BSc (Honours) in Computing - IoT Principles - Assignment 1 (20% of Module)

1 Background

You will be using the Raspberry Pi and sensor devices in your future projects. This assignment is structured to give you experience in accessing sensor readings, publishing them to the cloud, retrieving and graphing the results in a HTML 5 application. To complete the assignment you will need to familiarize yourself with the `dweet.io` API, and it's associated implementation in Python: dweepy. You will be able to utilize the example code for accessing the Grove sensors. Such examples are given in the user manual which accompanied the Grove Kit, a PDF of this guide is also on Moodle. The guide provides a GitHub repository where the python scripts for the Grove sensors can be downloaded from.

2 Tasks

Write a Python script which:

- Collects observations from three different sensors such as temperature, humidity, state of LED, or a button pressed boolean.
- Additionally submit some configuration information read in from a file e.g. latitude, longitude, 'thing' ID etc.
- Submits this information to `dweet.io`, see Listing 1 for an example of a partial solution.
- Saves the information to a local database

Write a client script in JavaScript to:

- Consume the sensor readings.
- Create a separate chart for each kind of data (HighCharts is recommended).
- Use HTML LocalStorage to store the last 100 readings.

Listing 1: Sample Python code which submits fake sensor data to `dweet.io` using the dweepy API implementation

```python
import fcntl, socket, struct, dweepy, time, platform, random

def getTemp():
    return random.randint(1,1000)

def getHumidity():
    return 10

def getOS():
    return platform.platform()

# from http://stackoverflow.com/questions/159137/getting-mac-address
def getHwAddr(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    info = fcntl.ioctl(s.fileno(), 0x8927,  struct.pack('256s', ifname[:15]))
    return ':'.join(['%02x' % ord(char) for char in info[18:24]])

def post(dic):
    thing = 'therapeutic-caption'
    print dweepy.dweet_for(thing, dic)

def getReadings():
    dict = {}
    dict["temperature"] = getTemp();
    dict["mac-address"] = getHwAddr('eth0')
    dict["humidity"] = getHumidity()
    dict["operating system"] = getOS()
    return dict

while True:
    dict = getReadings();
    post(dict)
    time.sleep(5)
```

The sample JavaScript provided on the Moodle gives a good basis for this but does not support multiple graphs or different graphing formats which you will need to implement. This can be done by using other features of the HighCharts library or through another library, if desired.

3 High-level Logic

Listing 2 provides the high-level logic of the upload script.

Listing 2: High-level pseudo-code for upload script

```
import all necessary packages
write data collection methods e.g. get_temp
while true
        get the latest readings
        post the data to the cloud
        sleep for X seconds
```

Listing 3 provides the high-level logic of the script which consumes and displays the data.

Listing 3: High-level pseudo-code for consuming script

```
while true
        download the data
        cache the data
        draw the graphs for each data type
        set timer to poll again
```

4 Recommendations

- 1 method per sensor e.g. getTemp(), getHumidity(), getLEDState(), getNoiseLevel()
- Use fake data initially this allows you to develop the script on your PC, verify that it can submit to dweet and that the data can be retrieved. For example getTemp() would return a random number.
- When doing it for real utilize the sample code given by Grove and adjust it to your needs. As such you will need to review the scripts provided from the GrovePi repository. For example the LCD tutorial has code to retrieve values from DHT, take this and extract it into a set of methods.

5 Submission

Submission is via Moodle. Submit the code, a video of the running system, and a 1-page report identifying what you did, and any major design decisions. Refer to the Table 1 for details on my expectations.

	>=70%	**>=60%**	**>=50%**	**>=40**	**Fail**
Sensor Data Collection Script	Script has functionality beyond that which was explicitly specified. All code is well documented and written.	Script has all functionality which was explicitly specified. All code is well documented and written.	Majority of the features are implemented, there are comments, it is clear how the learner would have approached the remainder of the problem	Some of the script is functional and documented.	Script is largely non-functional or has not been demonstrated as such
HTML 5 + JavaScript Sensor Data Viewer	Script has functionality beyond that which was explicitly specified. All code is well documented and written.	Script has all functionality which was explicitly specified. All code is well documented and written.	Majority of the features are implemented, there are comments, it is clear how the learner would have approached the remainder of the problem	Some of the script is functional and documented.	Script is largely non-functional or has not been demonstrated as such
Report + Video	Succinct document and video with no superfluous content and no time-wasting, the work which is beyond the scope is clearly documented	Document is in good shape, wasted words are at a minimum. Video tells me what I need to know.	Document is average, contains necessary information.	Report is poor. Still difficult to parse, has a poor structure. Video is of low quality	Report is of a very poor standard, difficult to read, sentences make no sense, spelling and grammar are woeful. Video is useless, it omits most information needed by the marker.

Table 1: Marking Scheme