# An overview of the EnTrak/ BuildAX eService delivery platform

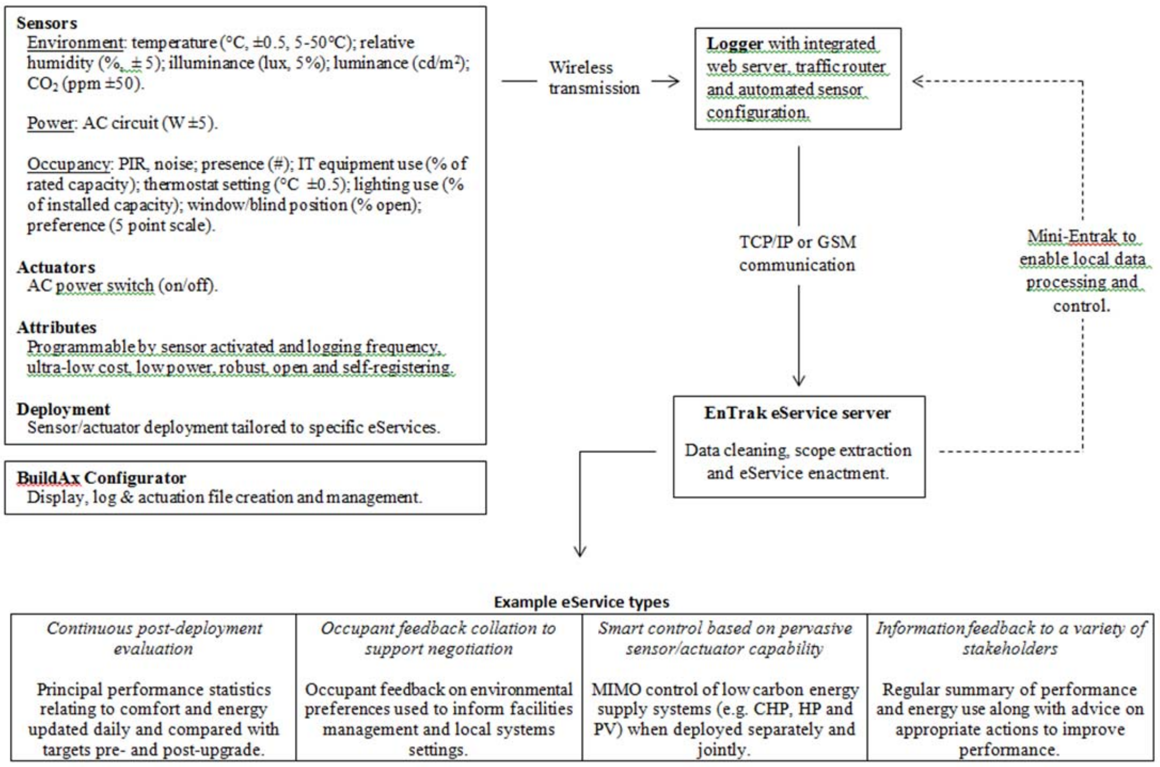
***Joe Clarke and Jon Hand***
***ESRU, University of Strathclyde***
***23 January 2015***

## Preamble

With grant funding from EPSRC (project EP/I000739/1), a collaboration between the Energy Systems Research Unit (ESRU) at the University of Strathclyde (www.strath.ac.uk/esru) and the Culture Lab at the University of Newcastle (www.ncl.ac.uk/culturelab/) established an *eService* delivery platform comprising three principal components: pervasive sensors for building monitoring ; a procedure to define the analytics to be applied to the captured data; and a mechanism for the delivery of outcomes to relevant stakeholders. To assist with the commercialisation of the research prototype, financial support from the Energy Technology Partnership (www.etp-scotland.ac.uk/) facilitated equipment CE marking and support for trial deployments by two SMEs operating in the 'internet of things' area. This report describes the platform and the procedures to deploy specific *eServices*.

## System summary

Figure 1 depicts the *eService* platform comprising wireless devices corresponding to environmental conditions, occupancy states and power usage linked wirelessly to a logger/router from which the monitored data may be fetched at the frequency required to enact information services tailored to the needs of particular recipients. The *eService* delivery component is a software application termed EnTrak, while the monitoring component is hardware termed BuildAX. Also shown in Figure 1 are the sensor specifications.



**Sensors**
Environment: temperature (°C, ±0.5, 5-50°C); relative humidity (%, ± 5); illuminance (lux, 5%); luminance (cd/m$^2$); $CO_2$ (ppm ±50).

Power: AC circuit (W ±5).

Occupancy: PIR, noise; presence (#); IT equipment use (% of rated capacity); thermostat setting (°C ±0.5); lighting use (% of installed capacity); window/blind position (% open); preference (5 point scale).

**Actuators**
AC power switch (on/off).

**Attributes**
Programmable by sensor activated and logging frequency, ultra-low cost, low power, robust, open and self-registering.

**Deployment**
Sensor/actuator deployment tailored to specific eServices.

**BuildAx Configurator**
Display, log & actuation file creation and management.

**Logger** with integrated web server, traffic router and automated sensor configuration.

**EnTrak eService server**
Data cleaning, scope extraction and eService enactment.

Wireless transmission
TCP/IP or GSM communication
Mini-Entrak to enable local data processing and control.

**Example eService types**

| *Continuous post-deployment evaluation* | *Occupant feedback collation to support negotiation* | *Smart control based on pervasive sensor/actuator capability* | *Information feedback to a variety of stakeholders* |
|---|---|---|---|
| Principal performance statistics relating to comfort and energy updated daily and compared with targets pre- and post-upgrade. | Occupant feedback on environmental preferences used to inform facilities management and local systems settings. | MIMO control of low carbon energy supply systems (e.g. CHP, HP and PV) when deployed separately and jointly. | Regular summary of performance and energy use along with advice on appropriate actions to improve performance. |

Figure 1: EnTrak/ BuildAX *eService* delivery platform.

For any particular *eService*, data is gathered from field sensors deployed as required. The BuildAX family comprises the following components.

*LRS* – a logger/router to store locally and/or transmit monitored data to a remote location where it may be utilised by EnTrak. The device also acts as a Web server.
*ENV* – for the monitoring of indoor environmental conditions, including temperature, relative humidity, illuminance, movement and surface contact (e.g. door opening).
*CO2* – for the monitoring of $CO_2$ concentration.
*GAS* – for the monitoring of gas consumption.
*PWR* – for the monitoring of electricity consumption.
*PSW* – a remote controllable electrical switch.

Figure 2 shows the LRS (left) and an ENV sensor. The system is made available under an Open Source/Common license agreement, with EnTrak available at no cost and BuildAX at low cost from a single supplier at the present time (Appendix 1 gives supplier and cost details).

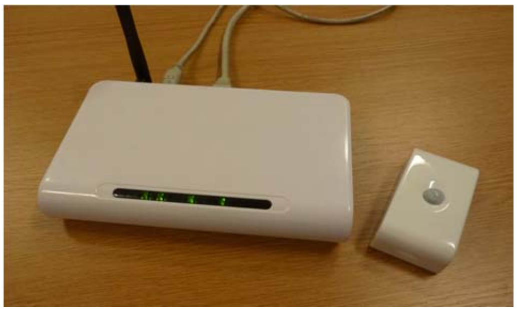
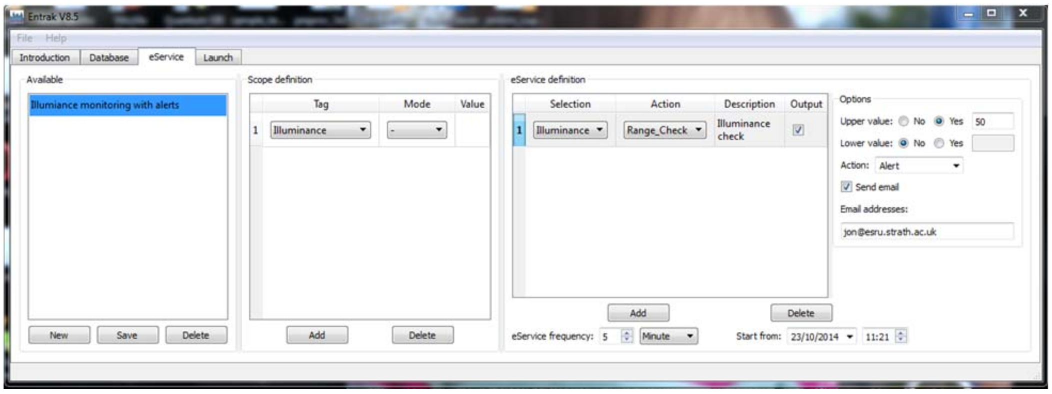


Figure 2: BuildAX LRS (left) and ENV sensor.

An example *eService* delivered by EnTrak when connected to 2 ENV sensors is shown in Figure 3. Here, time series data relating to light level are scrutinised and an alert issued when a low level is detected (e.g. due to a lamp failure).



Available

Illumiance monitoring with alerts

Scope definition

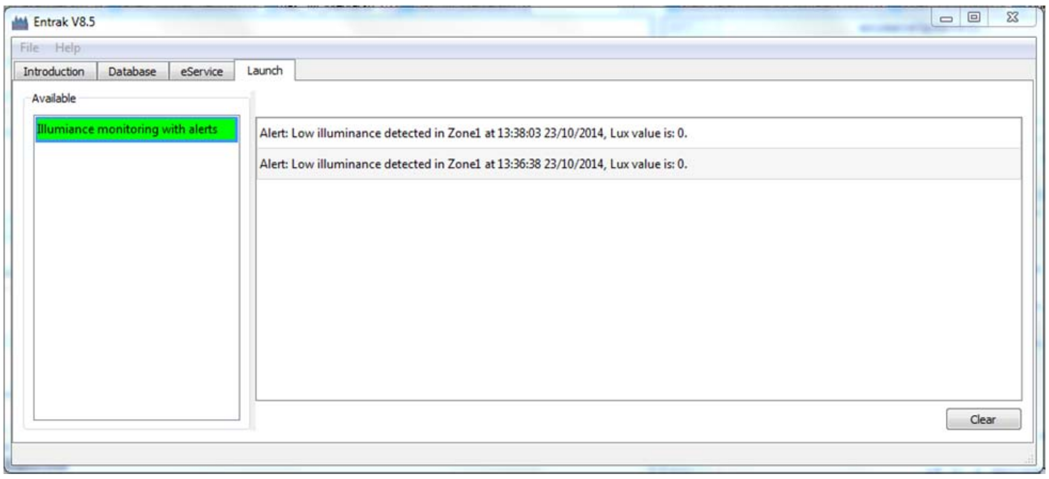
| | Tag | Mode | Value |
|---|---|---|---|
| 1 | Illuminance | - | |

eService definition

| | Selection | Action | Description | Output |
|---|---|---|---|---|
| 1 | Illuminance | Range_Check | Illuminance check | ✓ |

Options
Upper value: No [✓ Yes] 50
Lower value: [✓ No] Yes
Action: Alert
✓ Send email
Email addresses:
jon@esru.strath.ac.uk

eService frequency: 5 Minute
Start from: 23/10/2014 11:21



Alert: Low illuminance detected in Zone1 at 13:38:03 23/10/2014, Lux value is: 0.
Alert: Low illuminance detected in Zone1 at 13:36:38 23/10/2014, Lux value is: 0.

Figure 3: An EnTrak light level *eService* definition (upper) and outcome (lower).

As depicted in Figure 4, the LRS receives data from paired sensors as well as data fetch requests issued by EnTrak via *enget* at a time frequency associated with the *eService*. EnTrak may be located on the same or different network with all communication enacted by *enget* via the *wget* protocol. This fetch results in a data file being placed in a filestore located on the EnTrak (or at a remote cloud location for collection by another EnTrak instance). This filestore (or cloud location) is polled by EnTrak and the contents of the data files imported to the SQL database that underlies the *eService* being enacted.

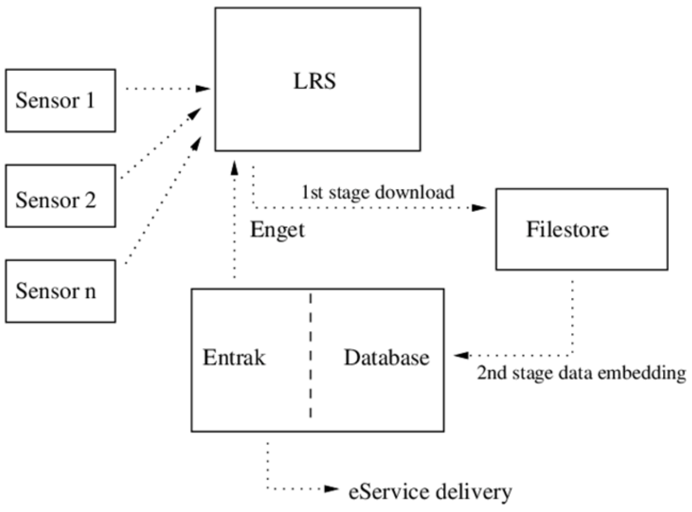
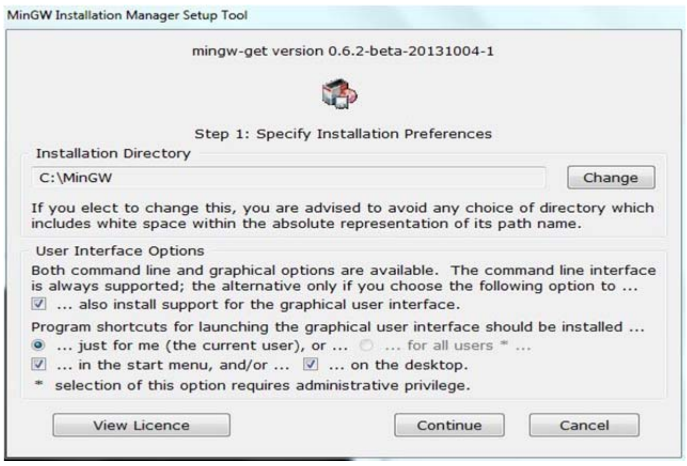


Figure 4: Communication paths associated with an *eService*.

Further information on the EnTrak and BuildAX components are available elsewhere: http://www.esru.strath.ac.uk/Programs/EnTrak.htm and http://openmovement.googlecode.com/svn/docs/buildax/site/index.html respectively.

**EnTrak server configuration**

The scripting facilities, command shell and applications found on Linux/OSX computers are used to automate the above data transfers based on the *wget* protocol. These tools are also available on Windows computers via a lightweight toolset named MinGW (Minimalist GNU for Windows - http://www.mingw.org). This allows for the deployment of common scripts for data gathering tasks across a variety of computer platforms and operating systems. Figure 5 summarises the installation procedure for MinGW.


MinGW Installation Manager Setup Tool

mingw-get version 0.6.2-beta-20131004-1

Step 1: Specify Installation Preferences

Installation Directory

C:\MinGW

If you elect to change this, you are advised to avoid any choice of directory which includes white space within the absolute representation of its path name.

User Interface Options

Both command line and graphical options are available. The command line interface is always supported; the alternative only if you choose the following option to ...

[✓] ... also install support for the graphical user interface.

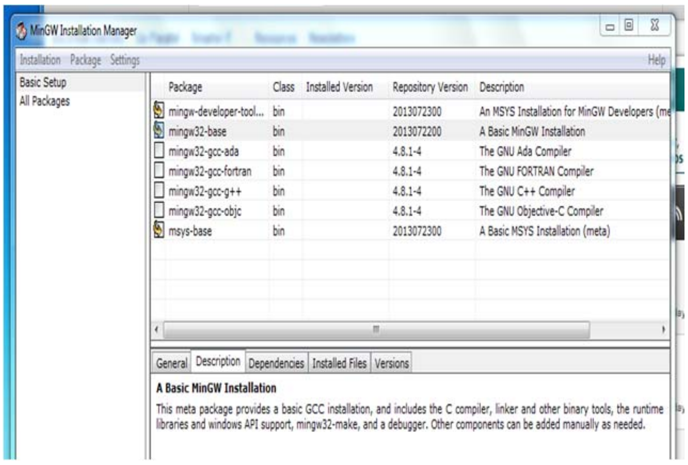
Program shortcuts for launching the graphical user interface should be installed ...

[✓ ... just for me (the current user)], or ... ... for all users * ...

[✓] ... in the start menu, and/or ... [✓] ... on the desktop.

* selection of this option requires administrative privilege.

Define the install location and other required options.

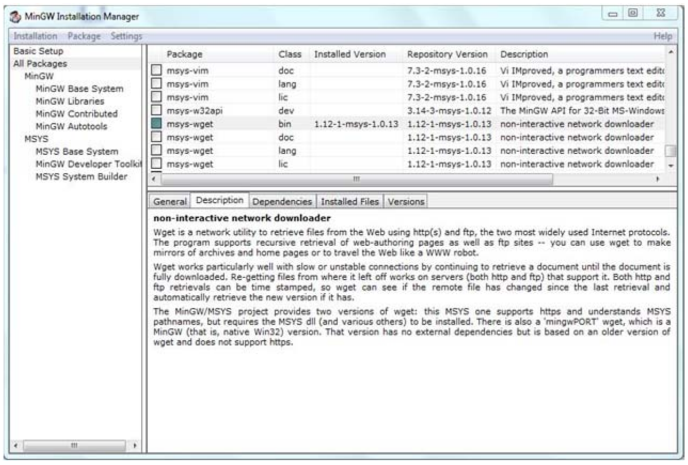

MinGW Installation Manager

| | Package | Class | Installed Version | Repository Version | Description |
|---|---|---|---|---|---|
| ✓ | mingw-developer-tool... | bin | | 2013072300 | An MSYS Installation for MinGW Developers (me |
| ✓ | mingw32-base | bin | | 2013072200 | A Basic MinGW Installation |
| ☐ | mingw32-gcc-ada | bin | | 4.8.1-4 | The GNU Ada Compiler |
| ☐ | mingw32-gcc-fortran | bin | | 4.8.1-4 | The GNU FORTRAN Compiler |
| ☐ | mingw32-gcc-g++ | bin | | 4.8.1-4 | The GNU C++ Compiler |
| ☐ | mingw32-gcc-objc | bin | | 4.8.1-4 | The GNU Objective-C Compiler |
| ✓ | msys-base | bin | | 2013072300 | A Basic MSYS Installation (meta) |

**A Basic MinGW Installation**

This meta package provides a basic GCC installation, and includes the C compiler, linker and other binary tools, the runtime libraries and windows API support, mingw32-make, and a debugger. Other components can be added manually as needed.

Include msys-base, mingw32-base and mingw-developer-toolkit. If an *eService* needs to compile code select the gcc and g++ compilers. Additional windows ports of applications and libraries can be found at <http://gnuwin32.sourceforge.net>.


MinGW Installation Manager

| | Package | Class | Installed Version | Repository Version | Description |
|---|---|---|---|---|---|
| ☐ | msys-vim | doc | | 7.3-2-msys-1.0.16 | Vi IMproved, a programmers text editc |
| ☐ | msys-vim | lang | | 7.3-2-msys-1.0.16 | Vi IMproved, a programmers text editc |
| ☐ | msys-vim | lic | | 7.3-2-msys-1.0.16 | Vi IMproved, a programmers text editc |
| ☐ | msys-w32api | dev | | 3.14-3-msys-1.0.12 | The MinGW API for 32-Bit MS-Windows |
| 🟩 | msys-wget | bin | 1.12-1-msys-1.0.13 | 1.12-1-msys-1.0.13 | non-interactive network downloader |
| ☐ | msys-wget | doc | | 1.12-1-msys-1.0.13 | non-interactive network downloader |
| ☐ | msys-wget | lang | | 1.12-1-msys-1.0.13 | non-interactive network downloader |
| ☐ | msys-wget | lic | | 1.12-1-msys-1.0.13 | non-interactive network downloader |

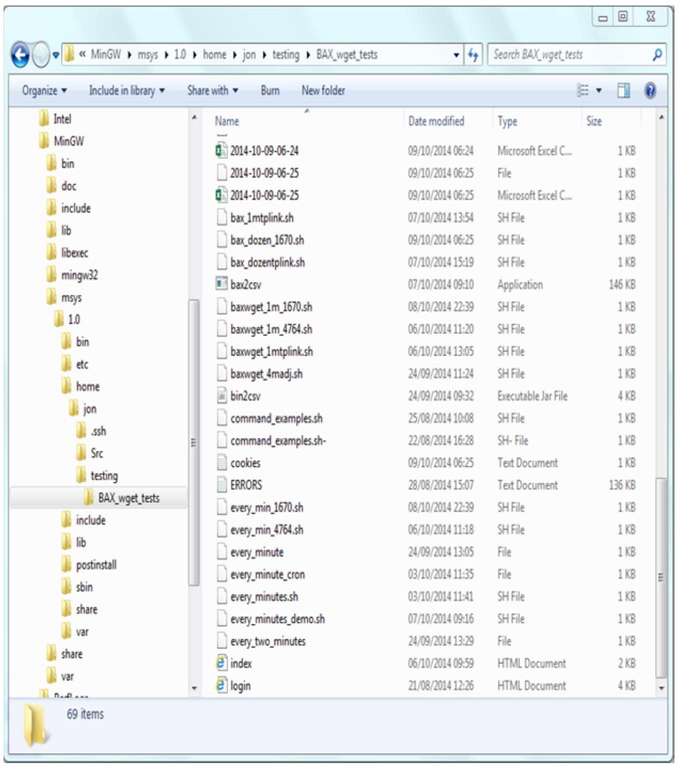
**non-interactive network downloader**

Wget is a network utility to retrieve files from the Web using http(s) and ftp, the two most widely used Internet protocols. The program supports recursive retrieval of web-authoring pages as well as ftp sites -- you can use wget to make mirrors of archives and home pages or to travel the Web like a WWW robot.

Wget works particularly well with slow or unstable connections by continuing to retrieve a document until the document is fully downloaded. Re-getting files from where it left off works on servers (both http and ftp) that support it. Both http and ftp retrievals can be time stamped, so wget can see if the remote file has changed since the last retrieval and automatically retrieve the new version if it has.

The MinGW/MSYS project provides two versions of wget: this MSYS one supports https and understands MSYS pathnames, but requires the MSYS dll (and various others) to be installed. There is also a 'mingwPORT' wget, which is a MinGW (that is, native Win32) version. That version has no external dependencies but is based on an older version of wget and does not support https.

Within the *all packages* section select *wget*. After selection use the file tab to 'apply changes'.



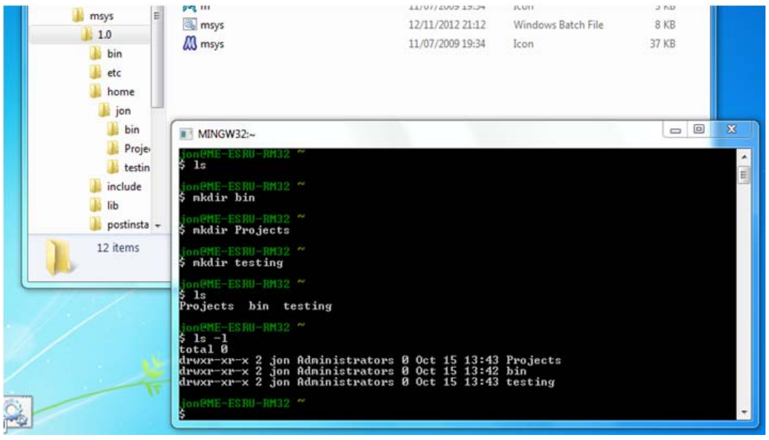
MinGW ▸ msys ▸ 1.0 ▸ home ▸ jon ▸ testing ▸ BAX_wget_tests

| Name | Date modified | Type | Size |
|---|---|---|---|
| 2014-10-09-06-24 | 09/10/2014 06:24 | Microsoft Excel C... | 1 KB |
| 2014-10-09-06-25 | 09/10/2014 06:25 | File | 1 KB |
| 2014-10-09-06-25 | 09/10/2014 06:25 | Microsoft Excel C... | 1 KB |
| bax_1mtplink.sh | 07/10/2014 13:54 | SH File | 1 KB |
| bax_dozen_1670.sh | 09/10/2014 06:25 | SH File | 1 KB |
| bax_dozentplink.sh | 07/10/2014 15:19 | SH File | 1 KB |
| bax2csv | 07/10/2014 09:10 | Application | 146 KB |
| baxwget_1m_1670.sh | 08/10/2014 22:39 | SH File | 1 KB |
| baxwget_1m_4764.sh | 06/10/2014 11:20 | SH File | 1 KB |
| baxwget_1mtplink.sh | 06/10/2014 13:05 | SH File | 1 KB |
| baxwget_4madj.sh | 24/09/2014 11:24 | SH File | 1 KB |
| bin2csv | 24/09/2014 09:32 | Executable Jar File | 4 KB |
| command_examples.sh | 25/08/2014 10:08 | SH File | 1 KB |
| command_examples.sh- | 22/08/2014 16:28 | SH- File | 1 KB |
| cookies | 09/10/2014 06:25 | Text Document | 1 KB |
| ERRORS | 28/08/2014 15:07 | Text Document | 136 KB |
| every_min_1670.sh | 08/10/2014 22:39 | SH File | 1 KB |
| every_min_4764.sh | 06/10/2014 11:18 | SH File | 1 KB |
| every_minute | 24/09/2014 13:05 | File | 1 KB |
| every_minute_cron | 03/10/2014 11:35 | File | 1 KB |
| every_minutes.sh | 03/10/2014 11:41 | SH File | 1 KB |
| every_minutes_demo.sh | 07/10/2014 09:16 | SH File | 1 KB |
| every_two_minutes | 24/09/2014 13:29 | File | 1 KB |
| index | 06/10/2014 09:59 | HTML Document | 2 KB |
| login | 21/08/2014 12:26 | HTML Document | 4 KB |

69 items

The standard install results in a C:\MinGW folder with a msys\1.0 subfolder. On this computer a user jon has created a testing folder (see the MinGW command syntax below) where *eService* scripts are run.

The installer may place an icon on your computer, if not, find it at C:\MinGW\msys\1.0\msys.bat and copy it to your desktop.



```
jon@ME-ESRU-RM32 ~
$ ls

jon@ME-ESRU-RM32 ~
$ mkdir bin

jon@ME-ESRU-RM32 ~
$ mkdir Projects

jon@ME-ESRU-RM32 ~
$ mkdir testing

jon@ME-ESRU-RM32 ~
$ ls
Projects  bin  testing

jon@ME-ESRU-RM32 ~
$ ls -l
total 0
drwxr-xr-x 2 jon Administrators 0 Oct 15 13:43 Projects
drwxr-xr-x 2 jon Administrators 0 Oct 15 13:42 bin
drwxr-xr-x 2 jon Administrators 0 Oct 15 13:43 testing

jon@ME-ESRU-RM32 ~
$
```

The msys.bat starts a command window with an interpreter similar to *bash* and *sh* on a Linux computer. It looks like a DOS window, but has a more powerful syntax.

Figure 5: The MinGW installation procedure.

The command window of MinGW supports a subset of the standard commands found on a Linux computer; essential commands include the following.

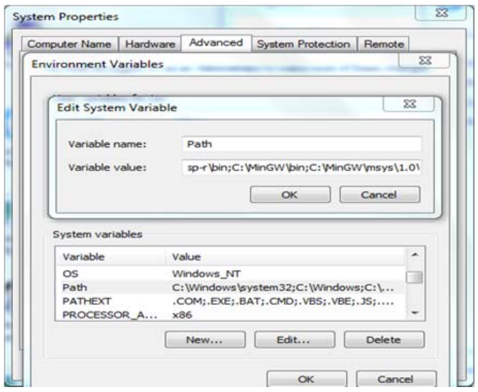
- ls (list), e.g. *ls Scripts* lists the contents of the folder Scripts.
- ls -l (long list), e.g. *ls -l Scripts* also includes the size of the files and modification date.
- cd (change directory), e.g. *cd Scripts* places you in the Scripts folder.
- cp (copy), e.g. *cp report.txt report_backup.txt* makes a copy of the file.
- rm (remove), e.g. *rm report.txt* deletes the file
- chmod (alter permissions), e.g. *chmod a+x every_minute.sh* makes a script executable.

MinGW command terminals do not respond to DOS commands but can be used instead of DOS commands to manage files in standard Windows folders. When a command window is launched the user is placed in folder C:\MinGW\msys\1.0\home\your_user_name. It is usual to populate this folder with sub-folders such as Projects, Scripts *etc*. via commands issued within the command window:

*cd (takes you to your HOME folder);*

*mkdir bin* (for your own scripts);
*mkdir Projects (for project data);*
*etc.*

To work with scripts a text editor is required (not a word processor), e.g. NotePad++ <http://notepad-plus-plus.org>. On Linux platforms use the *vim* editor within the MinGW terminal.
You will need to update the Windows system PATH environment variable to include C:\MinGW\bin and C:\MinGW\msys\1.0\bin as shown in Figure 6. Once you have done this you should log out of the MinGW terminal (type *exit*) and relaunch the msys.bat file.


Variable name: Path
Variable value: sp-r\bin;C:\MinGW\bin;C:\MinGW\msys\1.0\

System variables

| Variable | Value |
|---|---|
| OS | Windows_NT |
| Path | C:\Windows\system32;C:\Windows;C:\... |
| PATHEXT | .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.... |
| PROCESSOR_A... | x86 |

Figure 6: Editing the PATH system environment variable.

The final step is to download EnTrak from www.esru.strath.ac.uk/Programs/EnTrak.htm and comply with the given installation procedure.

**BuildAX deployment**

Depending on the requirements of the *eService* being established, a number of sensors and LRS devices are selected and made ready by pairing a group of the former with one of the latter. A typical deployment is as follows.

1. Select a location for the LRS adjacent to a power socket (the LRS requires a 5V, 1A power supply delivered via a USB cable) or a computer with a USB connection. Unless the LRS is running stand-alone (i.e. as a data logger with no routing capability) an Ethernet connection is also required.
2. Use the LRS mac address to determine the equivalent IP address and then establish a temporary network connection to the LRS by entering this IP address within a Web browser and providing the required user name and password. Switch to the sensors tab and the RSI (radio strength) topic.
3. Pair a fast response sensor with the LRS and check that the data is being received as depicted in Figure 7 (here there is a gap in readings between 11h45 and 11h52).
4. Typically, one person will check the sensor RSI graph as another person slowly walks around the building while communicating via mobile. Communication is usually secure as long as the RSI is reported to be better than -110db.

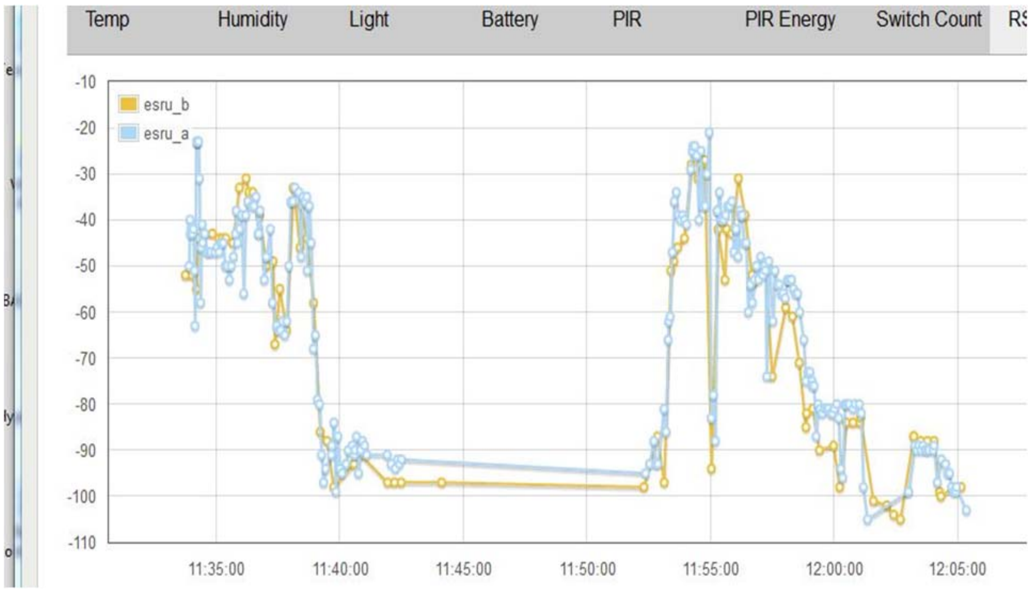


Figure 7: Data receipt and signal strength survey result.

5. Finally, launch the application used by EnTrak to fetch monitored data (in a MYSYS command window or Linux shell) on the computer hosting EnTrak to confirm that the data fetch procedure is operational.

   *enget* --IP <as above> --mode test

   A successful connection will result in the display of the retrieved data as depicted in Figure 8 (here for the case of a deployment with 2 ENV sensors).

```
Enget connected to IP=130.159.47.84 in test mode - success

Sample data:
2015/03/25,14:53:08,42FE2C58,-62,1,93,20,2565,23.00,252,121,1388,1401,1
2015/03/25,14:53:11,42081734,-72,1,208,20,2688,27.47,211,712,0,49251,1
2015/03/25,14:53:14,422E7342,-88,1,83,20,2748,26.60,222,172,3275,52624,1
2015/03/25,14:53:15,420490F0,-94,1,223,20,2767,38.17,101,1002,39431,25968,1
2015/03/25,14:53:30,42D89A75,-62,1,107,20,2844,33.35,182,0,14003,36781,1
2015/03/25,14:53:35,42AA57D9,-66,1,36,20,2608,26.19,212,84,54345,43829,1
2015/03/25,14:54:06,42FE2C58,-67,1,94,20,2565,23.00,252,121,1388,1975,1
2015/03/25,14:54:10,42081734,-72,1,209,20,2688,27.47,210,714,0,51014,1
2015/03/25,14:54:12,422E7342,-86,1,84,20,2748,26.60,222,168,3275,53473,1
```

where the data columns are:

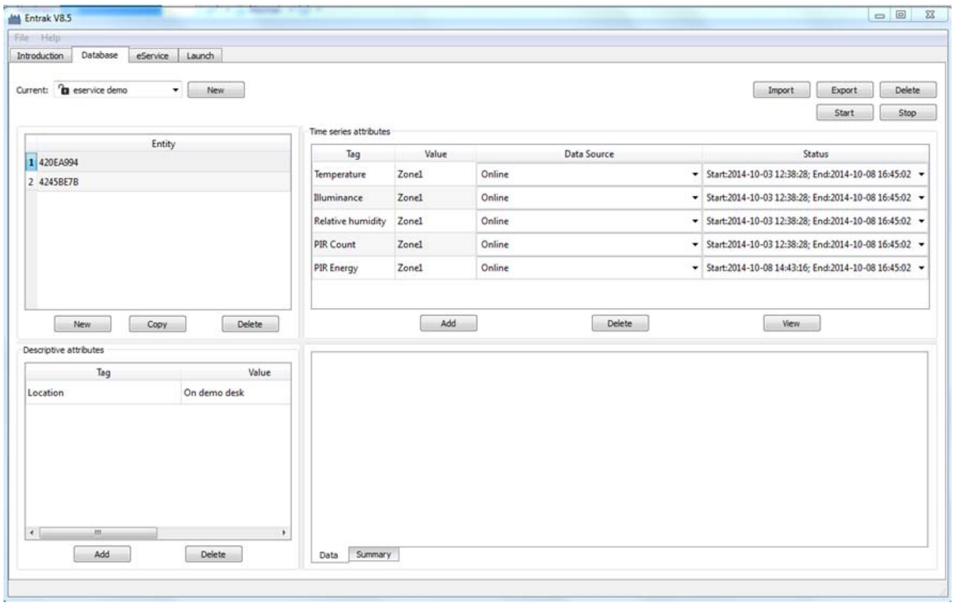
- Date – of the packet (yyyy/mm/dd formatted ISO 8601)
- Time – of the packet (hh:mm:ss)
- Sensor name or address if no name assigned
- Received Signal Strength Indication (dBm)
- Received packet type:
  - 0 – encryption packet type (not seen in CSV output)
  - 1 – normal packet received at sensor transmit interval
  - 2 – packet sent when PIR sensor triggered
  - 3 – packet sent when magnetic switch triggered
- Packet identifier (sensors send packets incrementally)
- Sensor-configured transmission power (dBm)
- ENV battery level (mV)
- Relative Humidity (%)
- Temperature (°C x 10)
- Luminous flux (Lux)

Activation counts of the PIR sensor
PIR energy last captured
Magnetic switch triggers

Figure 8: Example of feedback from a test connection using *enget*.

## EnTrak deployment

Typically the definition of a new *eService* follows a 3 stage procedure. First, a database is opened and populated as required. Figure 9 shows an example for the case of a 2 sensor deployment.



| | Entity |
|---|---|
| 1 | 420EA994 |
| 2 | 4245BE7B |

Time series attributes

| Tag | Value | Data Source | Status |
|---|---|---|---|
| Temperature | Zone1 | Online | Start:2014-10-03 12:38:28; End:2014-10-08 16:45:02 |
| Illuminance | Zone1 | Online | Start:2014-10-03 12:38:28; End:2014-10-08 16:45:02 |
| Relative humidity | Zone1 | Online | Start:2014-10-03 12:38:28; End:2014-10-08 16:45:02 |
| PIR Count | Zone1 | Online | Start:2014-10-03 12:38:28; End:2014-10-08 16:45:02 |
| PIR Energy | Zone1 | Online | Start:2014-10-08 14:43:16; End:2014-10-08 16:45:02 |

Descriptive attributes

| Tag | Value |
|---|---|
| Location | On demo desk |

Figure 9: defining the database underpinning the required *eService*.

Here, the sensors each have one static attribute and 5 dynamic attributes as shown, with the latter based on online data capture (i.e. from a matched BuildAX deployment). In this way deployments of arbitrary complexity may be defined, including data capture from different locations at different frequency.

Second, the eService content is defined by scoping on the entity attributes as required and defining the actions to be applied to the data returns when the *eService* is running (see Figure 3 upper). In this way separate *eServices* may relate to the same database.

Last, the required *eService* is launched and the output directed to the display type required, for example as shown in Figure 10 for the case of a real-time monitoring *eService*.

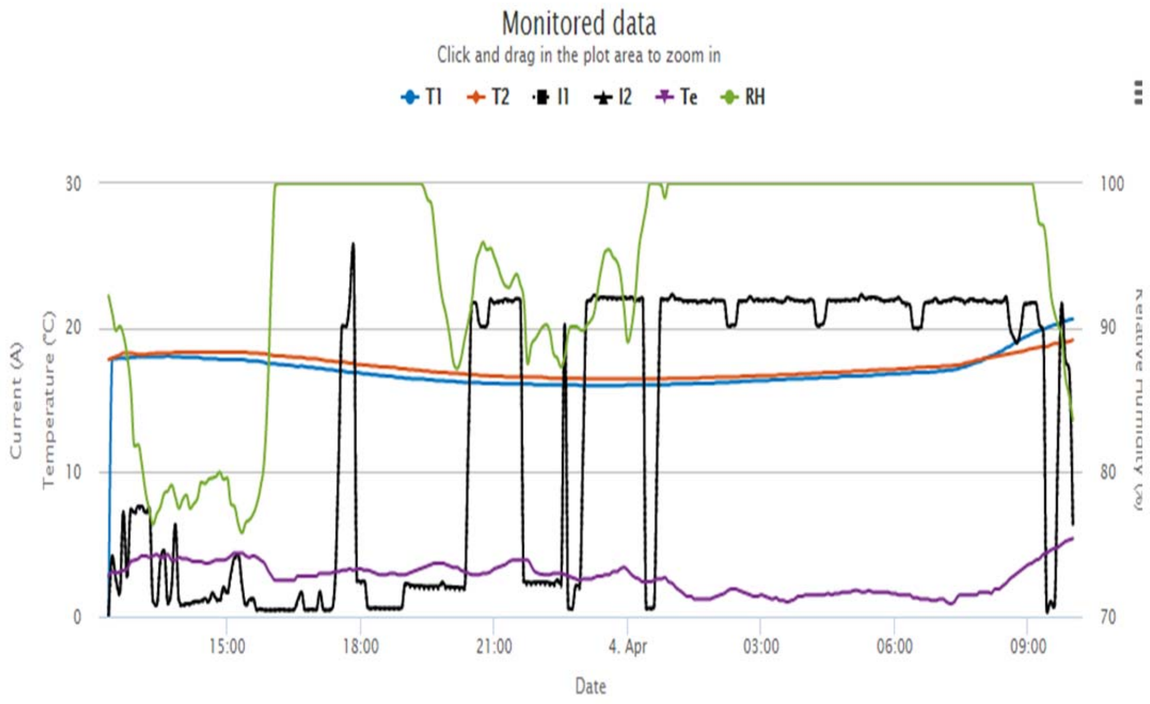


Figure 10: Sample output as delivered by an online monitoring *eService*.

**Appendix 1**
**BuildAX Price List January 2015**

BuildAX ENV £90.00
BuildAX LRS £200.00
BuildAX Bundle 1 1 Router, 5 sensors, USB Cables, Ethernet Cable and SD Card £500.00
BuildAX Bundle 2 1 Router, 25 sensors, USB Cables, Ethernet Cable and SD Card £1,875.00
BuildAX Bundle 3 2 Router, 50 sensors, USB Cables, Ethernet Cable and SD Card £2,500.00

All prices exclusive of VAT and delivery.

Orders to:
Axivity Ltd
4 Southhands Road
York YO23 1NP

T: 01904 215 950
E: info@axivity.com
W: www.axivity.com