Coursework Item 6

Olivia Howard, Newnham College, ofyh2

For this project an environment monitor with an alarm was developed. This addresses the problem of the difficulty in getting reliable data for the current environment you are in, and being notified whether the temperature, pressure, humidity or chance of rain goes outside required thresholds.

This is useful for a lot of people to be able to keep track of their exact current environment rather than just the overall area they're in. It can be helpful for people when putting clothes out to dry, or for people who have a book collection for which they do not want it to be too humid. Also because it can be controlled remotely via your phone, it allows the device to be put in a sealed environment such as a museum exhibit.

Currently it is possible to get overall readings for the local area using weather reports from a variety of websites/apps such as 'Weather.com'. It is also possible to get apps such as rain alarm, which notify you when the rain is near to your area. However these do not allow for monitoring the conditions exactly where you are, for example in your house.
In addition to this there are existing systems which allow for you to get monitors for your house which read the temperature/pressure/humidity such as "enviromon". These often have a unit which displays the data rather than having it connected to a mobile phone, although with the systems provided by enviromon, various alarms such as visual and SMS alarms can be set. However these are expensive and are often not portable, whereas my project allows for a portable sensing system, or for the Sensortag to be left in one place and the values read remotely and have alarms thresholds set via phone.
There are approaches that are similar to this projects that have been taken; out of the box by default the Sensortag is able to display the information from the sensors of the Sensortag from the phone, but do not allow for alarms to be set and have a lot more information that just that of the things relating to the environment. Similar apps have been developed such as a weather app, however this also just integrates information on the Sensortag.

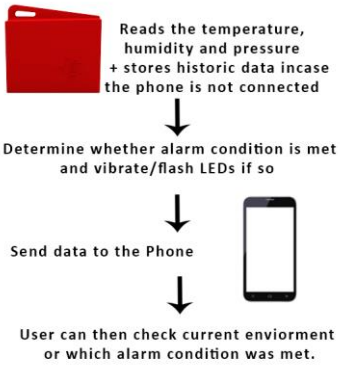

Figure 1 – The system of the Enviorment Monitor and typical use case.

To create an environment monitor it was decided after examining the available resources that the best approach would be to develop a firmware for the Sensortag and an interface for a mobile phone; using the Sensortag for its sensors and buzzer/led and to display this information on a phone. This system can be seen in Figure 1.

To create the firmware; initially tutorials on editing the software using Code Composer were carried out. Due to time constraints an already existing firmware was edited and added to.
The main components of the firmware included the Bluetooth connection between the Sensortag and the phone for the sending/receiving of the information between the two and the conditions for setting off the alarm and turning it off.
To allow for the communications between the phone and the Sensortag, already existing BLE documentation was used and altered to allow for UUID's, initially these were made for each of the parameters required for function: e.g temperature, the temperature thresholds, pressure, etc. This can be seen in the envservice.c and envservice.h files. Initial testing of was done using an already existing app "BLE Scanner", this can be seen in Figure 2. Using this the requesting of values from the sensors and setting variables (for the thresholds of the alarms) was tested.
After adding all the UUID's the code would not compile properly due to memory issues due to the large number of UUID's.

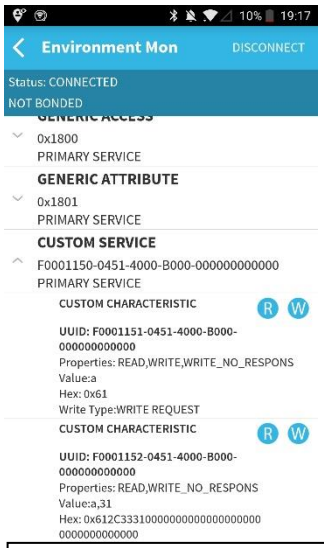

Figure 2 – Sending the value "a" to the Sensortag using BLE Scanner and receiving "a,31" back indicating a temperature of 31

To fix this it was changed so that only two UUID's were used; one for commands to be sent to the Sensortag and one for the returned information to be read back on the phone. To be able to implement all the different parameters that can be requested and set, single letter 'codes' were implemented to identify what was being set/requested to read, for example sending "T,50" sets the upper temperature boundary to 50. Full details of this implementation can be seen in the code files.

The next problem was getting the buzzer working as just switching the buzzer on did not work. Following research it became clear that a pwm signal would have to be sent. Existing code for an LED pwm was amongst the examples given and so was modified to instead use the port for the buzzer rather than LED; giving the Sensortag the ability to audibly able to notify the user when a condition is met. This is in the emon.c file (location described in the readme of the repository). In addition to this the comparisons for the alarm conditions function is also defined in this file, to determine the rain chance basic research was done and using the information found at "https://www.thoughtco.com/how-to-read-a-barometer-3444043" a basic implementation of whether it would rain was able to be implemented.

Once the Sensortag firmware was correctly working with BLE Scanner, software for an app was developed. The easiest way to was to use Evothings, which already had existing examples for applications using the Sensortag. Looking at these and using basic documentation about writing scripts in javascript/html5 it was possible to create the interface for the application; implementing a button to switch off the alarm and sliders to set the thresholds for the alarms, with the reasoning behind the alarm going off stated near the top.

From this implementation a successful app was able to be created and to be observed to work as was expected, with the final implemented application seen in Figures 4 and 5. Due to time constraints it was not possible to fully implement a system which worked robustly. At present it can only show the reason for the last alarm is shown, a queue with timestamps would have been the ideal approach. In addition to this only a basic rain percentage was implemented. Also the method chosen to detect change for the rain calculation requires at least 16mintes before it can start to determine a chance of rain.

With use it also became clear that the Sensortag battery appeared to be running out very quickly within a couple of days, and hence had to often be replaced, to monitor this a simple battery voltage monitor was implemented and can be seen in the app. Keeping note and plotting the voltage over time, Figure 5 was able to be derived, it also became clear that once the voltage reached below ~2.52V the humidity sensor no longer appeared to work as expected, Due to time limitations it was not possible to come up with an improvement to minimise the battery use for longer use, although Bluetooth polling was reduced to once a second from 10 times a second and readings are only done once a minute.

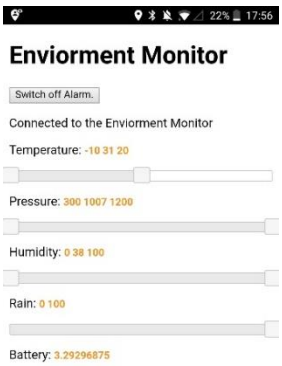

Figure 3 – A screenshot of the finished phone interface displayed when the app is run

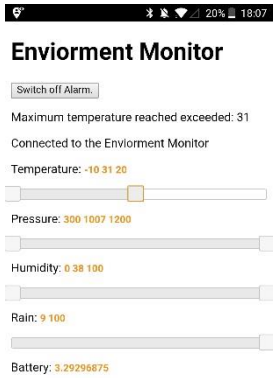

Figure 4 – A screenshot of the phone interface when an alarm condition is met

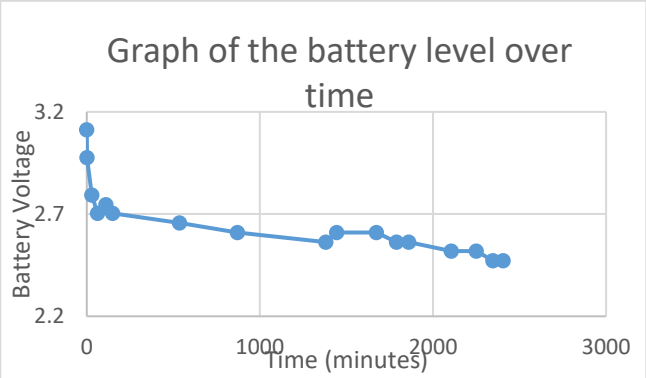

Figure 5 – A graph of readings of the battery voltage of the Sensortag over time

This results in there sometimes being a delay in in the alarm, however as environment conditions don't usually change very fast this seems acceptable. Another solution would be to look into integrating a battery with larger capacity.

It also became clear that the temperature sometimes appeared to read hotter than expected.

Finally it was possible to compare a set of readings for a Sensortag to the weather forcast on 'Weather.com' for the area; shown in Figure 6. It is clear that the temperature appears to be significantly higher, but other than this the readings appear to match up, even the rain suprisingly.

Measurement	*Sensortag Reading*	*'Weather.com' data*
Temperature	31	23
Pressure	1007	1007.1
Humidity	38	39
Rain %	9	10

Figure 6 – A table with readings of the Environment monitor and the information stated in the Cambridge area the 8th May 2018 at approximately 6pm