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.

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 CO₂ 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.](image)

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).

![Image showing eService interface](image)
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.

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.
Define the install location and other required options.

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>.

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

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:

\[ cd \text{(takes you to your HOME folder)}; \]
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.

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.
5. Finally, launch the application used by EnTrak to fetch monitored data (in a MYSIS 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:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Sensor name</th>
<th>Received Signal Strength Indication (dBm)</th>
<th>Received packet type</th>
<th>Packet identifier</th>
<th>Sensor-configured transmission power (dBm)</th>
<th>ENV battery level (mV)</th>
<th>Relative Humidity (%)</th>
<th>Temperature (°C x 10)</th>
<th>Luminous flux (Lux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015/03/25,14:53:08,42FE2C58,-62,1,93,20,2565,23.00,252,121,1388,1401,1</td>
<td>14:53:08</td>
<td>42FE2C58</td>
<td>-62,1,93,20,2565,23.00,252,121,1388,1401,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:53:11,42081734,-72,1,208,20,2688,27.47,211,712,0,49251,1</td>
<td>14:53:11</td>
<td>42081734</td>
<td>-72,1,208,20,2688,27.47,211,712,0,49251,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:53:14,422E7342,-88,1,83,20,2748,26.60,222,172,3275,52624,1</td>
<td>14:53:14</td>
<td>422E7342</td>
<td>-88,1,83,20,2748,26.60,222,172,3275,52624,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:53:15,420490F0,-94,1,223,20,2767,38.17,181,101,1002,39431,1</td>
<td>14:53:15</td>
<td>420490F0</td>
<td>-94,1,223,20,2767,38.17,181,101,1002,39431,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:53:30,42D89A75,-62,1,107,20,2844,33.35,182,0,14003,36781,1</td>
<td>14:53:30</td>
<td>42D89A75</td>
<td>-62,1,107,20,2844,33.35,182,0,14003,36781,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:53:35,42AA57D9,-66,1,36,20,2608,26.19,212,84,54345,43829,1</td>
<td>14:53:35</td>
<td>42AA57D9</td>
<td>-66,1,36,20,2608,26.19,212,84,54345,43829,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:54:06,42FE2C58,-67,1,94,20,2565,23.00,252,121,1388,1975,1</td>
<td>14:54:06</td>
<td>42FE2C58</td>
<td>-67,1,94,20,2565,23.00,252,121,1388,1975,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:54:10,42081734,-72,1,209,20,2688,27.47,210,714,0,51014,1</td>
<td>14:54:10</td>
<td>42081734</td>
<td>-72,1,209,20,2688,27.47,210,714,0,51014,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015/03/25,14:54:12,422E7342,-86,1,84,20,2748,26.60,222,168,3275,53473,1</td>
<td>14:54:12</td>
<td>422E7342</td>
<td>-86,1,84,20,2748,26.60,222,168,3275,53473,1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)
EnTrak deployment

Typically the definition of a new \textit{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.

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 Southands Road
York YO23 1NP

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