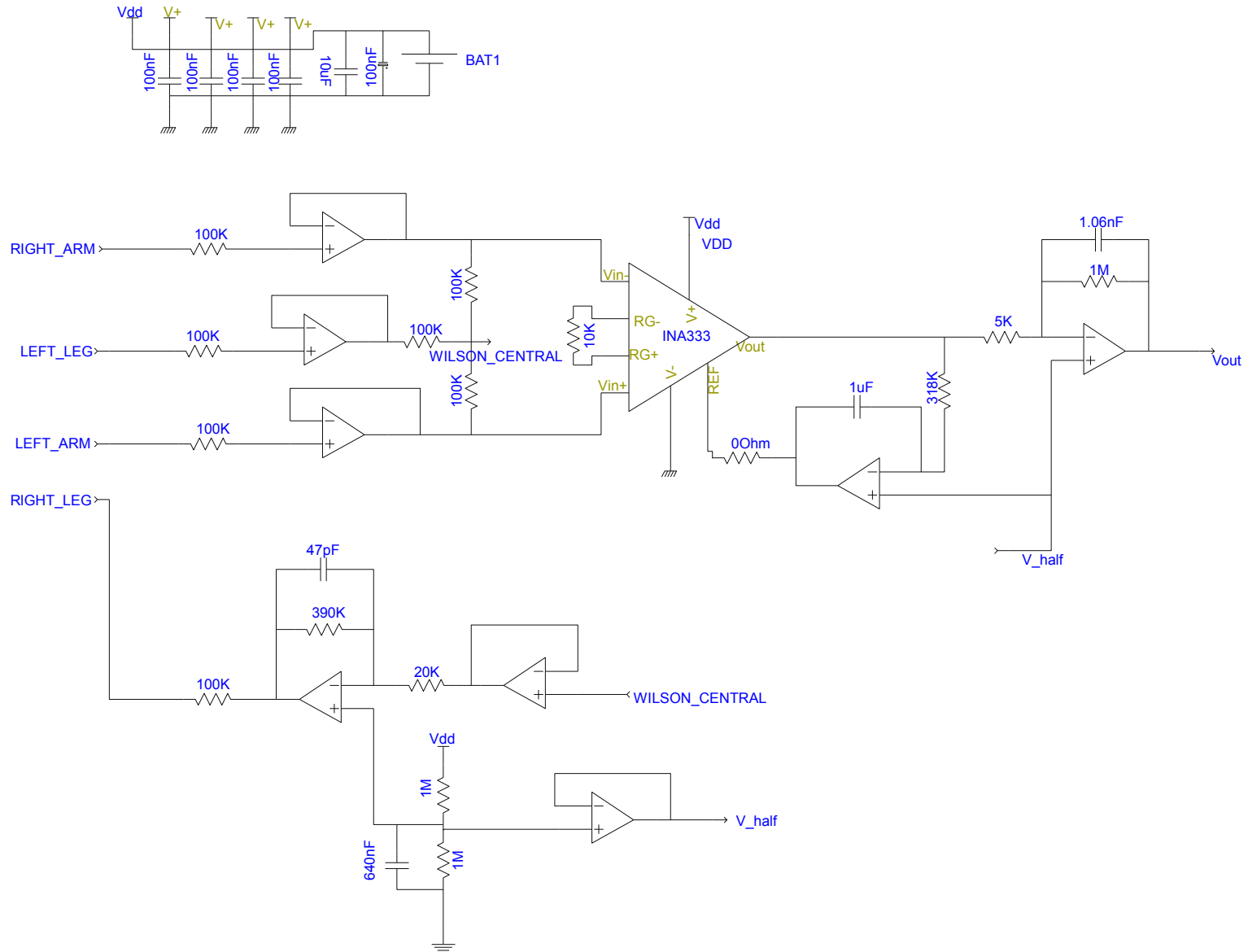


Figure B.4: Cardionode Circuit Diagram

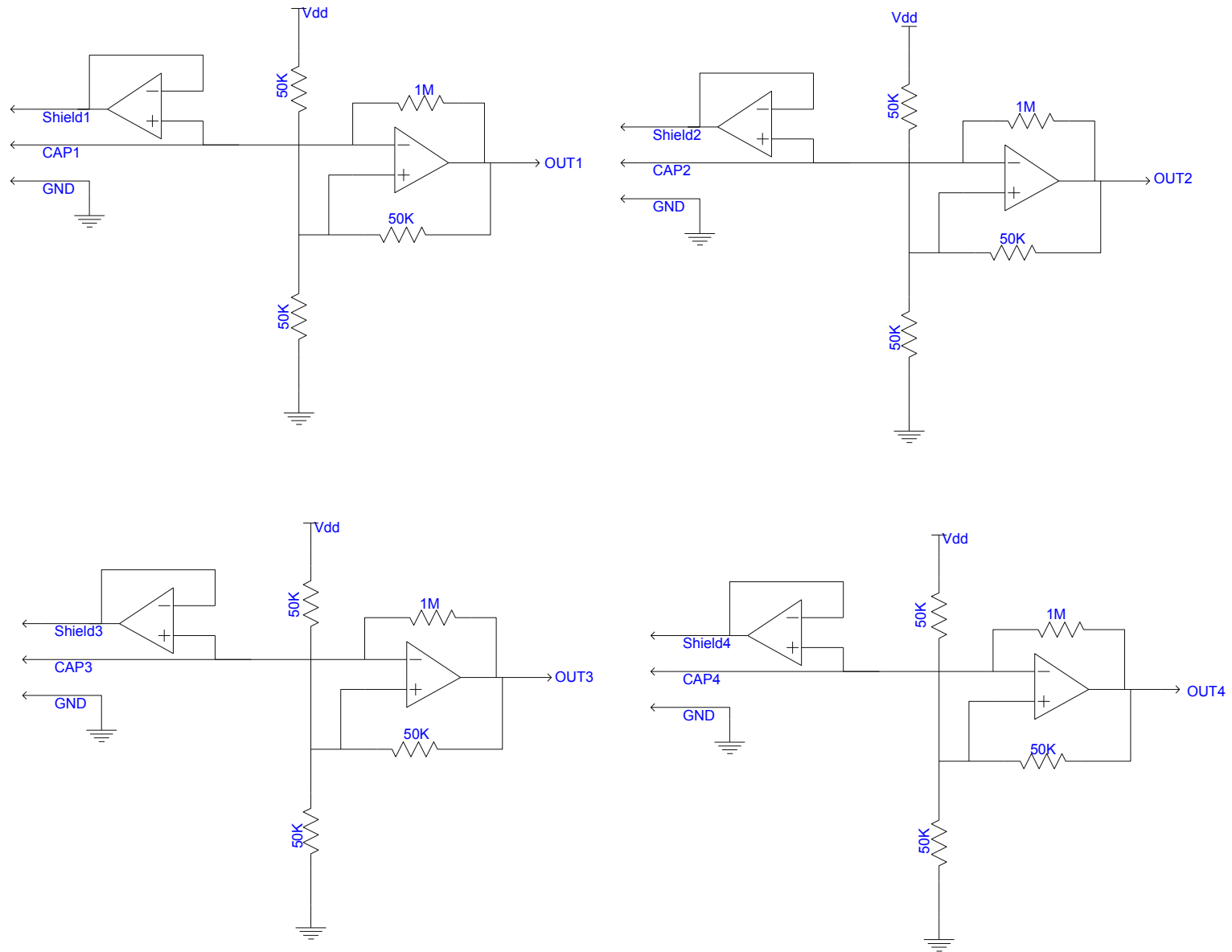


Figure B.5: Respinode Circuit Diagram

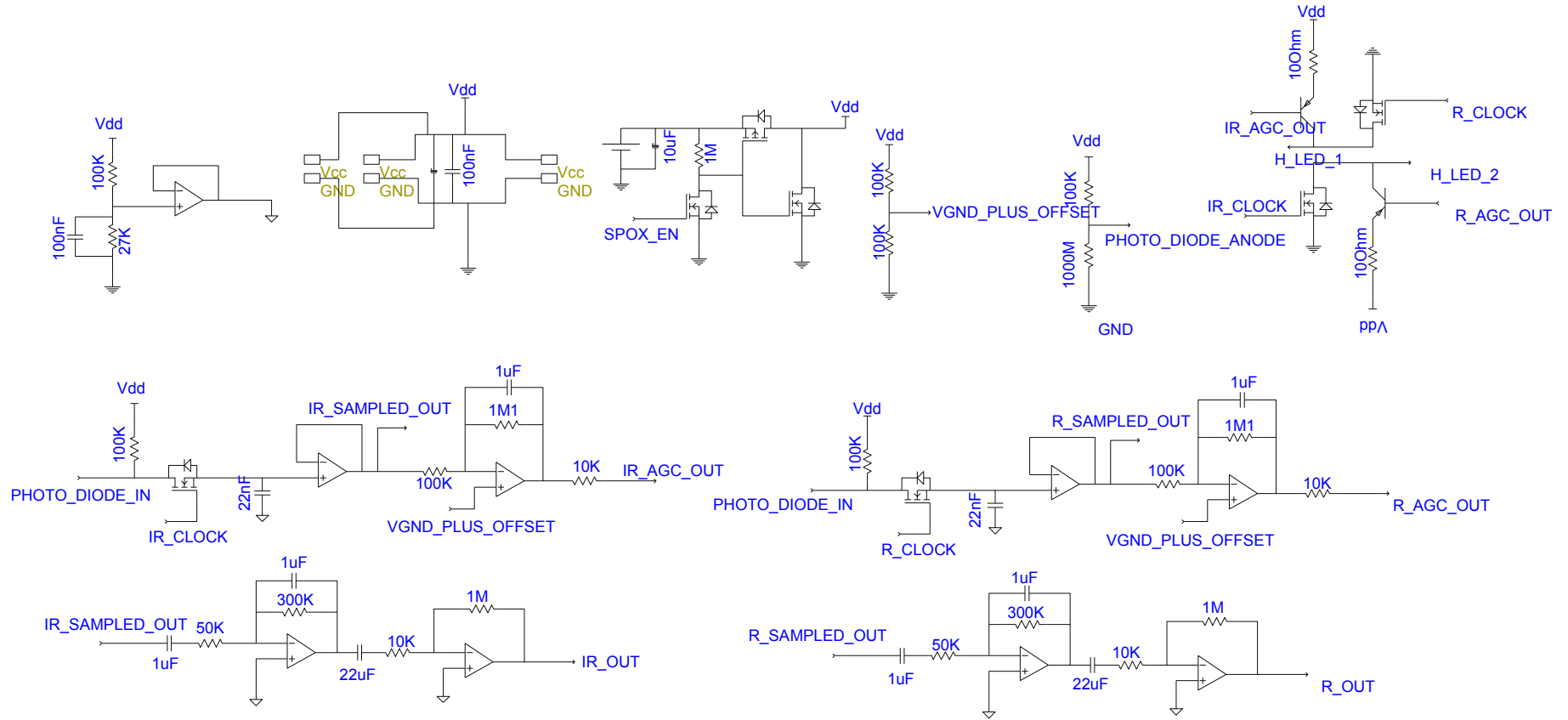


Figure B.6: Spoxnode Circuit Diagram



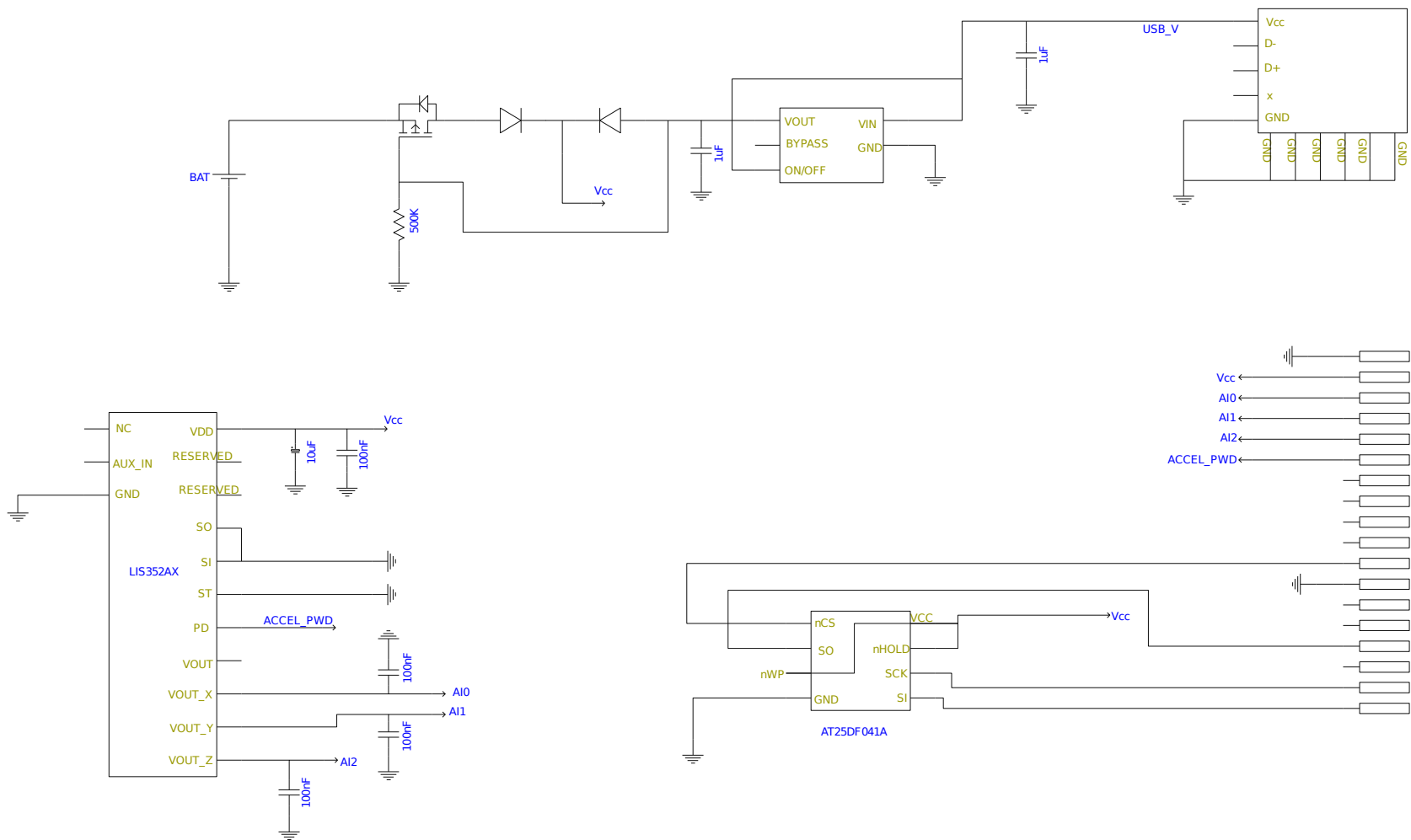


Figure B.7: Accelnode Circuit Diagram



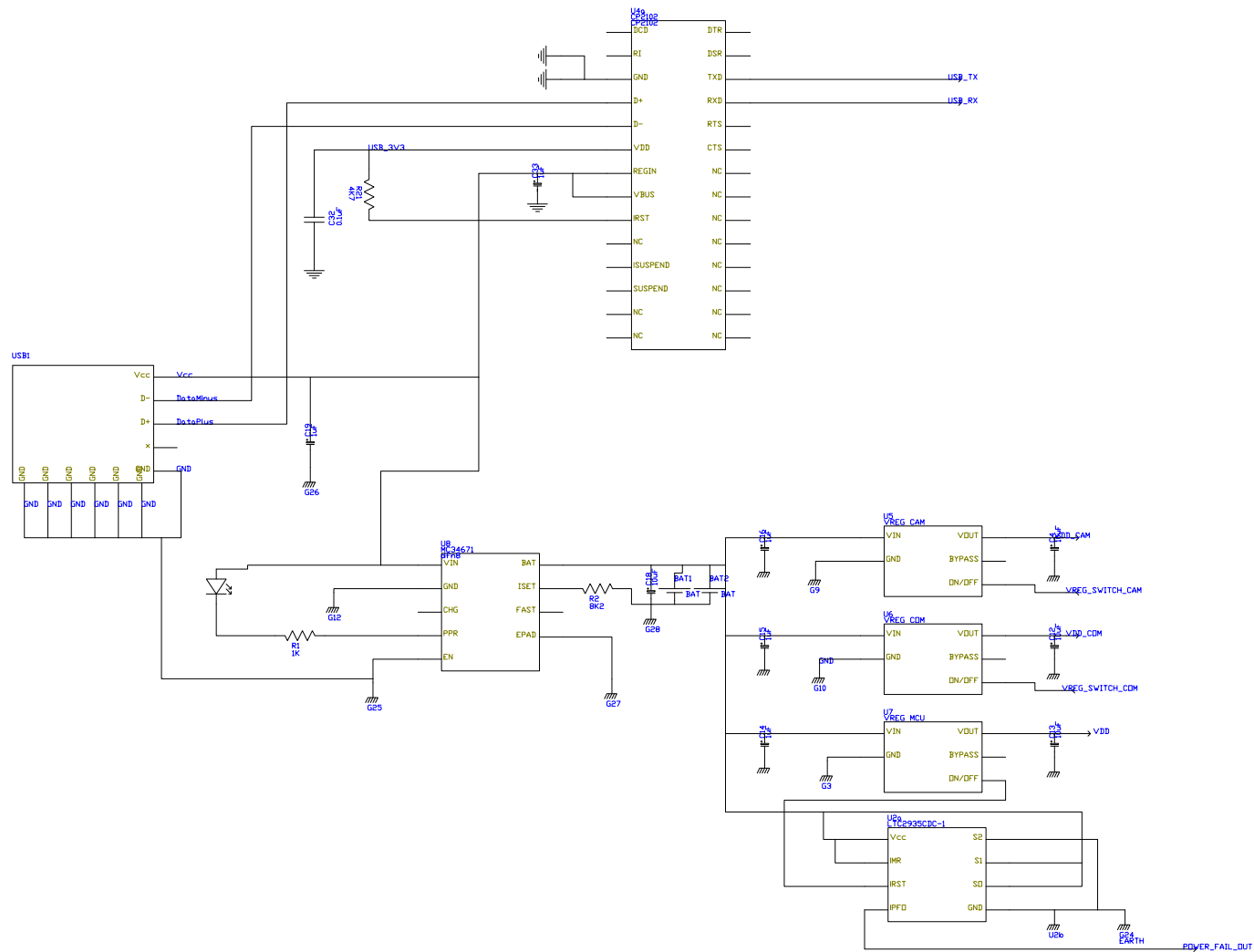


Figure B.9: DejaView Device Power and USB Interface

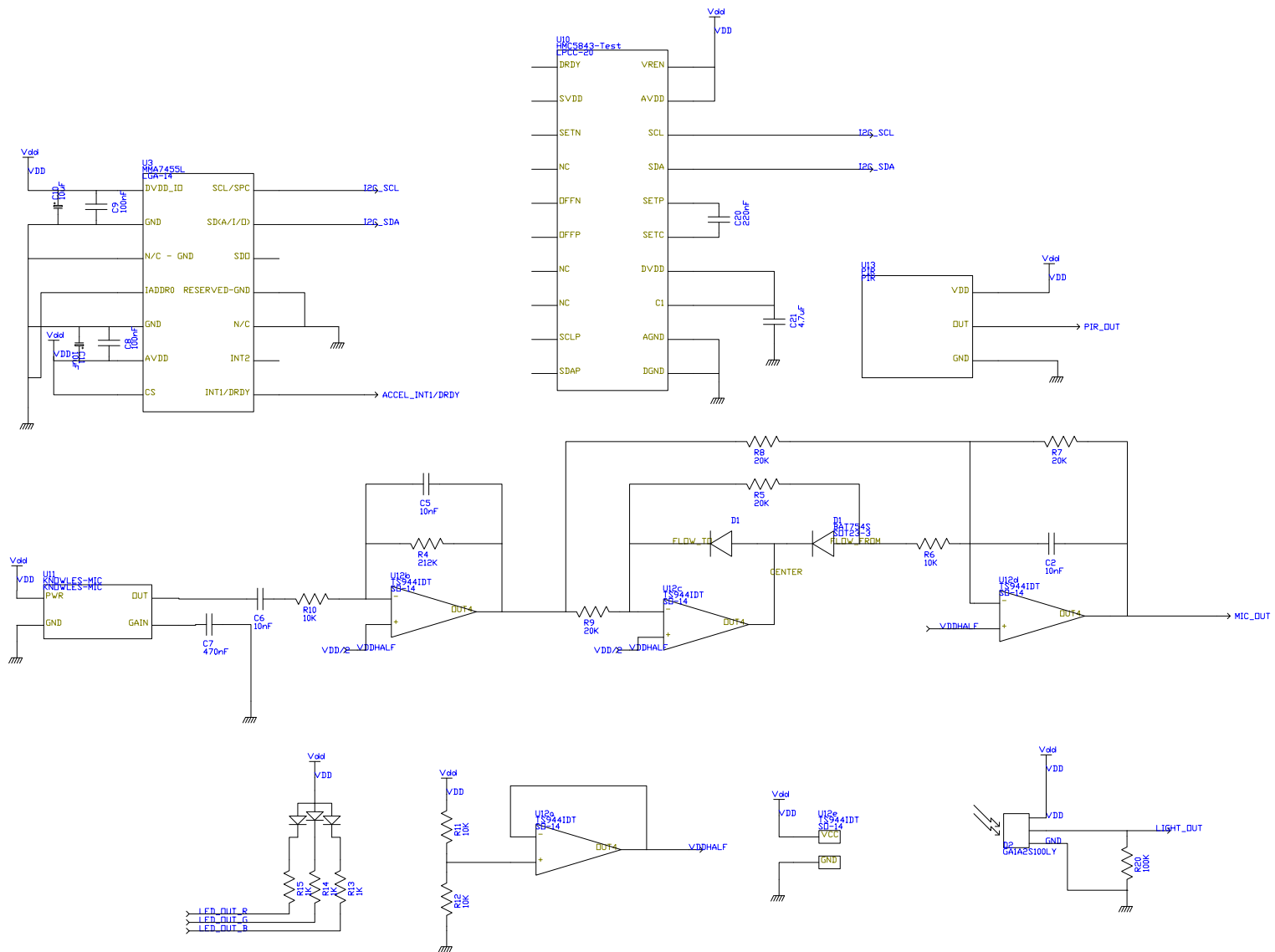


Figure B.10: DejaView Device Sensors

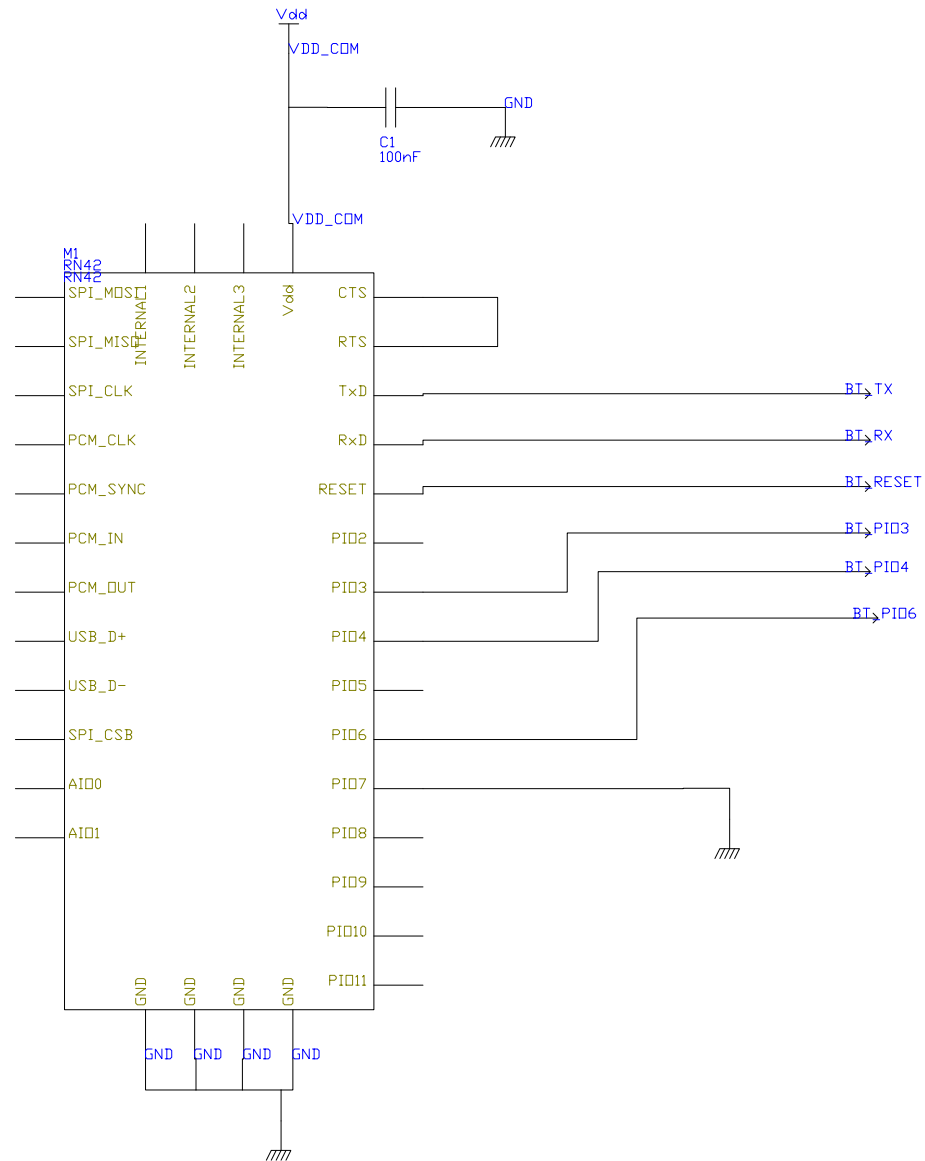


Figure B.11: DejaView Device Communication Connections





## Appendix C

# Printed Circuit Board Layouts

### List of printed circuit board designs

C.1 Accelnode PCB

C.2 BTNode PCB

C.3 Cardionode PCB

C.4 EFM2500T Processor Board PCB

C.5 Respinode PCB

C.6 Spoxnode PCB

C.7 DejaView Device CD2 PCB

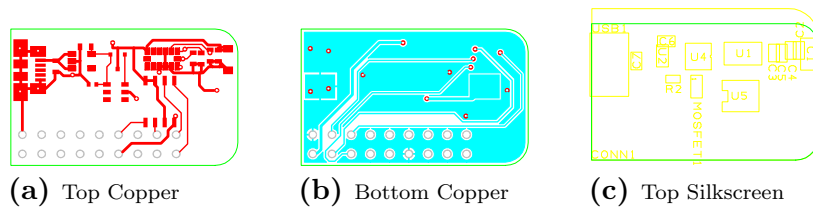


Figure C.1: Accelnode PCB



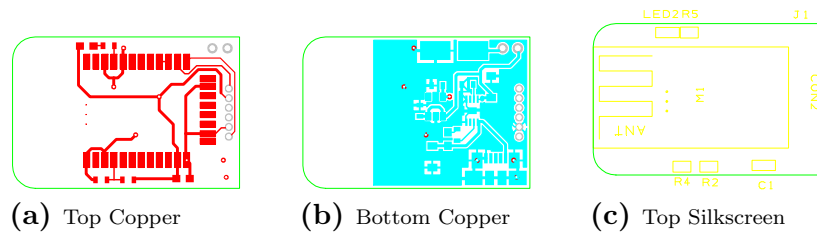


Figure C.2: BTNode PCB

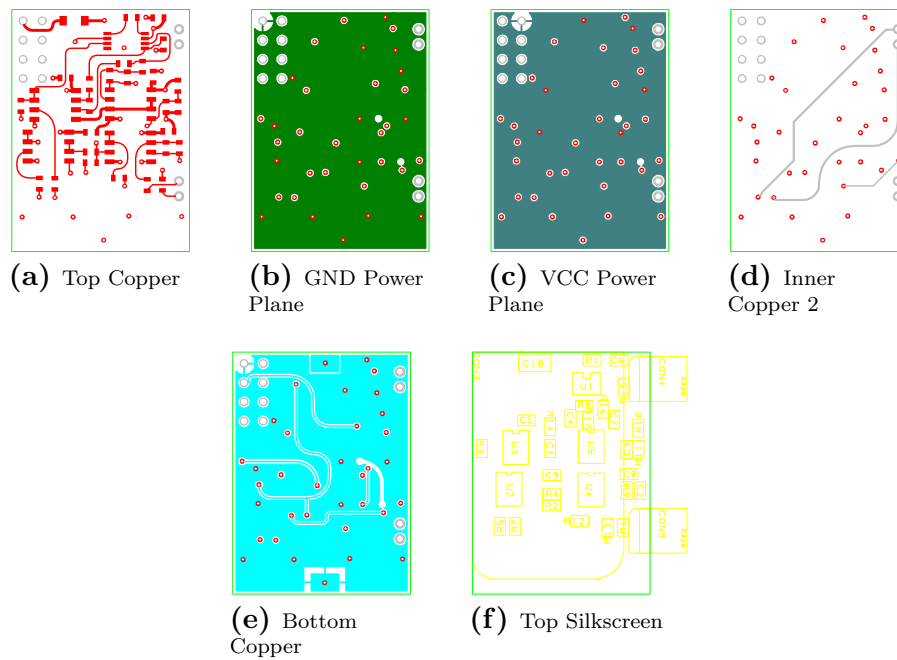


Figure C.3: Cardionode PCB

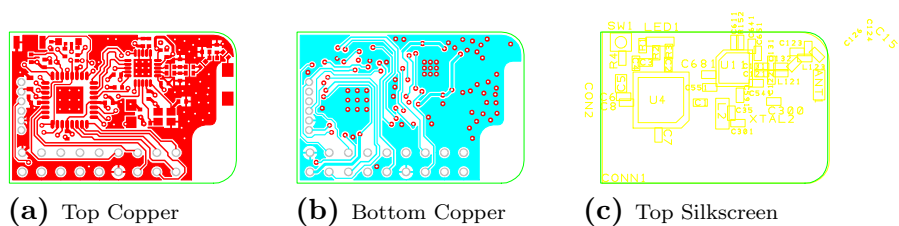


Figure C.4: EFM2500T Processor PCB

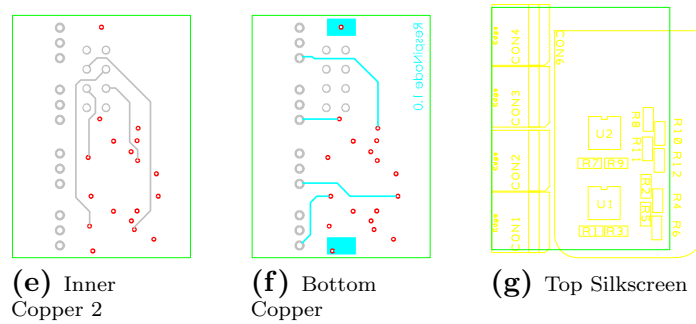


Figure C.5: Respinode PCB

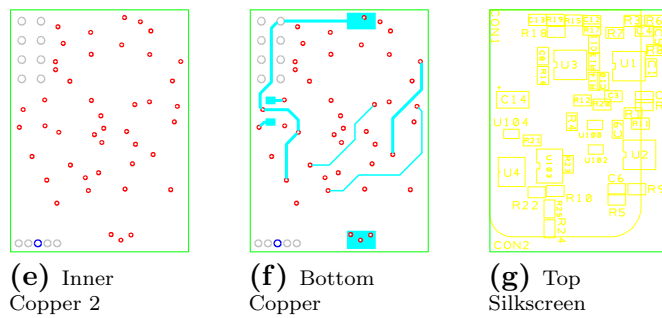


Figure C.6: Spoxnode PCB

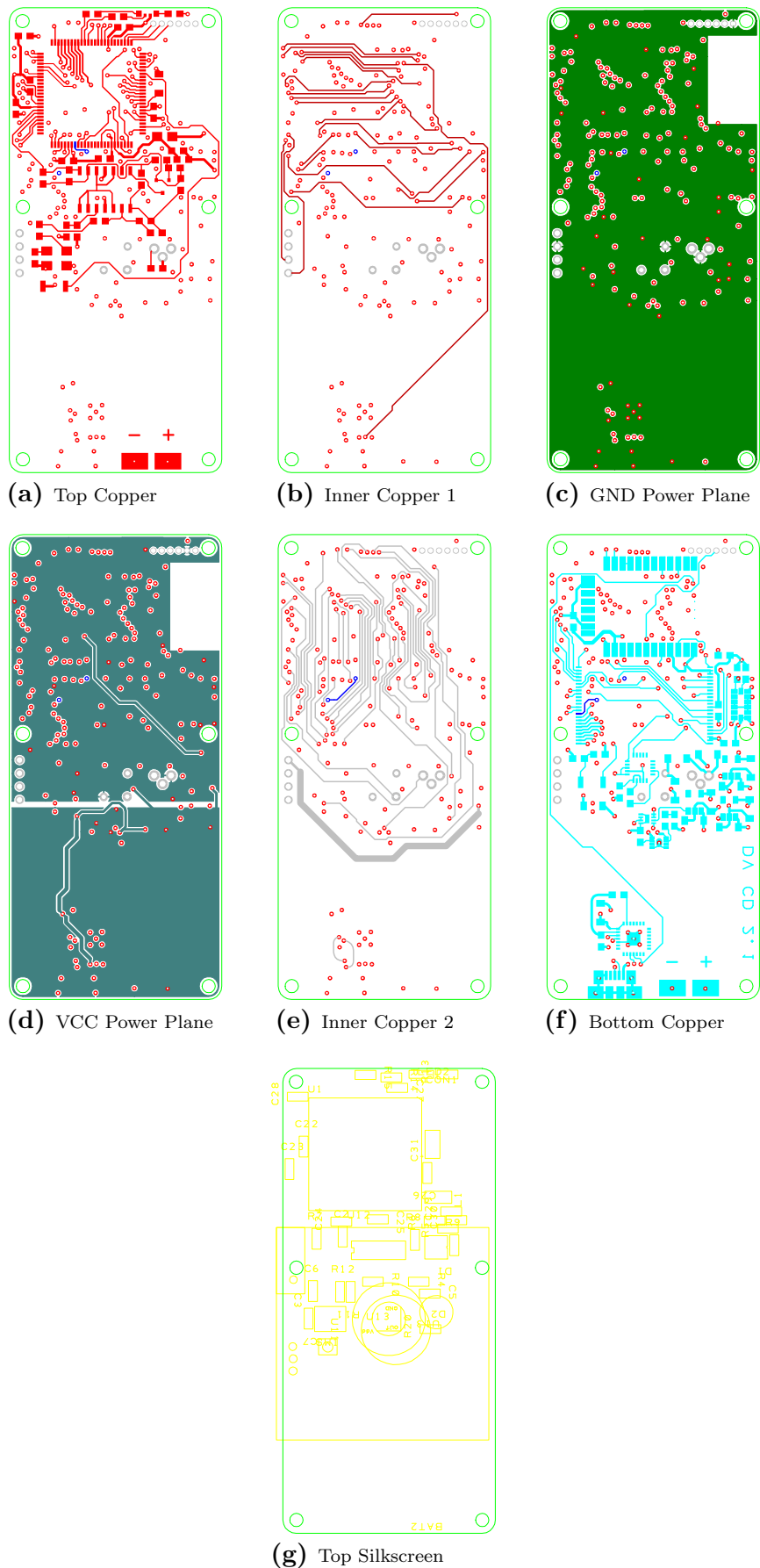


Figure C.7: DejaView Device Cutdown Version 2 (CD2) PCB

# References

- [7] G. Yang, *Body Sensor Networks*. Springer, 2006.
- [8] B. Lo and G. Z. Yang, “Key technical challenges and current implementations of body sensor networks,” *IEEE Proceedings of the 2nd International Workshop on Body Sensor Networks (BSN 2005)*, pp. 1–5, 2005.
- [9] U. Nations, “World economic and social survey 2007: Development in an ageing world,” *New York, Department of Economic and Social Affairs.(2007b), World Population Ageing*, 2007.
- [10] “WHO | preventing chronic diseases: a vital investment.” [Online]. Available: [http://www.who.int/chp/chronic\\_disease\\_report/en/](http://www.who.int/chp/chronic_disease_report/en/) Last accessed February 2008.
- [11] “WHO | global status report on NCDs.” [Online]. Available: [http://www.who.int/chp/ncd\\_global\\_status\\_report/en/index.html](http://www.who.int/chp/ncd_global_status_report/en/index.html) Last accessed April 2012.
- [12] “WHO | The world health report 2006 - working together for health,” WHO, Tech. Rep., 2006. [Online]. Available: <http://www.who.int/whr/2006/en/index.html>
- [13] “Dejaview website.” [Online]. Available: <http://dejaview.ecs.soton.ac.uk/> Last accessed May 2012.
- [14] J. Penders, B. Gyselinckx, R. Vullers, M. De Nil, S. Nimmala, J. van de Molengraft, F. Yazicioglu, T. Torfs, V. Leonov, P. Merken *et al.*, “Human++: from technology to emerging health monitoring concepts,” in *Medical Devices and Biosensors, 2008. ISSS-MDBS 2008. 5th International Summer School and Symposium on*, pp. 94–98, 2008.
- [15] V. Shnayder, B. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, “Sensor networks for medical care,” *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pp. 314–314, 2005.
- [16] A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe, “EyesWeb: toward gesture and affect recognition in interactive dance and music systems,” *Comput. Music J.*, vol. 24, no. 1, pp. 57–69, 2000.

- [17] J. W. Ng, B. P. Lo, O. Wells, M. Sloman, C. Toumazou, N. Peters, A. Darzi, and G. Z. Yang, "Ubiquitous monitoring environment for wearable and implantable sensors (UbiMon)," in *International Conference on Ubiquitous Computing (Ubi-comp)*, 2004.
- [18] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, and J. Stankovic, "An advanced wireless sensor network for health monitoring," *Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2)*, Apr, 2006.
- [19] S. Farshchi, A. Pesterev, P. Nuyujukian, I. Mody, and J. Judy, "Bi-Fi: an embedded Sensor/System architecture for remote biological monitoring," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 11, no. 6, pp. 611–618, 2007.
- [20] J. Lee, W. Jung, I. T. Kang, Y. Kim, and G. Lee, "Design of filter to reject motion artifact of pulse oximetry," *Computer Standards & Interfaces*, vol. 26, no. 3, pp. 241–249, 2004.
- [21] P. Mohseni, K. Najafi, S. Eliades, and X. Wang, "Wireless multichannel biopotential recording using an integrated FM telemetry circuit," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 13, no. 3, pp. 263–271, 2005.
- [22] A. Nieder, "Miniature stereo radio transmitter for simultaneous recording of multiple single-neuron signals from behaving owls," *Journal of Neuroscience Methods*, vol. 101, no. 2, pp. 157–164, Sep. 2000.
- [23] S. Takeuchi and I. Shimoyama, "An RF-telemetry system with shape memory alloy microelectrodes for neural recording of freely moving insects," in *Microtechnologies in Medicine and Biology, 1st Annual International, Conference On.*, pp. 491–496, 2000.
- [24] P. Irazoqui-Pastor, I. Mody, and J. Judy, "Transcutaneous RF-powered neural recording device," in *[Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society] EMBS/BMES Conference, 2002. Proceedings of the Second Joint*, vol. 3, pp. 2105–2106, 2002.
- [25] M. Modarreszadeh and R. Schmidt, "Wireless, 32-channel, EEG and epilepsy monitoring system," in *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, vol. 3, pp. 1157–1160, 1997.
- [26] C. Mundt, K. Montgomery, U. Udoh, V. Barker, G. Thonier, A. Tellier, R. Ricks, R. Darling, Y. Cagle, N. Cabrol, S. Ruoss, J. Swain, J. Hines, and G. Kovacs,

- “A multiparameter wearable physiologic monitoring system for space and terrestrial applications,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 9, no. 3, pp. 382–391, 2005.
- [27] J. Yao, R. Schmitz, and S. Warren, “A wearable point-of-care system for home use that incorporates plug-and-play and wireless standards,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 9, no. 3, pp. 363–371, 2005.
- [28] M. Rasid and B. Woodward, “Bluetooth telemedicine processor for multichannel biomedical signal transmission via mobile cellular networks,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 9, no. 1, pp. 35–43, 2005.
- [29] T. Fulford-Jones, G. Wei, and M. Welsh, “A portable, low-power, wireless two-lead EKG system,” in *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, vol. 1, pp. 2141–2144, 2004.
- [30] I. Obeid, M. A. L. Nicolelis, and P. D. Wolf, “A multichannel telemetry system for single unit neural recordings,” *Journal of Neuroscience Methods*, vol. 133, no. 1-2, pp. 33–38, Feb. 2004.
- [31] “IEEE 802.15 WPAN task group 6 (TG6) body area networks,” Jan. 2009. [Online]. Available: <http://www.ieee802.org/15/pub/TG6.html> Last accessed August 2009.
- [32] B. Gyselinckx, R. Vullers, C. V. Hoof, J. Ryckaert, R. F. Yazicioglu, P. Fiorini, V. Leonov, and E. IMEC, “Human++: Emerging technology for body area networks,” in *Very Large Scale Integration, 2006 IFIP International Conference on*, pp. 175–180, 2006.
- [33] B. Gyselinckx, C. Van Hoof, J. Ryckaert, R. F. Yazicioglu, P. Fiorini, and V. Leonov, “Human++: autonomous wireless sensors for body area networks,” in *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, pp. 13–19, 2005.
- [34] T. Torfs, V. Leonov, and R. Vullers, “Pulse oximeter fully powered by human body heat,” *Sensors and Transducers Journal*, vol. 80, no. 6, pp. 1230–1238, 2007.
- [35] “Bluetooth.” [Online]. Available: <https://www.bluetooth.org/apps/content/> Last accessed August 2009.
- [36] “IEEE 802.15.4 WPAN-LR task group.” [Online]. Available: <http://www.ieee802.org/15/pub/TG4.html> Last accessed August 2009.
- [37] “ZigBee alliance.” [Online]. Available: <http://www.zigbee.org/> Last accessed August 2009.
- [38] “IEEE 802.15.1.” [Online]. Available: <http://www.ieee802.org/15/pub/TG1.html> Last accessed August 2009.

- [39] “NORDIC SEMICONDUCTOR - nRF24LU1+.” [Online]. Available: <http://www.nordicsemi.com/index.cfm?obj=product&act=display&pro=96> Last accessed August 2009.
- [40] “SimpliciTI - RF software protocol.” [Online]. Available: <http://www.ti.com/corp/docs/landing/simpliciTI/index.htm> Last accessed August 2009.
- [41] “SAPHE.” [Online]. Available: <http://ubimon.doc.ic.ac.uk/saphe/m338.html> Last accessed June 2008.
- [42] “SAPHE vision statement.” [Online]. Available: <http://ubimon.doc.ic.ac.uk/saphe/public/SAPHE-vision-statement-pub.pdf> Last accessed June 2008.
- [43] “TRIL centre, technology research for independent living, centre, research, cognitive, social, technology platform, ethnography, falls prevention - TRIL centre.” [Online]. Available: <http://www.trilcentre.org/> Last accessed June 2008.
- [44] K. Lorincz, B. Kuris, S. Ayer, S. Patel, P. Bonato, and M. Welsh, “Wearable wireless sensor network to assess clinical status in patients with neurological disorders,” in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pp. 563–564, 2007.
- [45] “BioMOBIUS - home.” [Online]. Available: <http://biomobius.trilcentre.org/> Last accessed June 2008.
- [46] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, “TinyOS: an operating system for sensor networks,” *Ambient Intelligence*, vol. 35, pp. 115–148, 2005.
- [47] J. G. Jetcheva and D. B. Johnson, “Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks,” *Proceedings of the 2nd ACM International Symposium on Mobile ad hoc Networking & Computing*, pp. 33–44, 2001.
- [48] M. A. Sackner, J. A. Adams, A. De Groote, Y. Verbandt, M. Paiva, and P. Mathys, “Piezoelectric sensor vs. respiratory inductive plethysmograph,” *J Appl Physiol*, vol. 90, no. 1, pp. 403–404, Jan. 2001.
- [49] P. Martinot-Lagarde, R. Sartene, M. Mathieu, and G. Durand, “What does inductance plethysmography really measure?” *J Appl Physiol*, vol. 64, no. 4, pp. 1749–1756, Apr. 1988.
- [50] M. Oehler, V. Ling, K. Melhorn, and M. Schilling, “A multichannel portable ECG system with capacitive sensors,” *Physiological Measurement*, vol. 29, pp. 783–793, Jul. 2008.

- [51] M. Oehler, P. Neumann, M. Becker, G. Curio, and M. Schilling, "Extraction of SSVEP signals of a capacitive EEG helmet for human machine interface," in *30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2008. EMBS 2008*, pp. 4495–4498. IEEE, Aug. 2008.
- [52] Y. Chi and G. Cauwenberghs, "Wireless non-contact EEG/ECG electrodes for body sensor networks," in *2010 International Conference on Body Sensor Networks*, pp. 297–301, 2010.
- [53] A. Ueno, Y. Uchikawa, and M. Noshiro, "A capacitive sensor system for measuring laplacian electromyogram through cloth: A pilot study," in *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007. EMBS 2007*, pp. 5731–5734. IEEE, Aug. 2007.
- [54] J. Vranish, R. McConnell, and S. Mahalingam, "Capaciflector collision avoidance sensors for robots," *Comput. Electr. Eng.(USA)*, vol. 17, no. 3, pp. 173–9, 1991.
- [55] J. Vranish and R. McConnell, "Driven shielding capacitive proximity sensor," United States of America Patent 5 166 679. [Online]. Available: <http://www.freepatentsonline.com/5166679.html> November 24, 1992.
- [56] S. Gabriel, R. W. Lau, and C. Gabriel, "The dielectric properties of biological tissues: II. measurements in the frequency range 10 hz to 20 GHz," *Physics in Medicine and Biology*, vol. 41, no. 11, pp. 2251–2269, Nov. 1996.
- [57] I. Akyildiz, T. Melodia, and K. Chowdury, "Wireless multimedia sensor networks: A survey," *Wireless Communications, IEEE*, vol. 14, no. 6, pp. 32–39, 2007.
- [58] A. Seema and M. Reisslein, "Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a Flexi-WVSNP design," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 462–486, 2011.
- [59] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," *IPSN 07: Proceedings of the 6th International conference*, pp. 360–369, 2007.
- [60] M. Rahimi, R. Baer, O. Iroez, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, pp. 192–204, 2005.
- [61] A. Rowe, A. Goode, D. Goel, and I. Nourbakhsh, "CMUcam3: an open programmable embedded vision sensor," *CMU-RI-TR-07-13, Robotics Institute, Carnegie Mellon University*, 2007. [Online]. Available: [cmucam.org](http://cmucam.org)



- [62] A. Kandhalu, A. Rowe, and R. Rajkumar, "DSPcam: a camera sensor system for surveillance networks," in *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, p. 17, 2009.
- [63] P. Chen, P. Ahammad, C. Boyer, S. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan *et al.*, "CITRIC: a low-bandwidth wireless camera network platform," in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, p. 110, 2008.
- [64] "BT. 601 studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios," Mar. 2011. [Online]. Available: <http://www.itu.int/rec/R-REC-BT.601/en> Last accessed September 2011.
- [65] "LMC6462 dual micropower, rail-to-rail input and output cmos operational amplifier." [Online]. Available: <http://www.ti.com/product/lmc6462> Last accessed May 2012.
- [66] G. V. Merrett, A. S. Weddell, N. R. Harris, N. M. White, and B. M. Al-Hashimi, "The unified framework for sensor networks: A systems approach," Sep. 2006. [Online]. Available: <http://eprints.ecs.soton.ac.uk/12955/> Last accessed September 2010.
- [67] "SWNF: Simple wireless node framework project page." [Online]. Available: <http://swnf.ugforge.ecs.soton.ac.uk/> Last accessed May 2012.
- [68] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.
- [69] C. E. Shannon, "A mathematical theory of communications," *Bell System Technical Journal*, vol. 27, no. 7, pp. 379–423, 1948.
- [70] G. Fabeck and R. Mathar, "Chernoff information-based optimization of sensor networks for distributed detection," in *Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium on*, pp. 606–611, 2009. [Online]. Available: 10.1109/ISSPIT.2009.5407551
- [71] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley and sons, 2006.
- [72] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *Information Theory, IEEE Transactions on*, vol. 19, no. 4, pp. 471–480, 1973.
- [73] P. Wang, C. Li, and J. Zheng, "Distributed data aggregation using clustered slepian-wolf coding in wireless sensor networks," in *Communications, 2007. ICC'07. IEEE International Conference on*, pp. 3616–3622, 2007.

- [74] J. J. Xiao, A. Ribeiro, Z. Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 23, no. 4, pp. 27–41, 2006.
- [75] J. Meng, H. Li, and Z. Han, "Sparse event detection in wireless sensor networks using compressive sensing," in *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pp. 181–185, 2009.
- [76] S. Lee, S. Patten, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega, "Compressed sensing and routing in multi-hop networks," Technical Report, University of Southern California, Tech. Rep., 2009.
- [77] A. Puri and A. Eleftheriadis, "MPEG-4: an object-based multimedia coding standard supporting mobile applications," *Mob. Netw. Appl.*, vol. 3, no. 1, p. 532, Jun. 1998.
- [78] P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings, "A utility-based sensing and communication model for a glacial sensor network," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1353–1360. Hakodate, Japan: ACM, 2006.
- [79] S. Arrabi and J. Lach, "Adaptive lossless compression in wireless body sensor networks," in *Proceedings of the Fourth International Conference on Body Area Networks*, ser. BodyNets '09, pp. 19:1–19:8. Los Angeles, California: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.
- [80] T. Schoellhammer, E. Osterweil, B. Greenstein, M. Wimbrow, and D. Estrin, "Lightweight temporal compression of microclimate datasets," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pp. 516–524, 2004.
- [81] S. Patten, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, ser. IPSN '04, pp. 28–35. Berkeley, California, USA: ACM, 2004.
- [82] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, pp. 1713–1723, 2003.
- [83] T. Banerjee, K. R. Chowdhury, and D. P. Agrawal, "Using polynomial regression for data representation in wireless sensor networks," *International Journal of Communication Systems*, vol. 20, no. 7, pp. 829–856, 2007.

- [84] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, ser. VLDB '04, vol. 30, pp. 588–599. Toronto, Canada: VLDB Endowment, 2004.
- [85] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 491–502, 2003.
- [86] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, 2005.
- [87] G. Merrett, B. M. Al-Hashimi, N. M. White, and N. R. Harris, "Information managed wireless sensor networks with energy aware nodes," 2005. [Online]. Available: <http://eprints.ecs.soton.ac.uk/10546/> Last accessed September 2010.
- [88] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International Journal of Image Processing (IJIP)*, vol. 3, no. 1, p. 1, 2009.
- [89] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *Wireless Communications, IEEE Transactions on*, vol. 1, no. 4, pp. 660–670, 2002.
- [90] —, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, vol. 2, p. 10, 2000.
- [91] E. B. Mazomenos, J. S. Reeve, and N. M. White, "A Range-Only tracking algorithm for wireless sensor networks," pp. 775–780. IEEE, May 2009. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=5136743](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5136743)
- [92] C. Alippi and G. Vanini, "Wireless sensor networks and radio localization: a metrological analysis of the MICA2 received signal strength indicator," pp. 579–580. IEEE, Nov. 2004.
- [93] J. A. Gutierrez, E. H. Callaway, and R. Barrett, *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15. 4*. IEEE Standards Association, 2004.
- [94] T. Instruments, "CC2420 datasheet," 2004. [Online]. Available: <http://focus.ti.com/docs/prod/folders/print/cc2420.html> Last accessed May 2012.
- [95] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp. 221–235, 2001.

- [96] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," p. 95. ACM Press, Nov. 2004.
- [97] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wireless Networks*, vol. 12, no. 1, pp. 63–78, 2006.
- [98] M. Hayajneh, I. Khalil, and Y. Gadallah, "An OFDMA-based MAC protocol for under water acoustic wireless sensor networks," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, pp. 810–814, 2009.
- [99] H. Li and P. D. Mitchell, "Full interference model in wireless sensor network simulation," in *Wireless Communication Systems, 2009. ISWCS 2009. 6th International Symposium on*, pp. 647–651, 2009.
- [100] K. Lorincz, B. Chen, J. Waterman, G. Werner-Allen, and M. Welsh, "Resource aware programming in the pixie os," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pp. 211–224, 2008.
- [101] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 12, no. 11, pp. 1122–1133, Nov. 2001.
- [102] S. Lee, B. Bhattacharjee, and S. Banerjee, "Efficient geographic routing in multihop wireless networks," ser. MobiHoc '05, pp. 230–241. New York, NY, USA: ACM, 2005. [Online]. Available: <http://doi.acm.org/10.1145/1062689.1062720>
- [103] Y. Wang, X. Li, W. Song, M. Huang, and T. Dahlberg, "Energy-Efficient localized routing in random multihop wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 8, pp. 1249–1257, Aug. 2011.
- [104] Z. Sun, R. Yu, and S. Mei, "A robust power-aware routing algorithm for wireless sensor networks," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–7, Oct. 2006.
- [105] "Intel lab data," 2004. [Online]. Available: <http://db.csail.mit.edu/labdata/labdata.html> Last accessed May 2011.
- [106] R. Albert, H. Jeong, and A. L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.
- [107] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," p. 134. ACM Press, 2004. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1031512>

- [108] G. Zhou, T. He, J. Stankovic, and T. Abdelzaher, "RID: radio interference detection in wireless sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, pp. 891 – 901 vol. 2, Mar. 2005.
- [109] "MICA2DOT wireless microsensor mote." [Online]. Available: <https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf> Last accessed May 2012.
- [110] G. Moore *et al.*, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.
- [111] G. Frantz, "Digital signal processor trends," *IEEE Micro*, vol. 20, no. 6, pp. 52–59, Dec. 2000.
- [112] B. Biswas and H. Singh, "TinyDB2: porting a query driven data extraction system to TinyOS2. x," *Trends in Network and Communications*, pp. 298–306, 2011.
- [113] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462. IEEE, Nov. 2004.
- [114] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The LiteOS operating system: Towards unix-like abstractions for wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, pp. 233–244, 2008.
- [115] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.
- [116] "CLOC – count lines of code." [Online]. Available: <http://cloc.sourceforge.net> Last accessed September 2010.
- [117] "Codesourcery lite edition for ARM EABI." [Online]. Available: <https://sourcery.mentor.com/GNUToolchain/release1294> Last accessed September 2010.
- [118] "Minimalist GNU for windows." [Online]. Available: <http://www.mingw.org/> Last accessed September 2010.
- [119] C. Stenhouse, S. Coates, M. Tivey, P. Allsop, and T. Parker, "Prospective evaluation of a modified early warning score to aid earlier detection of patients developing critical illness on a general surgical ward," *British Journal of Anaesthesia*, vol. 84, no. 5, p. 663, May 2000.
- [120] "INA333 Micro-Power (50A), Zero-Drift, Rail-to-Rail out instrumentation amplifier," Oct. 2008. [Online]. Available: <http://www.ti.com/lit/ds/symlink/ina333.pdf> Last accessed August 2011.

- [121] J. W. Severinghaus and Y. Honda, “History of blood gas analysis. VII. pulse oximetry,” *Journal of Clinical Monitoring and Computing*, vol. 3, no. 2, pp. 135–138, 1987.
- [122] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System architecture directions for networked sensors,” *SIGPLAN Not.*, vol. 35, no. 11, pp. 93–104, 2000.
- [123] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood, “SenseCam: A Retrospective Memory Aid,” in *UbiComp 2006: Ubiquitous Computing*, 2006, pp. 177–193.
- [124] M. L. Lee and A. K. Dey, “Providing good memory cues for people with episodic memory impairment,” in *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, pp. 131–138. Tempe, Arizona, USA: ACM, 2007.
- [125] H. of Commons Health Committee, “The electronic patient Record, HC 422-I, sixth report of session 2006-07 volume i: Report, together with formal minutes,” Sep. 2007. [Online]. Available: <http://www.publications.parliament.uk/pa/cm200607/cmselect/cmhealth/422/422.pdf> Last accessed May 2012.