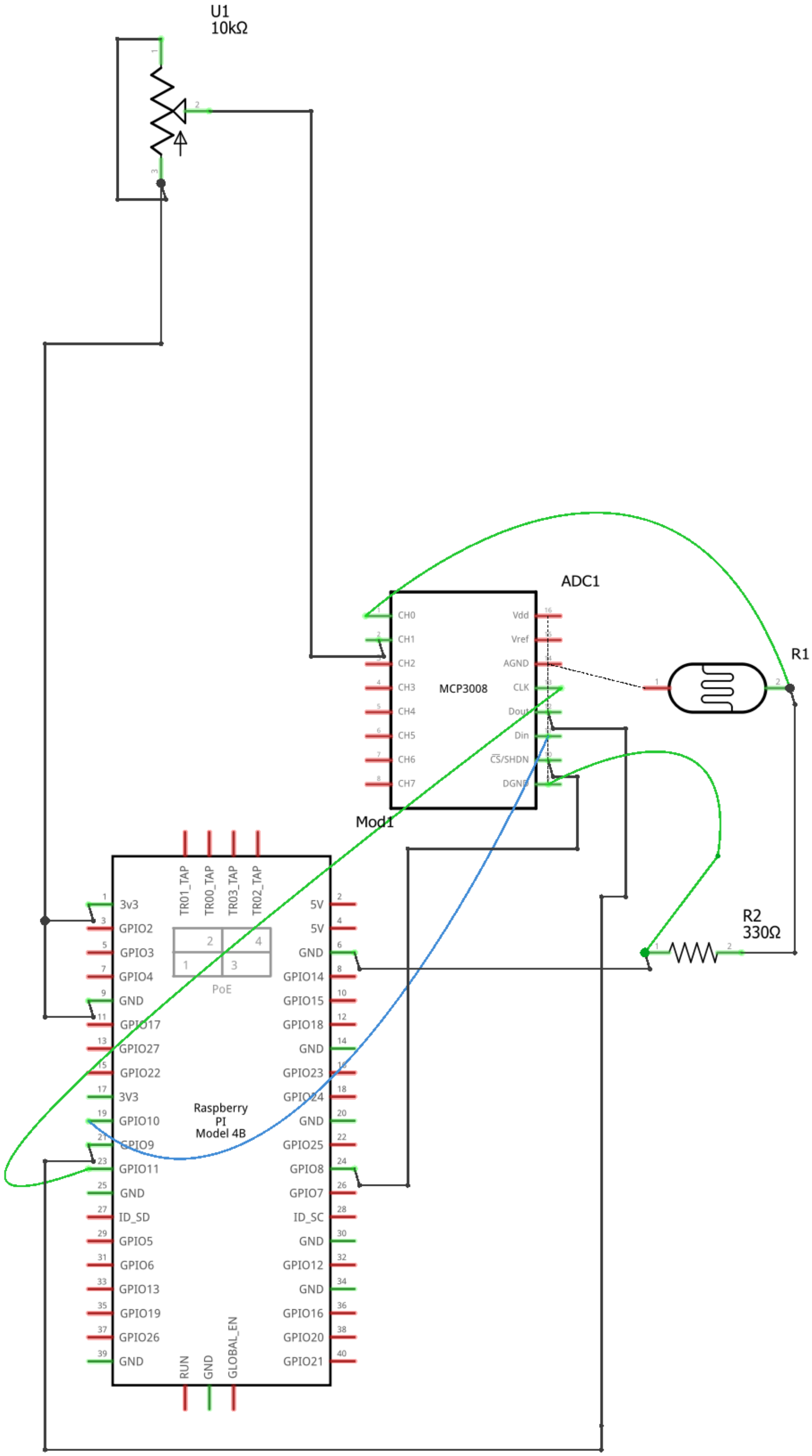
Percentage Contribution

- Dinesh - 33%
- Pratik - 34%
- Siddhant - 33%

Contribution Table

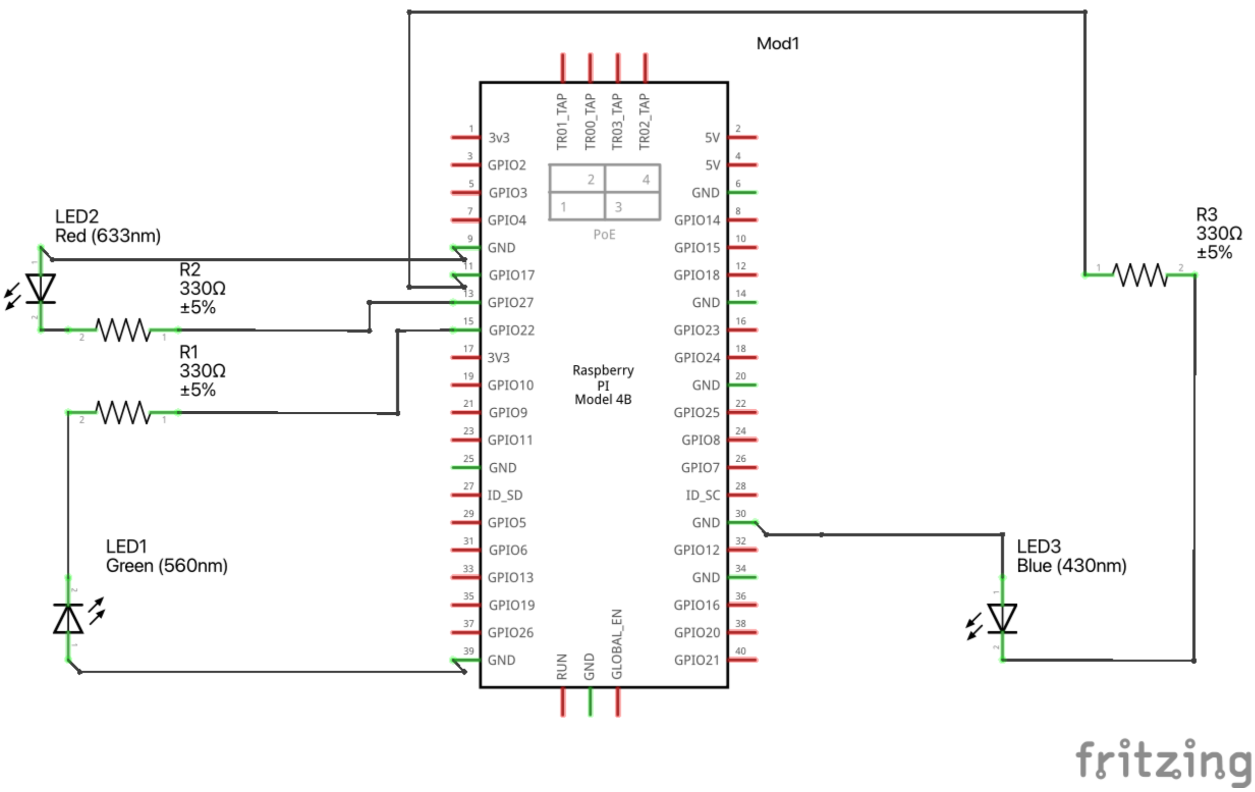
Task	**Dinesh**	**Pratik**	**Siddhant**
Installing & configuring MQTT broker (Laptop 1)	0%	100%	0%
Coding Raspberry Pi A (LDR + Potentiometer + ADC)	0%	100%	0%
Implementing the Raspberry Pi A Circuit	33%	33%	33%
Coding Raspberry Pi B (LED logic & status updates)	100%	0%	0%
Implementing the Raspberry Pi B Circuit	33%	33%	33%
Coding Raspberry Pi C (comparison & LightStatus logic)	0%	0%	100%
Laptop 2/Smartphone logging & timestamp recording	33%	34%	33%
Schematics (connections of LEDs, LDR, potentiometer)	50%	0%	50%
Documentation (design choices, instructions)	33%	34%	33%
Testing & Demo preparation	33%	34%	33%

Schematic Diagram Of Raspberry Pi A


U1
10kΩ
ADC1
MCP3008
CH0
CH1
CH2
CH3
CH4
CH5
CH6
CH7
Vdd
Vref
AGND
CLK
Dout
Din
CS/SHDN
DGND
R1
R2
330Ω
Mod1
Raspberry PI Model 4B
TR01_TAP
TR00_TAP
TR03_TAP
TR02_TAP
PoE
3v3
GPIO2
GPIO3
GPIO4
GND
GPIO17
GPIO27
GPIO22
3V3
GPIO10
GPIO9
GPIO11
GND
ID_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
5V
5V
GND
GPIO14
GPIO15
GPIO18
GND
GPIO23
GPIO24
GND
GPIO25
GPIO8
GPIO7
ID_SC
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21
RUN
GND
GLOBAL_EN

fritzing

Schematic Diagram Of Raspberry Pi B



Description

For the MQTT broker, we decided to use Mosquitto since it is an open-source message broker that implements the MQTT protocol. The broker enables communication between clients and IoT devices via MQTT. It receives messages from publishers, verifies their publishing rights, and queues the messages according to their Quality of Service (QoS) levels.

To install the Mosquitto broker and clients on a Linux-based system, we run:

sudo apt update

sudo apt install mosquitto mosquitto-clients

After installation, the broker service can be started with:

sudo systemctl start mosquitto

To verify that the broker is running:

sudo systemctl status mosquitto

For effective ADC sampling, we used time.sleep(0.1), which corresponds to 10 Hz (every 100 milliseconds). During each sampling cycle, we read both the LDR sensor on channel 0 and the potentiometer on channel 1.

To scale the values from both the potentiometer and LDR, we defined:

minimum_ldr = 10, maximum_ldr = 100

minimum_potentiometer = 90, maximum_potentiometer = 250

We then normalized the raw ADC values using the formula:

Normalized = (value − min_val) / (max_val − min_val)

and constrained the result between 0.0 and 1.0:

max(0.0, min(1.0, normalized))

This ensures that the scaled values are always in the range [0.00, 1.00]. Alternatively, functions such as the sigmoid function could also be used for normalization.