# Sensor Networks and Grid Middleware for Laboratory Monitoring

Jamie M, Robinson; Jeremy G, Frey;
School of Chemistry
University of Southampton, SO17 1BJ United Kingdom
j.m.robinson@soton.ac.uk

Andy J, Stanford-Clark; Andrew D, Reynolds; Bharat V, Bedi;
IBM UK Laboratories
Hursley Park, SO21 2JN, United Kingdom

## Abstract

*By combining automatic environment sensing and experimental data collection with broker based messaging middleware, a system has been produced for the real-time monitoring of experiments whilst away from the lab.*

*Changes in the laboratory environment are encapsulated as simple messages, which are published using an MQTT compliant broker. Clients subscribe to the MQTT stream, and perform a data transform on the messages; this may be to produce a user display or to change the format of the message for republishing.*

*For example, an MQTT client written for the Java MIDP platform can be run on a smart-phone with a GPRS Internet connection, freeing us from the constraints of the network. We present an overview of the technologies used, and how these are helping chemists make the best use of their time.*

## 1. The Chemistry Experimental Problem.

Improvements in automation technology have made it increasingly common to leave chemistry experiments running unattended. In many cases this can lead to a safer working environment (e.g. experiments involving ionising radiation or laser sources) and better results (e.g. liquid surface experiments can be sensitive to vibration). In this extreme case this can be considered to be a "Dark Laboratory".

However, by being present during an experiment an experienced chemist will notice problems as they occur and either alleviate them, or abort the experiment early and restart it, having taken measures to avoid the problem recurring. By providing the chemist with the ability to monitor their experiment and its environment remotely, the safety and quality of result improvements of allowing an experiment to run unattended can still be obtained, while maintaining the user interaction.

Using a combination of off the shelf electronic components, and a software solution from IBM UK laboratories at Hursley[1], a system has been created that allows for the real-time monitoring of the chemistry laboratory from a variety of clients. By using standards-compliant middleware, in the form of an MQTT[2] data broker, the system can be easily, and rapidly, extended to include more sensors and different output devices.

In the current system we present the laboratory information feed using a Java MIDP[3] dashboard interface running on a smart-phone. This interface represents the information streams from the lab using graphical icons, with additional detailed notes available. We also have a representation of the data as a webpage, that can be viewed on desktop PC.

## 2. Background

### 2.1. *e*Science : Science on the Grid

"e-Science is about global collaboration in key areas of science, and the next generation of infrastructure that will enable it."[4]

"[The Grid] intends to make access to computing power, scientific data repositories and experimental facilities as easy as the Web makes access to information." *Tony Blair, 2002*[4]

Broadly speaking eScience is the use of computers to enhance science. The UK eScience project spans a vast range of topics including Mobile Medical Monitoring (part of EQUATOR)[5], Biomolecular Simulation (BiosimGRID)[6], Distributed Aircraft Maintenance (DAME)[7], Collaborative Knowledge (CoAKTing)[8] and providing support methodologies such as storage and resource management in Bioinformatics (MyGRID)[9].

## 2.2. *e*Science in the Chemistry Lab – The Comb*e*chem Project

Comb*e*Chem is promoting consideration of the whole end-to-end pathway for the generation of chemical knowledge, keeping track of the whole process from chemical laboratory to the literature. The project encompasses electronic lab books in the organic synthesis lab through automated crystal structure collection and resolution; Data collection for laser driven surface spectroscopy to ab-initio chemical property calculation, and the use of statistical methods to improve laboratory efficiency.

**Publish@Source.** Frey and Hursthouse[4] note that the current publishing methods, whilst being an important storage and archival mechanism (the technology of paper libraries is well defined), are unable to contend with the speed at which new science is being performed. Taking their example of X-Ray Crystallography, the published data set of structures is currently in the region of 300,000 which is collated in the Cambridge Structures Database[10], however they estimate that approximately 1.5 million structures have been successfully resolved, out of a potential dataset of 10 million compounds. The disparity between the 1.5 million structures known, and the 300,000 published is their major point of concern. They propose that chemists self publish their data in a common format. By providing a traceable provenance for the material right back to its preparation, end users will be able to make judgements about the quality of the data.

**End-to-end data Storage.** By placing the record of the chemical process into an electronic storage facility from its inception, the route is opened for completing the end-to-end system. Taking the example of a chemical synthesis, if at the planning stage the sample is given a unique identifier, this value can be stay with the sample throughout its lifecycle. When a spectroscopic technique is performed on the sample, by using its unique ID the spectra could be automatically filed into the samples record. When a material is taken on as the precursor to another, by using its unique ID a provenance chain is created. By providing all the raw data from the analysis techniques carried out on a compound, especially where there exists a possibility for doubt in its processing, the end user is able to re-process the data, hopefully, achieving the same results.

Alongside the chemical information that is stored in this provenance chain, a record of laboratory conditions is also important, procedures may be sensitive to changes in temperature, light level or vibrations from people in the lab. Aligning these data chains (experimental and environmental) may be problematic. A trivial way to overcome this is to reference against time. If all the servers used are synchronised to a central time server, time-stamps can be used to align datasets which are collected separately.

## 2.3. Middleware

Middleware is the name given to software which provides a messaging fabric to link applications and systems together. The implication from the name is that it is something that occupies the space between the operating system and the applications, and this is pretty much accurate. The alternative to not using a middleware system, is that the application writer has to deal with the mechanics of getting messages from A to B, dealing with connection failures, network outages, duplicate messages, etc, etc.

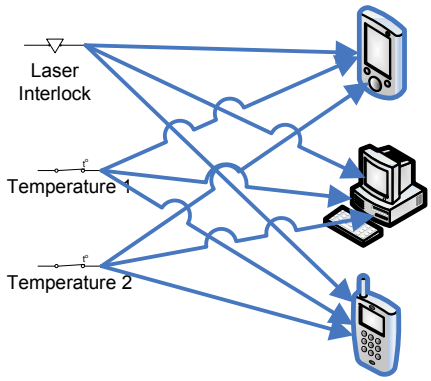


**Figure 1. Client-Server Dataflow without Messaging Middleware. All producers talk directly to the clients, and have to deal with potential networking instabilities**

A middleware system takes that responsibility away from the application writer, and provides a convenient interface to enable the application to send a message, and be confident that it will get to its destination. This enables the application writer to focus on the domain-specific part of the problem (i.e. closer to the user and the data), and not have to worry about moving messages around the system. So with a middleware system, the problem of "by what mechanism will a message be delivered from application A to application B", becomes: "what would you like to send, and what will you do with it when it arrives?"

## 2.4. Publish and Subscribe Technologies[11]

The publish/subscribe model is built around a central broker and a number of clients which connect to the broker. The broker acts as a go-between: an agent that matches subscribers to information with publishers of information thats relevant to them. Clients can be publishers of, and/or

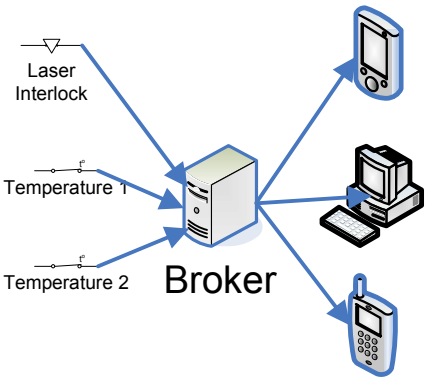


**Figure 2. Dataflow with Messaging Middleware. All data flows through the broker, decoupling data producers from data users.**

subscribers to, data and can range from big enterprise-based servers to hand-held pervasive computing devices or unattended remote telemetry devices.

Publishers send units of information (called messages) to the broker, on a specific topic. The topic is like the subject line of an e-mail. It tells you what the body of the message contains. Subscribers register their interests in certain topics with the broker. The broker manages connections from publishers and subscribers, and deals with authentication and access control lists, to control who is allowed to publish and subscribe to which topics.

It's important to remember that publishers and subscribers are usually unknown to each other, the broker acts as a matchmaker. This decoupling makes the system future-proof by generalising the use of data; it is not tied to specific applications. At any time, you can deploy new applications that can use a new combination of topics. It is this potential that makes publish/subscribe such a powerful concept.

Topics are arranged hierarchically, with slashes (/) between the levels, similar to a URL. The hierarchy, which defines an information space, must be carefully designed to help ensure that data is available in a sensible, logical, structure. For example, if you had two labs with two temperature sensors in each, sensible topics might be;

```
/environment/lab1/temp1
/environment/lab1/temp2
/environment/lab2/temp1
/environment/lab2/temp2
```

When a message from a publisher arrives at the broker, the broker examines the topic, and matches it against the expressed interests of the currently registered subscribers (including wild-card matches). The broker then sends a copy of the message to each subscriber whose subscription matches that of the incoming publication. This is a true "push" model: data is sent from the publisher to the broker, then directly sent by the broker to the subscriber. The subscriber must maintain an open socket connection to the broker in order to receive those pushed messages.

### 2.5. Using middleware in the laboratory space

A powerful feature of using messaging middleware, is that the architecture allows collaborating applications to intercommunicate via a central hub, known as a Message Broker. Each application sends its data to the broker, and the broker sends it on to the intended recipients. This decoupling of producer (or publisher) from consumer (or subscriber) is extremely powerful, as it means that neither the publisher nor the subscriber needs to know about the other party. This means that data producers can be simply set up to publish their data to the broker, and that's all they need to do. On the other side, subscriber applications then tell the broker what kind of information they're interested in, and the broker forwards any messages that come in from publishers, matching those interests, to the interested subscribers. This permits great flexibility in the rapid exploration of new ideas and the easy deployment of new applications which are written as new uses are found for the data that's being published. Similarly, if a piece of equipment is swapped for another type of machine, as long as it publishes the same information as the previous one, none of the subscribing applications need to know that the swap has taken place: they simply continue to receive the data they expect, as before.

In an environment where things are often changing, and new things are being tried out, with the research lab being a case in point, the decoupling of publishers from subscribers has particular benefit as the back end applications which process, display, etc, the information can remain the same while the equipment generating that data may be changed, improved, etc. Another significant benefit is the one-to-many capability of publish/subscribe - several people and applications may receive one piece of data being published from a device in the lab.

The presence of the broker also helps define the responsibility in case of network communication failure. All nodes in the system (both publishers and subscribers) make a TCP connection to the broker as clients, and hence if the communications network fails are responsible for either exiting gracefully, or attempting reconnection. The broker does however provide tools to help minimise the problems this can cause. If a subscriber needs to receive a message as soon as it connects, to fill a data structure for example, then the broker can be configured to use 'retained topics'. When a retained topic is published, the broker stores a copy of it in memory, and then publishes out this old message whenever a new subscriber requests a topic. The broker also manages a system of "Last Will and Testament" this allows a client to

specify, when it connects, a message (and the topic to publish on) that notifies other clients of its failure. This could be used to alert the System administrator of problems, or warn users that the current data may be out of date. All these features mean that smooth running is supported without special code for each sensor output.

### 2.6. MQTT

MQTT (MQ Telemetry Transport)[2] is one of the protocols supported by the IBM WebSphere Message Broker products as a way of getting data in and out of the broker. The protocol was designed specifically for remote telemetry applications, with three specific design goals:

1. It should offer a once-and-once-only assured delivery mode to enable a message to be reliably transferred all the way from a remote sensor to a back-end application.
2. The protocol should be as lightweight as possible across the ”wire”; most remote telemetry is done over low bandwidth, high cost networks, and so minimising the overhead of each message is highly desirable.
3. The protocol should be very easy to implement on embedded devices such as sensors and gateways.

The MQTT protocol has an open, published specification[12], which is available for anyone to implement on a client device, and reference implementations are available from IBM in Java[13] and C[14].

## 3. Implementation

### 3.1. Implementation Overview

The initial implementation of this was in a Laser lab used for measuring surface optical properties. It was decided to monitor temperature, presence of people in the lab (and their movement in and out) and the state of the lab lights (on/off). A sensor connection to the laser interlock was also made, as this provides a quick indication of unauthorised access to the lab (which would result in the interlock changing state), and if the interlock is tripped, laser output is stopped, which makes the experiment data useless.

Data from the sensors in the laboratory are captured into a computer system, which looks for changes in the measurements. When a change is detected, a message is published as a MQTT message over an IBM WebSphere Connection Server Micro Edition (”MicroBroker”) running locally in Southampton. These messages are then distributed to a range of clients, some of these being end user displays, some being storage agents (for example writing data to an SQL database), and some being transform agents to reformat the data for other clients.

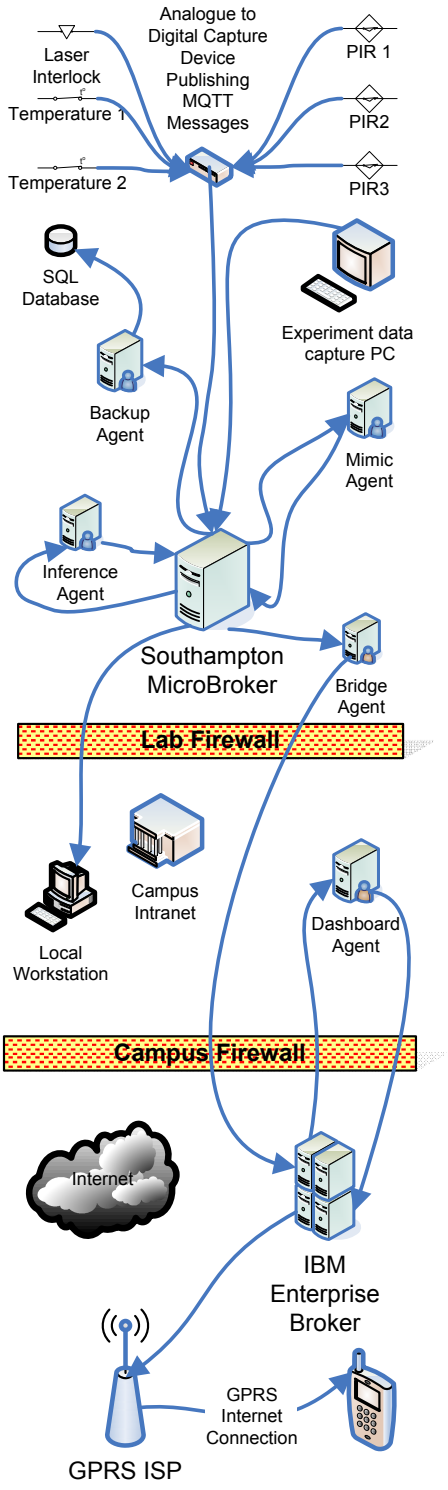


**Figure 3. Dataflows within the Laboratory messaging system.**

### 3.2. Electronics

Temperature was measured using three semiconductor temperature sensors supplied by RS-Components which provide a linear temperature to voltage output. This signal is then captured using an existing Data Acquisition card. Currently these sensors measure temperature in three areas of the lab, away from the rig, near the rig and inside the safety covers alongside the input-side optics.

The state of room lighting (on/off) is being monitored by a photo-diode placed alongside one of the light fittings. The signal from this is monitored by the data acquisition card, and passed through a threshold filter to generate a toggle that mimics the light switch. Currently we're only interested in light being on or off, as the actual level is reasonably constant in the region being studied however for an experiment detecting in the visible light region, it may be more appropriate to monitor the actual light level, with a sensor mounted on the experimental rig.

Monitoring of personnel movements was performed using security alarm components. A PIR sensor in the lab detects people moving, it has been found that due to the layout of the lab, and the necessity of operators to stay still, that the PIR loses people in the lab, and stops triggering. This was predicted when we were at the planning stage, so the ability to check people passing through the lab doors was also added. Door state is monitored using simple magnetically controlled reed switches. All these toggle sensors connect back to the TTL Input of the Data Acquisition Card, using the card's internal 5V supply with a bias resistor to generate the high state.

### 3.3. MQTT messages in the SmartLab

Within the smartlab a system was required that could provide message transmission reliability, the ability to distribute messages to a range of clients, ease of implementation on an range of machines (although mostly PC based), a clear degradation scheme in case of system failure, and the ability to filter the message stream hence allowing minimal data to be sent, enabling the use of expensive communications technologies such as GPRS[15]. MQTT provides these facilities as part of it's specification, and through using existing software such as the MicroBroker, development time has been significantly reduced.

The MQTT specification[12] defines a generic messaging system, providing transport controls, and a container to hold application specific message. In the Southampton Smartlab monitoring system we chose to use an XML like messaging format. This decision was driven by two core reasons.

1. Format Standardisation, the Comb*e*chem project is using XML based formats where-ever sensible, to aid future file conversion, and interpretation.
2. Compatibility with other Projects, The Floodnet project[16] uses an XML based messaging delivered over MQTT, and already has some tools to use this data, that can be adapted for the SmartLab monitoring work.

By using the MQTT libraries provided by IBM[14, 13] we are saved from having to encode the underlying MQTT messages, and need only produce the text of the message, and pass this to the provided library. Our messages follow the XML fragment shown in figure 4.

```
<msg>
  <data>Wiring_Box</data>
  <value>20.996094</value>
  <time>1095690315</time>
</msg>
```

**Figure 4. Example of the XML message that carry sensor information in the SmartLab system.**

### 3.4. Transformation Agents

A number of small "Agents" are used within the SmartLab Messaging system. These perform translation tasks, to make the messages accessible from some other system. These can either be transferring messages away from the MQTT based system, or may be translating message from one format to another, but then encapsulating the result as MQTT message, and publishing them back to the Broker (or another Broker). The agents used in the SmartLab as shown in Figure 3, and discussed in more depth here.

**Mimic Agent.** The workstation client shown in figure 3 includes an applet that plots live data from the broker as it is received. This plotting applet is linked to the temperature data feed. The sensor system in the lab only produces messages when the temperature changes, and the applet only plots a point when it receives messages. A more useful plot is one showing temperature over the most recent time period, to get the applet used to plot this required publishing a message to it every second. To achieve this the mimic agent was written, which subscribes to the temperature topics from the lab, then publishes messages once a second on the temperature mimic topic.

**Bridge Agent.** The Southampton MicroBroker operates within the University of Southampton network domain, and access to it is restricted by the University's data access policy (enforced by the campus firewall). The mobile phone client, is a remote client on the Internet, using a commercial ISP (in this case a mobile GPRS provider) and as such is outside the campus administrative domain. Hence for the mobile phone client to receive data from the lab it needs to be published by a publicly visible broker. The simplest solution to this would be to make the Southampton broker publicly visible (by liaising with the campus firewall team) However this would make all of our data streams publicly visible, and could potentially allow other users on the Internet to inject data into the streams. The solution to these problems is to bridge the data from our MicroBroker onto a publicly visible broker, in this case an IBM WebSphere Business Integration Message Broker ("Enterprise Broker") run by IBM. The bridge agent, which is a component of the IBM MicroBroker, allows us to only send that data which we want to be publicly visible, and is a one-way connection, so external clients can't inject data into our private data streams.

**Backup Agent.** Data on the broker is inherently transient in nature (or at best retaining the last message). One aim of the lab monitoring work is to be able to review old data, so that we can consider the lab conditions when analysing data (and possibly justifying poor data), therefore we need a way of retrieving old data. The backup agent performs the "store" function of this recall, by subscribing to all the lab data streams, and writing the data from the messages into a SQL database.

**Dashboard Agent.** The phone client requires messages suitable for client display, and is less concerned with the raw message format. Generating these messages requires a simple transform, mapping messages from the lab topics onto dashboard topics, and also extracting the data from the lab topics, and writing it out as dashboard display texts. The dashboard agent also performs some basic inferencing. An example of this is that by storing previous values, an indication of direction of change can be added to the messages eg "The Temperature rose by $1^{o}$C to $23^{o}$C".

**Inference Agent.** By combining the existing data streams, and performing some action on the data, new "inferred" streams can be generated. At this time a simple example of this is the "lab occupied" topic. To generate this, an agent listens for messages from the three PIR sensors in the lab, and records the time they last triggered. Then when the door *closes* checks to see if there was a PIR trigger in the past 5 seconds. If there wasn't then the door closing must be someone entering the lab, so we send a message stating lab occupied. If the PIRs had triggered in the 5 seconds preceding the door closing, then it could be someone leaving, or someone else entering. However if someone new entered, then the PIRs will continue to trigger after the door closed. Hence we can send a "Lab Unoccupied" message, but when we get a PIR trigger message, if the current state is "unoccupied", then this must be wrong so send a "Lab Occupied" to correct it. The time between the door closing, and the next PIR firing will be quite short, so the blip is unlikely to be noticed.

### 3.5. Security and Firewalls in the Smartlab

Security is always a concern, especially where sensitive or important data is involved, when the data goes across the public Internet, and where control messages are being sent to modify experimental parameters or turn devices on or off.

Figure 3 shows the flow of data within the Broker in the Smartlab. There is some system re-use within the Smart-Lab, ie the majority of the services are hosted by the same physical machine (MQTT Broker, Bridge Agent, Mimic Agent, Backup Agent, SQL database and Inference Agent), however as all MQTT communication is done using TCP/IP sockets, they could be separated. This would be particularly useful if other laboratories were linked into this system, and the load was found to be too great for the one machine.

After concerns about system reliability and security, a stateful packet filtering firewall[17] was implemented using a dedicated machine, and the Linux netfilter[18] software. This was designed to protect the Microsoft Windows based laboratory data capture machines after an outbreak of the Sasser Worm[19]. The speed with with this worm spread resulted in the experiment data capture machine becoming infected before the appropriate security patches could be installed. Similar local (software) firewalls have also been implemented on all public facing Linux machines in the Smart(SHG)Lab. The Lab is also protected from the Internet at large by a Campus Level firewall, however as discussed above this causes problems for remote access, resulting in the need to use the bridge agent and a publicly visible broker.

MQTT deliberately has a very minimalist approach to security, enabling appropriate security to be layered on top of it as required for any given application. Encrypting the message payload is an obvious first step in securing the data, which can be done using PKI certificates if required, to additionally provide signing for authentication and non-repudiation. Challenge/response security can be incorporated at the application level, by sending the challenge/response flows as MQTT messages over pub/sub. As MQTT is a protocol on top of TCP/IP, standard VPN (Virtual Private Network) products can be used to secure the connection, and hence the data flowing inside it.

## 4. Discussion

### 4.1. Effect on the chemist

The current implementation was designed as an exemplar of the technology, to be implemented quickly, with the possibility of providing the chemist with some added value. The data provided by the current sensors is of little use in real-time. However as a recallable data-set the temperature, and room access information is valuable for corroborating poor quality experimental data.

An exception to note, is that whilst at a conference in Paris discussing associated work, it was noticed that the temperature in the lab was somewhat higher than usual. This was reported back to the people working in the lab by email, who then discovered that the Air-Conditioning wasn't performing efficiently, and an engineer was called to rectify the problem.

At present we don't transmit the actual experiment data, this decision was reached for two reasons. Firstly the data produced is reasonably large (and the aim is to keep MQTT messages small to minimise transmission costs), and more importantly, the display capabilities of smaller clients (such as the phone) makes a graphical display of the data of little use. More useful will be to send a message stating that the experiment has finished, and giving a unique identifier. This could then be entered into a web page, to display the data on a device with better display capabilities, such as a PC .

By storing all the experimental, and environmental data surrounding an experiment, we can build this into a virtual laboratory simulator. This could then playback the events of the laboratory, which would be useful during analysis to help answer the "why did it do that" question. It could also be used as a teaching tool.

### 4.2. Safety Concerns

Remote monitoring and control technology such as MQTT and the message broker can be used to implement "lights-out" operations of labs, factories, oil wells, etc, with no need for anyone to be physically present at the location being monitored or controlled. This being particularly problematic when dangerous equipment is being remotely controlled. The use of local mechanical and electrical interlocks can help to aid this, but adding remote monitoring the operator is placed in a more knowledgeable position. Security devices such as PIR sensors and door/pressure switches can be used to raise an alert if anyone unexpectedly enters a room or building. Identification technology such as RFID[20] tagging can also be used to verify the identity of personnel who do enter or leave a controlled area, again using MQTT to alert the appropriate parties.

### 4.3. Conclusions

The use of a message broker based approach gave a significant head-start in the implementation of the laboratory monitoring solution. The MQTT message broker provides message transmission reliability, the ability to distribute messages to a range of clients, and the ability to filter the message stream based on client requests. The availability of standard libraries eases the implementation of MQTT clients. By automatically collecting and distributing data, additional metadata can be provided to the experimental report that would otherwise have been missed.

## 5. Future Work

### 5.1. Additional Data Streams

The current data streams were chosen for their speed of implementation. Based on knowledge gained here, more chemically useful data streams need to be implemented. This will require the addition of extra sensors, for example, indication of laser running, and laser emitting. It will be helpful to publish experiment status from the data collection software (eg experiment start, experiment end, experiment needs human intervention).

From these (and the existing data streams) more inferred data streams can been added. Possible examples include "un-authorised entry", "Lab safe to enter indicator" and "Laser Ready to use".

### 5.2. Other Output Devices

It is planned to extend the system to allow the remote user to provide the laboratory with feedback via the message broker, and to investigate ways of presenting more complex data, such as experimental data plots, based on awareness of the chemists location.

### 5.3. Other Laboratories

We've shown the potential usefulness of these techniques within our laboratory (studying liquid surfaces), a logical follow-on is to apply these techniques to other spaces. The SmartTea project is looking at ways to automate the collection of laboratory data in the synthetic organic chemistry laboratory, with the aim to providing an end-to-end data curation solution. Part of this will be to record the environmental conditions that experiments are performed under. There is also scope to monitor reactions remotely, allowing the chemist to perform other work. Discussions are currently under way to discover how the technology described here can be applied to their work.

## Appendix

## A. IBM Message Brokers : Choosing the correct broker for the Job

IBM has several "message broker" products in the product portfolio. In decreasing order of size, range of functions, sophistication and price, these are WebSphere Business Integration Message Broker (WBIMB)[21], WebSphere Business Integration Event Broker (WBIEB)[22], and WebSphere Connection Server Micro Edition (WCSME). The first, WBIMB, is often referred to as the Enterprise Broker, and that runs on a server-class machine, supports multiple input and output messaging protocols, and has a rich set of tools and functions to develop message transformations and routing logic to enable the broker to act as a powerful, central communications and transformation hub for an Enterprise-scale middleware operation. By contrast the last mentioned, WCSME, more widely known as the "MicroBroker", is a small footprint (about 500K of Java) message broker, designed for use in embedded applications, such as for integrating sensors, actuators and applications at a remote location (for example, in the research lab). The MicroBroker uses MQTT as its communications protocol, and is able to selectively "bridge" some of the information topics to another broker (often an Enterprise Broker). This hierarchy of brokers implements the middleware fabric that enables the seamless delivery of messages from sensors to back-end applications, with aggregation and additional processing being performed at the point in the network where it makes most sense.

## References


[1] IBM. IBM United Kingdom Laboratories, Hursley Park. http://www.hursley.ibm.com.

[2] IBM. Mqtt.org. http://www.mqtt.org.

[3] Sun Microsystems. Mobile Information Device Profile. http://java.sun.com/products/midp/index.jsp.

[4] Jeremy G. Frey and Mike B. Hursthouse. From e-science to publication@source. In *National Policies on Open Access (OA) Provision for University Research Output, Southampton, UK*. University of Southampton, 2004.

[5] Equator. Website. http://www.equator.ac.uk/.

[6] BiosimGRID. A GRID Database for biomolecular simulations. http://www.biosimgrid.org/.

[7] DAME. Distributed Aircraft Maintenance Environment. http://www.cs.york.ac.uk/dame.

[8] CoAKTinG. Collaborative Advanced Knowledge Technologies in the grid. http://www.aktors.org/coakting.

[9] MYGRID. Directly Supporting the E-Scientist. http://www.mygrid.org.uk/.

[10] CCDC. The cambridge structural database - the world repository of small molecule crystal structures. http://www.ccdc.org.uk/products/csd/, 2004.

[11] Andy Stanford-Clark. Integrating monitoring and telemetry devices as part of enterprise information resources. Technical report, IBM, 2002.

[12] IBM. WebSphere MQ Telemetry Transport Specification. http://publib.boulder.ibm.com/infocenter/wbihelp/index.jsp?topic=/com.ibm.etools.mft.doc/ac10840_htm.

[13] IBM. IA92: WBI Brokers - Java implementation of WebSphere MQ Telemetry transport. http://www-1.ibm.com/support/docview.wss rs=171&uid=swg24006006&loc=en_US&cs=utf-8&lang=en.

[14] IBM. IA93: WBI Brokers - C implementation of WebSphere MQ Telemetry transport. http://www-1.ibm.com/support/docview.wss rs=171&uid=swg24006525&loc=en_US&cs=utf-8&lang=en.

[15] Wikipedia. General Packet Radio Service, defined at wikipedia.org. http://en.wikipedia.org/wiki/GPRS/.

[16] FloodNet. Pervasive Computing in the Environmment. http://envisense.org/floodnet/floodnet.htm.

[17] Wikipedia. Stateful firewall from wikipedia. http://en.wikipedia.org/wiki/Stateful_firewall, 2004.

[18] CombEchem. Firewalling, NAT and packet mangling for linux. http://www.netfilter.org/, 2004.

[19] Sophos. Virus information w32/sasser-a. http://www.sophos.com/virusinfo/analyses/w32sassera.html, 2004.

[20] Wikipedia. Radio Frequency IDentification, as defined at wikipedia.org. http://en.wikipedia.org/wiki/RFID.

[21] IBM. WebSphere Business Integration Message Broker. http://www-306.ibm.com/software/integration/wbimessagebroker.

[22] IBM. WebSphere Business Integration Event Broker. http://www-306.ibm.com/software/integration/wbieventbroker/.