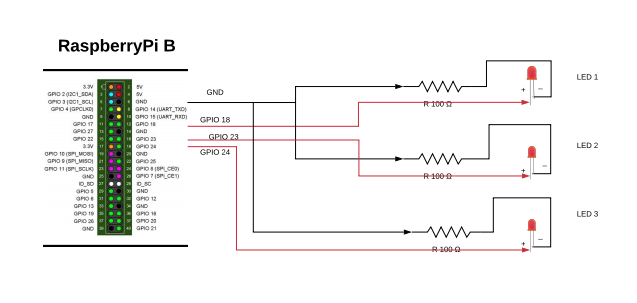
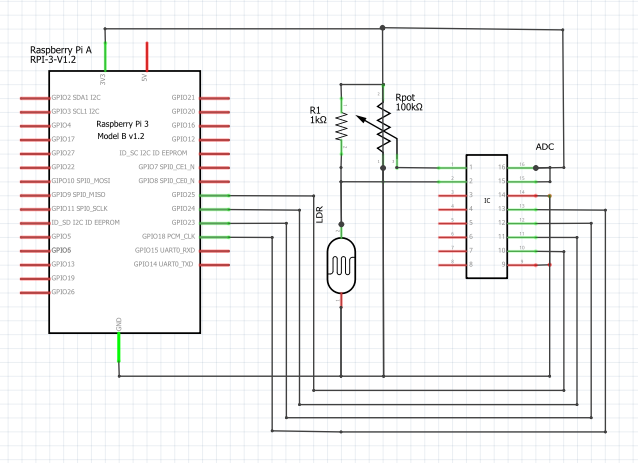
1. Homework 2 Writeup
   1. Names of each team member and percentage contribution of each team member. Please be fair and truthful. The grade of this assignment will be divided among team members according to their percentage contributions
      1. **Graham Flinchum - 33.3%**
      2. **Leonardo Falcon Izzi - 33.3%**
      3. **Sweta Rout - 33.3%**
   2. What exactly did each team member do? Make a table. Assuming you have 4 members in your team, make a table with 5 columns. The first column should state various subtasks that your team undertook for this assignment and the next four columns should state the percentage contribution of the four team members for each subtask.

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Graham | Leonardo | Sweta |
| Getting an MQTT broker running |  | 100% |  |
| Coding Raspberry Pi A | 100% |  |  |
| Coding Raspberry Pi B |  |  | 100% |
| Coding Raspberry Pi C |  | 100% |  |
| Setting up breadboard with ADC, LDR, and potentiometer. | 100% |  |  |
| Getting laptop 2 to log and display messages correctly |  |  | 100% |

* 1. Schematics diagram of connection of LEDs to Raspberry Pi B****
  2. Schematics diagram of connection of LDR and potentiometer to Raspberry Pi A

****

* 1. Describing all your design choices such as which MQTT broker did you implement along with a step by step instruction to install it on a laptop/computer, what frequency did you sample the ADC at, how did you scale values from potentiometer/LDR before posting to their corresponding topics so that they could be compared by Raspberry Pi C, what was the range of raw values (min and max) that your ADC got from the LDR, what are the range of raw values that your ADC got from potentiometer (min and max), what are the range of scaled values (min and max) that resulted after you scaled the values from the potentiometer and/or LDR, etc. The text for this part should be no more than 1 page, double column format, 10pt font size.

BROKER

The MQTT broker we chose is ‘mosquitto’ because it is a lightweight protocol and is suitable to be used for low power single board computers to full servers. The mosquitto\_pub and mosquitto\_sub commands are easy to use and can be directly used from the command line. Moreover, it is portable as well as available for a wide range of platforms. Mosquitto was also chosen because of its features - it supports last-will message, message persistence, quality of service level 2. All of which were required to fulfill the requirements of the assignment.

*Installing Mosquitto on a Windows machine*

Step1 : First download mosquitto from <https://mosquitto.org/download/>

Step2 : Install mosquitto. After installation, there will be error or warning messages like, “missing dll”.

Step3 : Download the additional files as mentioned in <https://sivatechworld.wordpress.com/2015/06/11/step-by-step-installing-and-configuring-mosquitto-with-windows-7/> and place all the files in the mosquitto folder. In my case it is, C:\Program Files (x86)\mosquitto

Step4 : Install mosquitto again

Step5 : Mosquitto is successfully installed and can be started by opening a command prompt and go to the location C:\Program Files (x86)\mosquitto and type mosquitto to run the broker

We followed the link <http://www.steves-internet-guide.com/install-mosquitto-broker/> to install mosquitto.

CLIENTS

We chose to use the Paho Python MQTT API for the raspberry PIs. It provides a simple interface for requesting last-will messages, publishing messages with persistence, and automatic reconnects.

*Installing Paho Python Client on Raspberry Pi*

The Paho Python Client provides a client class with support for MQTT on Python.

Step1: pip install paho-mqtt

Step2:git clone https://github.com/eclipse/paho.mqtt.python.git  
cd paho.mqtt.python  
python setup.py install

We followed the link <https://www.eclipse.org/paho/clients/python/#>

Once paho is installed on raspberry pi, we just have to

import paho.mqtt.client as mqtt to

And then we can easily connect our raspberry pi to the laptop using mqtt.

RASPBERRY PI C

The major design challenge here was to deal with buffered threshold and ldr values. This would occur when raspberry PI C had been disconnected, but Raspberry PI A was still sending updated values. On the reconnect call back of raspberry PI C, the broker would flood it with threshold and ldr values. Because of this, the decision to Turn On and Turn Off the LED was moved to an asynchronous task running every 1 second. This way multiple Turn On and Turn Off commands were not unnecessarily published.

ADC

When using the ADC, we sampled the values from the LDR and potentiometer every 10 ms. The potentiometer gave us values from 0-1023, while the LDR gave us values from about 900-1000. In order to normalize the data from the two, I divided the value from the potentiometer by 10, and subtracted 900 from the LDR value in order to give them both a range of approximately 0-100.