SIT107 - Software Engineering 1: Connecting The Cyber And Physical Worlds

Saving Motion Data Activity Sheet

Hardware Required

- Arduino Board
- USB cable
- HCSR505 PIR Passive Infra Red Motion Detector (https://tronixlabs.com.au/sensors/motion/hcsr505-pir-passive-infra-red-motion-detector-australia/)
- Male to Female Dupont Jumper Wires

Pre-requisites: You must do the following before this task

Read this sheet from top to bottom

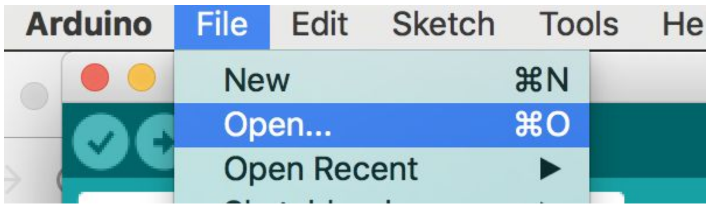
Steps

1. Attach the PIR motion sensor to the Arduino board as you did in Task 2.2P. Refer to "Sensing Motion Activity Sheet" if you need step-by-step instructions. But do not write any code yet!

2. Open your Arduino IDE

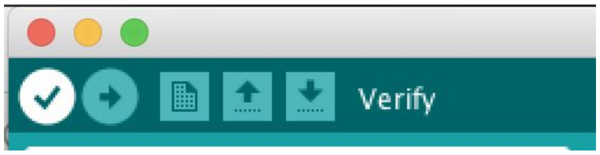
3. Download the code given in https://github.com/niroshini/motion_data_logging . If you download it in .zip format, you must extract it to a location on your computer after you download it.

4. Go to the Arduino IDE. Select "File -> Open" and it will open a dialog box.

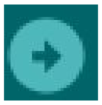


5. Select the motion_data_logging.ino file inside the motion_data_logging folder & click Open.

6. In your Arduino IDE, click on the 'Verify' button. This will check for errors and compile your code.



7. Use the "tools" menu to check the port to which the Arduino board is connected. Once the right port is selected, it is possible to upload sketches to Arduino.

8. Now click the upload icon to upload the code to the Arduino board. If you get an error, check to be sure you've selected the correct device and port. 

9. Open Serial Monitor and observe the output.

What does this code do?

Phew! This week's code is fairly long compared to last two weeks.
Don't worry if all the lines of code do not make much sense right now. At this point, you just need to have an overall understanding of what the code is doing.

This week, we have a sensor reading values and writing it to a .csv file on the SD card attached to the Arduino board.
The SD card is already attached to the board using what we call a 'shield'. If you need further info about shields, refer to week 3 lecture and also have a look here:
https://learn.adafruit.com/adafruit-data-logger-shield/overview .

Let's dissect the program to understand what happens inside.

1. Firstly, the code needs to make sure that the SD card is properly attached. This is what's happening in
   `initSDcard();` in the code. We need this because we are going to save the motion data to SDCard.

2. Next, the code tries to create an empty .csv file named 'MLOG00.CSV'. We do a little trick here, we basically want the files to be called something like 'MLOG00***nn***.csv where ***nn*** is a number. So for example, the very first time we run this code, we want the motion values to be written to 'MLOG00.CSV. The next time, to be written to 'MLOG01.CSV, and then to be written to 'MLOG02.CSV and so on. This is what is happening in `createFile();` in the code.

3. You may be wondering what a .CSV file is! CSV stands for comma separated value file. They provide a simple way to save data in a table structured format. CSV files can be read by lots of different applications. You can use Excel to open .CSV files.

4. Next, `initRTC();` checks if the RTC (Real Time Clock) is working properly. This is needed because we need the date and time for each motion data that is saved.

5. The next step is to create the 'headers' in the .csv file. In this example, the headers are: millis,stamp,datetime,motion. What we are going to do is to store our data similar to a table like so:

millis	stamp	datetime	motion
999	1497825202	2017/6/18 22:33:22	Active

1999	1497825203	2017/6/18 22:33:23	Inactive
2999	1497825204	2017/6/18 22:33:24	Active
3999	1497825205	2017/6/18 22:33:25	Inactive

Only, in the .csv file, we will write these values separated by commas, and not inside a table.
For example, in the .csv file, we will write the above data as:

```
millis,stamp,datetime,motion
999,1497825202, 2017/6/18 22:33:22,Active
1999,1497825203,2017/6/18 22:33:23,Inactive
2999,1497825204,2017/6/18 22:33:24,Active
3999,1497825205,2017/6/18 22:33:25,Inactive
```

6. `pinMode(6, INPUT);`

In this line of code, we configure Arduino to read digital pin number 6. This is what the motion sensor is connected to.

7. Now the setup part is done, the code enters the loop. This is the part that will be executed over and over. Here, we first take a snapshot of the current time in `now = RTC.now();` This is what we use to get the date time information that will be logged.

```
logfile.print(now.unixtime());
logfile.print(", ");
logfile.print(now.year(), DEC);
logfile.print("/");
logfile.print(now.month(), DEC);
logfile.print("/");
logfile.print(now.day(), DEC);
logfile.print(" ");
logfile.print(now.hour(), DEC);
logfile.print(":");
logfile.print(now.minute(), DEC);
logfile.print(":");
logfile.print(now.second(), DEC);
```

8. The next part is reading motion sensor data and saving it to the logfile:

```
//Read data and store it to variable
  if (digitalRead(6) == HIGH) {
    //    Serial.println("Active");
    state = activeMotion;
  }
  else {
    //    Serial.println("Inactive");
    state = inactiveMotion;
  }

  logfile.print(", ");
  logfile.println(state);
```

9. Finally, we call logfile.flush(); to ensure the file is properly saved to the SD card.

References

https://github.com/adafruit/RTClib
https://learn.adafruit.com/adafruit-data-logger-shield/using-the-real-time-clock-3
http://www.instructables.com/id/How-to-use-DHT-22-sensor-Arduino-Tutorial/
https://plot.ly/arduino/dht22-temperature-tutorial/
http://cactus.io/hookups/sensors/temperature-humidity/dht22/hookup-arduino-to-dht22-temp-humidity-sensor