CM2110 Internet of Things – Topic 4

Python Modules

In this session, you'll improve the coding standards of the door / window sensor that you developed last week.

By the end of these exercises, you'll

- Increased your knowledge of creating and using modules in Python.
- Improved your software development practices.
- Be able to develop code implementing the logic of IoT devices that can be reused by yourself and others on hundreds of different devices.

Getting setup

As with last week, get your RPi connected to the lab equipment and booted up.

Reminder: if the resolution of the RPi is very low, then reboot it – this is likely because it was powered up before the monitor cable was attached, and so it did not know what resolution to use, so used the lowest to save resources.

At the end of each lab, remember to unplug from the power, as well as to reinstall the Ethernet cable back into the desktop computer, along with any monitor cables that you had to remove.

Cloning from GitHub

You have an option here – you can continue developing your solution from last week, or you can clone the provided GitHub repository to give you some starter code.

Getting setup / starter code

The start code has an initial template for a Python module called sensors, within it is a Sensor class based on the one discussed in the lecture. If you're extending your code from last time, then you should create the following structures:

- Directory called **sensors**
  - Within that, a file called __init__.py which has no content – just an empty text file, and a
  - Python class called Sensor.py – you can download this from the Moodle page
- At the same level, a __init__.py file which has no content
- A door_sensor_test.py file which contains some code to test your solution.

Your Task

Your task is to create a DoorWindowSensor class in Python, that

- extends the Sensor class
- includes a __init__ method, which receives the id of the sensor as a parameter
- includes a get_id() method, which returns the id of the sensor
- includes a calc_mag_field() method which calculates and returns the strength of the magnetic field – you should have implemented this last week.
- includes a calibrate() method, which sets a:
    - baseline variable to the strength of the current magnetic field
    - current_state variable to "OPEN"
- includes an observe() method, which performs the logic of detecting when the door/window changes state (i.e. opens/closes) and prints out a message (which should include all the information about feature of interest, observed property, etc. from last week)
    - Don't worry that you only printout the message – we'll send it somewhere next week.

Create and/or run the door_sensor_test.py script, which does something similar to the following:

```
# import the DoorWindowSensor

from sensors import DoorWindowSensor

# Create a new one

door_sensor = DoorWindowSensor.DoorWindowSensor()

# calibrate it

door_sensor.calibrate()

# start observing

door_sensor.observe()
```

Extend the Sensor Module

Now that you have created a class for the door / window sensor, start adding additional classes to the module that utilise either the sensors on the Sense HAT, or use some of the sensors available from the School IT office, which includes:

- Ultrasonic Distance sensor, which can be used following https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/
- PIR sensor, https://projects.raspberrypi.org/en/projects/physical-computing/11
- DHT-22 temperature and humidity sensor - https://www.instructables.com/Raspberry-Pi-Tutorial-How-to-Use-the-DHT-22/
- KY-018 Photo resistor - https://notenoughtech.com/raspberry-pi/light-sensor-lm393-ky018/
- Magnetometer HMC5883L - https://www.electronicwings.com/raspberry-pi/triple-axis-magnetometer-hmc5883l-interfacing-with-raspberry-pi
- Force sensitive resistor - https://learn.adafruit.com/force-sensitive-resistor-fsr