GrovePI and Raspberry PI Worksheet 2

What are we doing:

This worksheet is an introduction to using sensor data from the GrovePI boards in order to sense things about a situation.

You will need:

- Raspberry PI, Grovepi board and power.
- Grove LCD display (so you can see the board's network address)
- Either an ultrasonic distance sensor or an infrared motion sensor(PIR)
- A computer with Grovepi Emulator running.

We are going to use this equipment to build a simple system to detect and count people walking past the system. Such systems are used for example in museums to calculate how many people are visiting an exhibition and in secure settings to count how many people are still in an area.

Stage 1: Build a data capture python script

1) In the grovepi emulator, do **file, new**, then add the sensor you are using to digital pin 2.
2) Connect the real sensor to digital pin 2 on your real grovepi.
3) Go to **file, Write python for current sensors** This will save a python script to read data from the current sensor setup and output as CSV. Make sure you know where you saved this, you'll need it later. Saving this python script will load it as the currently running script in the emulator.
4) Hit '**run in emulator**' and check in the emulator console window, you should see something like

```
timestamp,button2
12121.0,1
12111.9,0
...
```

5) Click **run on real pi**, and enter *g54mrt@<address>* where <address> is the address shown on your grovepi screen (e.g. 10.154.188.14)
6) You should see a similar output in the console window, this time it is running on the real grovepi. You can hit stop and play buttons in the middle bottom of the emulator window to reset the script. Play with the sensor and see how the values change – for example if you wave in front of the PIR, then it should go to 1 for a bit, then back to 0, or if you put a hand in front of the ultrasonic ranger, the value should decrease to roughly the distance your hand is away from the sensor front in cm.

Stage 2: Collect some data

1) Think of a few situations that are likely to happen in real world deployment of this sensor. For example: Nobody walks past. 1 person walks past. 2 people walk past together. Someone hangs around the sensor without walking off.

2) For each of this scenarios, in the emulator, click 'capture script to file', and select a memorable name for the data file you output e.g. 1_person_walks_past.csv.
3) Act out the scenario (e.g. actually walk past the sensor).
4) When you finish acting it out, click 'stop' to save the script.
5) Do this for all your scenarios.

Stage 3: Build an algorithm

1) Look at your data in excel (or a text editor if you prefer). Think about what happens to the sensor when someone walks past.
2) Open the python file you saved for data capture and save it under a new name, this will be your sensing algorithm.
3) Alter the python to generate a further column 'numpeople', showing how many people you think have gone through. As an example, the below code will do a very poor attempt at counting people.

```
import time
import grovepi
# show a header line so that this is a readable CSV
print("timestamp,pir2,numPeople")
numPeople=0

while True:
    # read all the sensors
    timestamp=time.time()
    pir2=grovepi.digitalRead(2)

    # output a line of text, separated by commas
    print("%f,%d,%d"%(timestamp,pir2,numPeople))

    # sensed something on the PIR, add a person
    if pir2 == 1:
        numPeople+=1

    # wait for 0.1 second - change this to alter how quickly
    # sensor data is collected
    time.sleep(0.1)
```

4) Try and make your code so that it at least works okay if one person walks past the sensor with a big time gap between people.

Stage 4: Test your algorithm

1) Record a new set of test data in which you walk past multiple times, multiple people walk past at points. Keep a note of what you did when, and how many people had walked past at each point.
2) Run your script using this test data in the emulator, by clicking '**replay File**' in the bottom right of the window. Record the output to csv again.
3) Look at the output csv file. For each event (person walks past), check a)whether your algorithm detected an event, and if applicable b)whether it detected the right number of

people. Calculate what percentage of events were correctly detected (These are called **true positives**).

4) Also look at how many times did your program report an event when it should not have done (e.g. my code above will correctly detect 1 person walking in, but then will think they are followed by about 5 people). These are called **false positives.**
5) Then look at each of the failures of the system, look at the underlying sensor data, and consider how you might change the system to make it better.