Course: EGDF20 Diploma in Electronic and Computer Engineering

Module: EGE356 IoT System Architecture & Technology

Lab 3: Edge Device to Cloud – Assignment 1

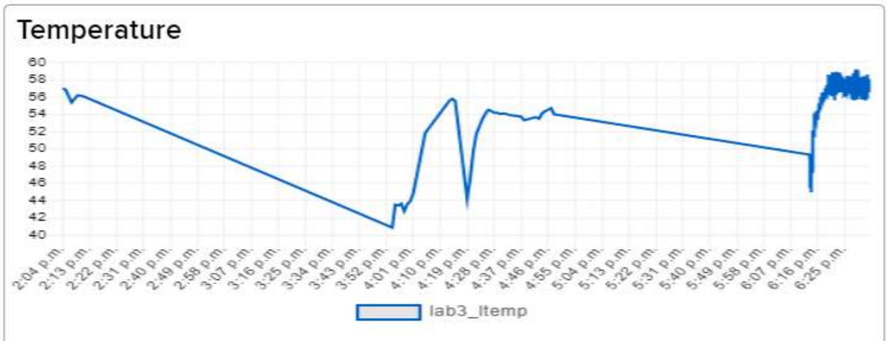
Objectives:

1. Create an Endpoint Temperature to Dashboard Line Chart (40 marks)
2. Build an Alert and Control System to Alert and Auto-Set PWM Control when Temperature exceeds Threshold level. (35 marks)
3. Create a Text Block UI to enable setting of Temperature Threshold setting for User (25 marks)

Part 1: Create an Endpoint Temperature to Dashboard Line Chart

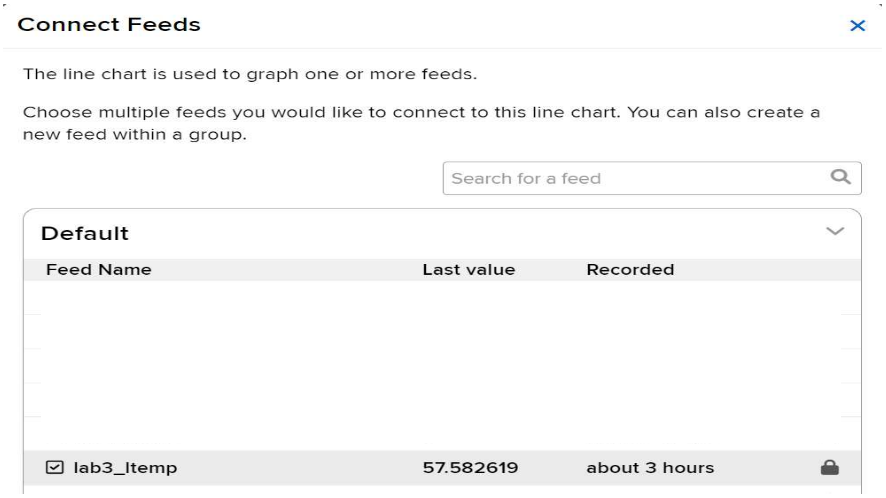
1. Use the ege356_lab2_led.ino file from Lab 2, and save it as ege356_lab3_assn1.ino.

2. Create a new Line Chart Block as shown below. The line chart should display the temperature reading from the M5StickC device's IMU.



3. The following Feed should be created and used for the Line Chart Block

AdafruitIO_Feed *analog_ltemp = io.feed("lab3_ltemp");



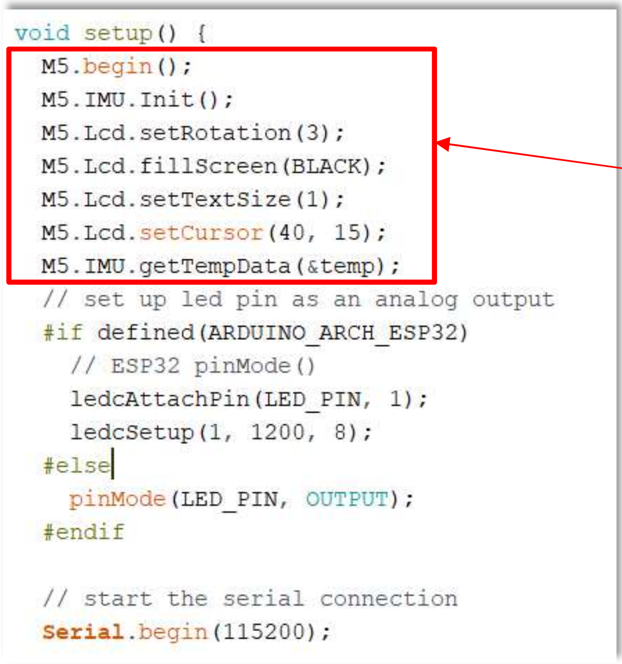
Connect Feeds ×

The line chart is used to graph one or more feeds.

Choose multiple feeds you would like to connect to this line chart. You can also create a new feed within a group.

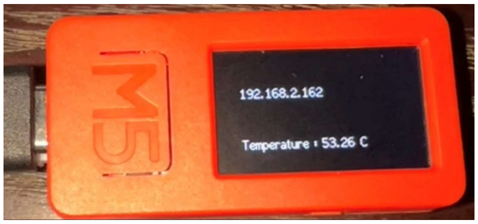
Feed Name	Last value	Recorded
☑ lab3_ltemp	57.582619	about 3 hours

4. Add the following codes to **void setup()** to access the M5StickC IMU to get the system's internal temperature data and display to the TFT screen.



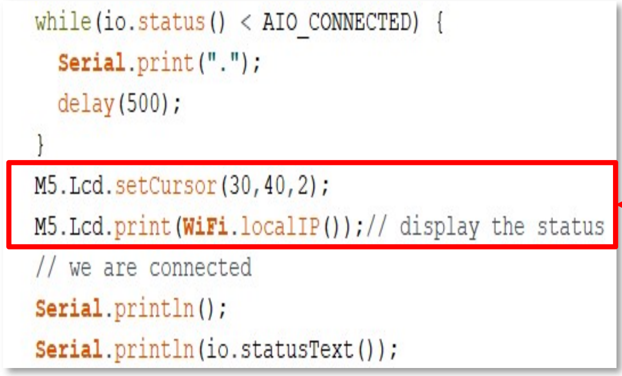
```
void setup() {
  M5.begin();
  M5.IMU.Init();
  M5.Lcd.setRotation(3);
  M5.Lcd.fillScreen(BLACK);
  M5.Lcd.setTextSize(1);
  M5.Lcd.setCursor(40, 15);
  M5.IMU.getTempData(&temp);
  // set up led pin as an analog output
  #if defined(ARDUINO_ARCH_ESP32)
    // ESP32 pinMode()
    ledcAttachPin(LED_PIN, 1);
    ledcSetup(1, 1200, 8);
  #else
    pinMode(LED_PIN, OUTPUT);
  #endif

  // start the serial connection
  Serial.begin(115200);
```

```
M5.begin();
M5.IMU.Init();
M5.Lcd.setRotation(3);
M5.Lcd.fillScreen(BLACK);
M5.Lcd.setTextSize(1);
M5.Lcd.setCursor(40, 15);
M5.IMU.getTempData(&temp);
```



. . . .

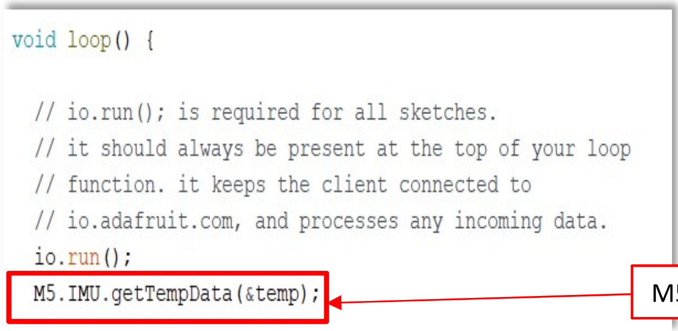
. . . .



```
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  M5.Lcd.setCursor(30,40,2);
  M5.Lcd.print(WiFi.localIP());// display the status
  // we are connected
  Serial.println();
  Serial.println(io.statusText());
```

```
M5.Lcd.setCursor(30,40,2);
M5.Lcd.print(WiFi.localIP());
```

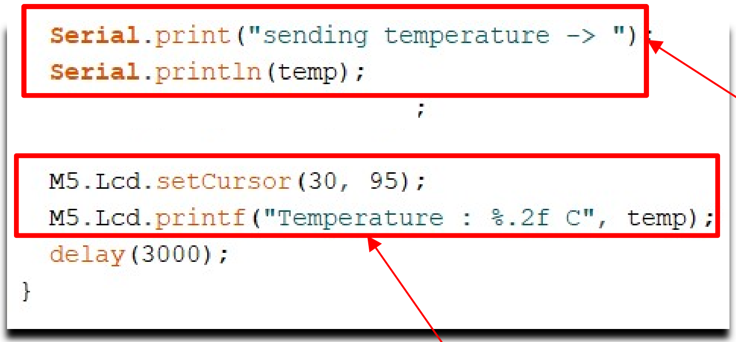
Add the following code to void loop() so that the M5StickC will constantly display the updated temperature.



```
void loop() {

  // io.run(); is required for all sketches.
  // it should always be present at the top of your loop
  // function. it keeps the client connected to
  // io.adafruit.com, and processes any incoming data.
  io.run();
  M5.IMU.getTempData(&temp);
```

Get temperature data from IMU and store in 'temp'.

```
M5.IMU.getTempData(&temp);
```



```
  Serial.print("sending temperature -> ");
  Serial.println(temp);
                              ;

  M5.Lcd.setCursor(30, 95);
  M5.Lcd.printf("Temperature : %.2f C", temp);
  delay(3000);
}
```

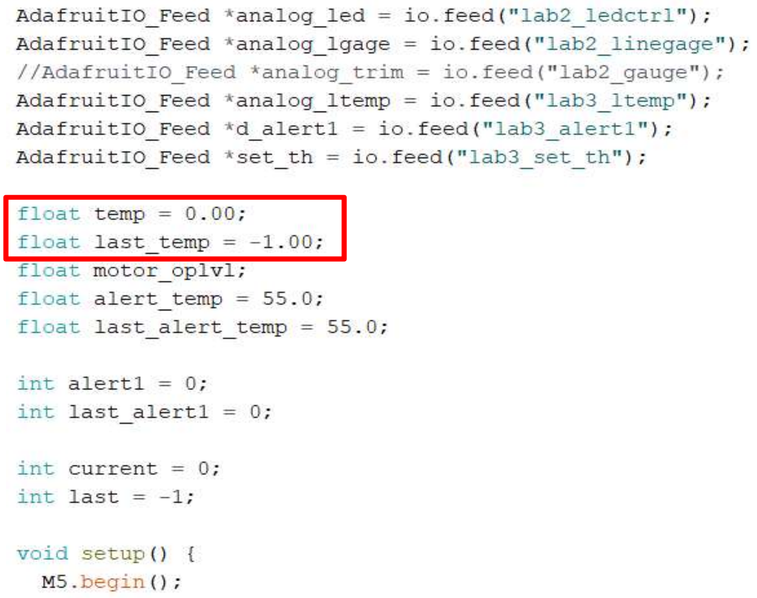
Serial output displays temperature data.

Serial.print("sending temperature -> ");
Serial.println(temp);

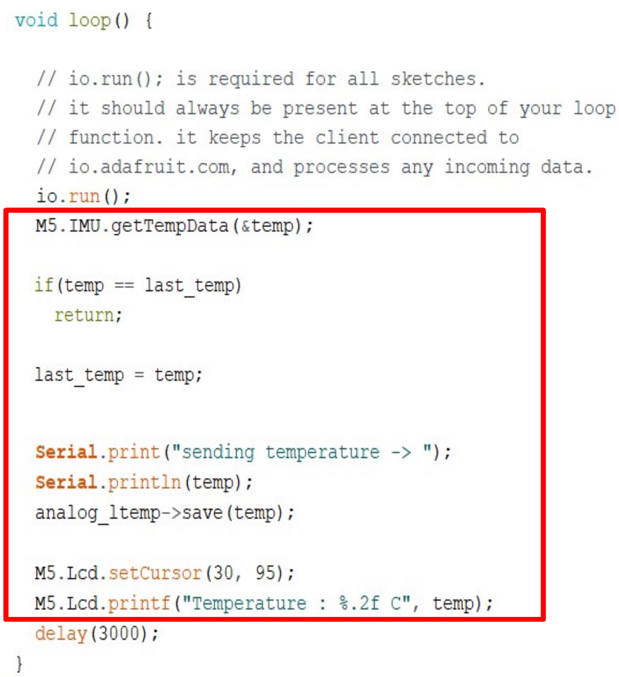
M5.Lcd.setCursor(30, 95);
M5.Lcd.printf("Temperature : %.2f C", temp);

M5StickC display screen displays the temperature data.

5. Refer to codes in Lab 1 and Lab 2, and make the necessary modification to the the codes in **void loop()** to capture the temperature data and send the data to the Adafruit Feed. Note that the temperature data should be updated to the cloud only when the value is different from the last data. This is to preserve the bandwidth.



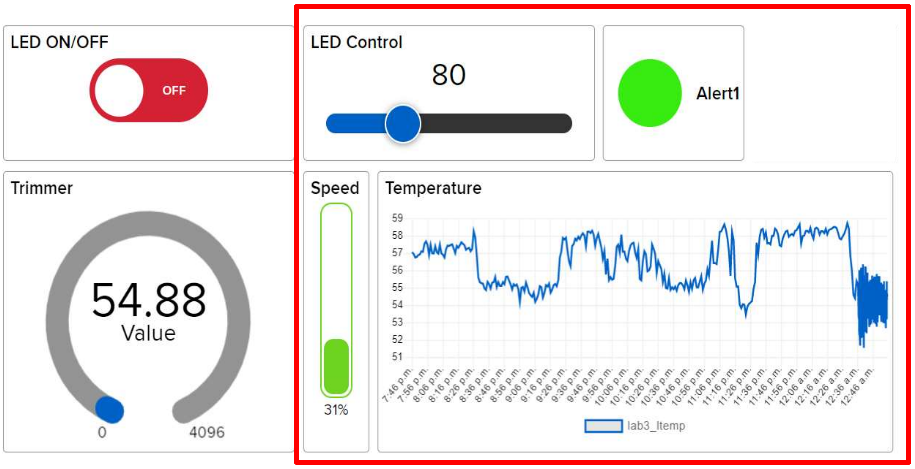
```
AdafruitIO_Feed *analog_led = io.feed("lab2_ledctrl");
AdafruitIO_Feed *analog_lgage = io.feed("lab2_linegage");
//AdafruitIO_Feed *analog_trim = io.feed("lab2_gauge");
AdafruitIO_Feed *analog_ltemp = io.feed("lab3_ltemp");
AdafruitIO_Feed *d_alert1 = io.feed("lab3_alert1");
AdafruitIO_Feed *set_th = io.feed("lab3_set_th");

float temp = 0.00;
float last_temp = -1.00;
float motor_oplvl;
float alert_temp = 55.0;
float last_alert_temp = 55.0;

int alert1 = 0;
int last_alert1 = 0;

int current = 0;
int last = -1;

void setup() {
  M5.begin();
```



```
void loop() {

  // io.run(); is required for all sketches.
  // it should always be present at the top of your loop
  // function. it keeps the client connected to
  // io.adafruit.com, and processes any incoming data.
  io.run();
  M5.IMU.getTempData(&temp);

  if(temp == last_temp)
    return;

  last_temp = temp;


  Serial.print("sending temperature -> ");
  Serial.println(temp);
  analog_ltemp->save(temp);

  M5.Lcd.setCursor(30, 95);
  M5.Lcd.printf("Temperature : %.2f C", temp);
  delay(3000);
}
```

Part 2: Build an Alert and Control System to Alert and Auto-Set PWM Control when Temperature exceeds Threshold level

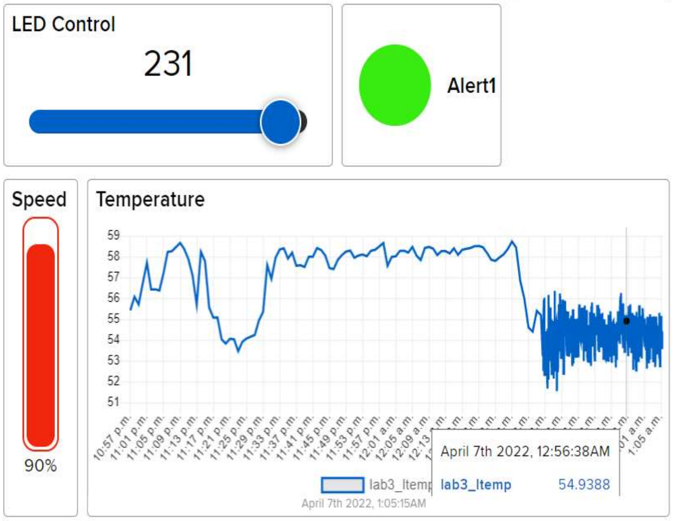
1. Make use of Labs 1, 2 and part 1 of this lab to create the Alert and Control System. This system should contain the following highlighted Blocks in the Dashboard.

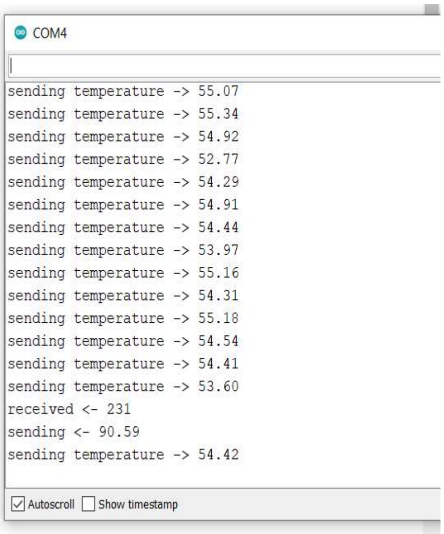


Under normal operations, the system Alert1 indicator will display Green. Note that the LED Control values are used to calculate the Speed gauge in %. The arbitrary calculation used is as shown:

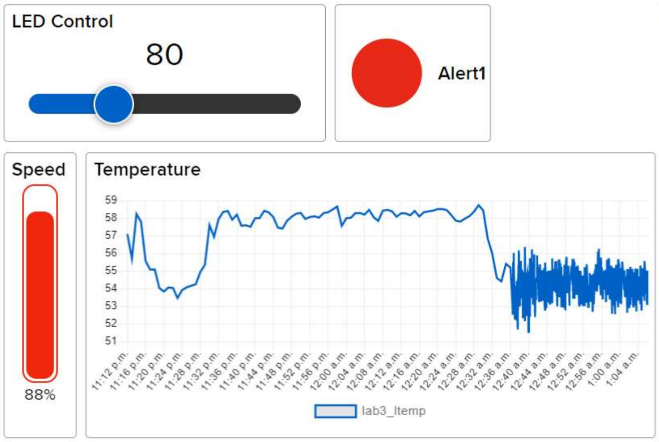
Speed (%) = (LED Control Value) / 255.0 * 100

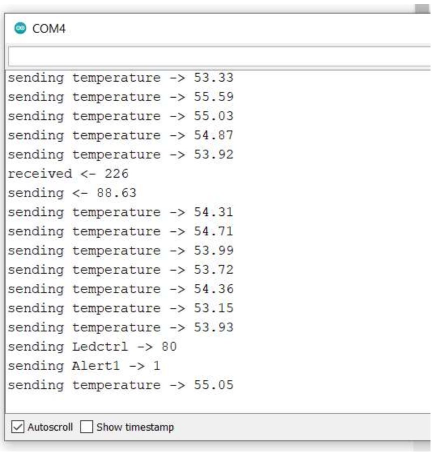
The system will continue to work normally (Alert1 is green). When the Speed % is > 80.0 and the temperature is above the threshold value of 55.0 °C (while Alert1 is still green), the system will trigger Alert1 (Alert1 turns red) and immediately set LED Control to 80. During this period, LED Control will remain at 80 until temperature lowers below the threshold temperature of 55.0 °C. This will then trigger Alert1 to turn Green. System is now back in normal operation and the LED Control slider value can be increased now.





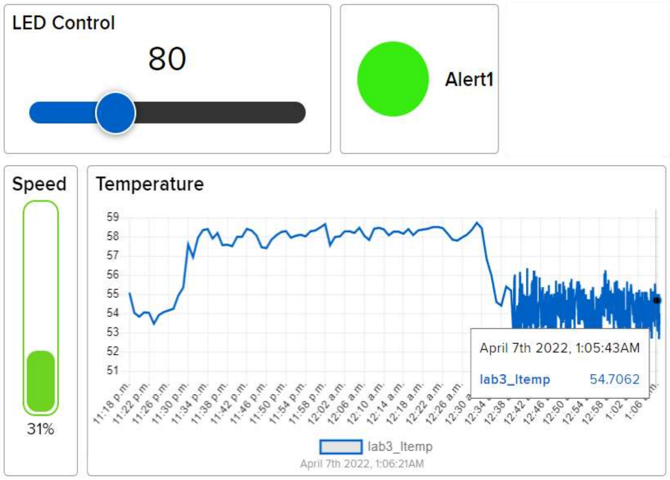
```
sending temperature -> 55.07
sending temperature -> 55.34
sending temperature -> 54.92
sending temperature -> 52.77
sending temperature -> 54.29
sending temperature -> 54.91
sending temperature -> 54.44
sending temperature -> 53.97
sending temperature -> 55.16
sending temperature -> 54.31
sending temperature -> 55.18
sending temperature -> 54.54
sending temperature -> 54.41
sending temperature -> 53.60
received <- 231
sending <- 90.59
sending temperature -> 54.42
```

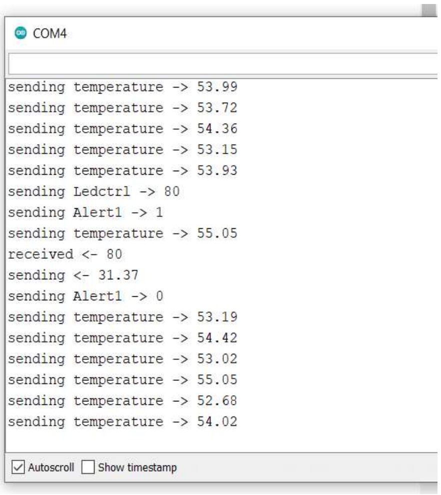




```
sending temperature -> 53.33
sending temperature -> 55.59
sending temperature -> 55.03
sending temperature -> 54.87
sending temperature -> 53.92
received <- 226
sending <- 88.63
sending temperature -> 54.31
sending temperature -> 54.71
sending temperature -> 53.99
sending temperature -> 53.72
sending temperature -> 54.36
sending temperature -> 53.15
sending temperature -> 53.93
sending Ledctrl -> 80
sending Alert1 -> 1
sending temperature -> 55.05
```

During this period when Alert1 is RED, the LED Control will remain at 80 until temperature lowers below the threshold temperature of 55.0 °C. User will not be able to adjust the LED Control slider above 80.0 during this period.





```
sending temperature -> 53.99
sending temperature -> 53.72
sending temperature -> 54.36
sending temperature -> 53.15
sending temperature -> 53.93
sending Ledctrl -> 80
sending Alert1 -> 1
sending temperature -> 55.05
received <- 80
sending <- 31.37
sending Alert1 -> 0
sending temperature -> 53.19
sending temperature -> 54.42
sending temperature -> 53.02
sending temperature -> 55.05
sending temperature -> 52.68
sending temperature -> 54.02
```

Once the Alert1 turns Green (when IMU temperature drops below the threshold, the system operates normally and the LED Control slider value can now be modified.

Things to Note:

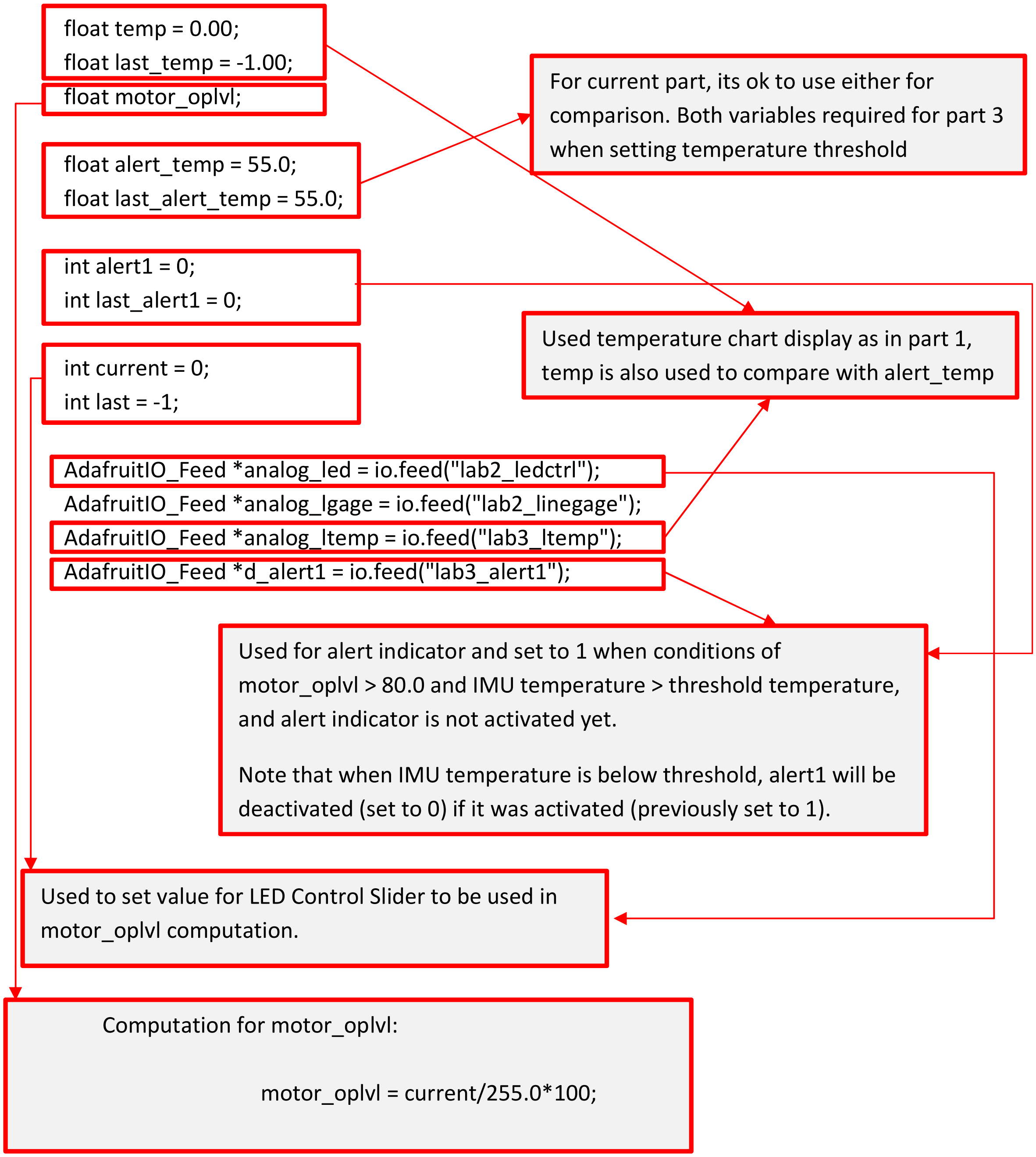
a. Temperature threshold value can be fixed in the program

```
Example:
float alert_temp = 55.0;
```

b. LED Control slider is set to 80.0. User may change the slider value to be above 80, but the system will check and revert it back to 80.0 as long as Alert1 is RED.

2. **Watch the video to understand how the program works.**
   https://1drv.ms/v/s!AjOmcshLwjW_gZzGNQRBuXDRopCZ52U?e=A82fvh

3. Hint: Here are some of the variables that can be used for the Arduino application



```
float temp = 0.00;
float last_temp = -1.00;
float motor_oplvl;

float alert_temp = 55.0;
float last_alert_temp = 55.0;

int alert1 = 0;
int last_alert1 = 0;

int current = 0;
int last = -1;

AdafruitIO_Feed *analog_led = io.feed("lab2_ledctrl");
AdafruitIO_Feed *analog_lgage = io.feed("lab2_linegage");
AdafruitIO_Feed *analog_ltemp = io.feed("lab3_ltemp");
AdafruitIO_Feed *d_alert1 = io.feed("lab3_alert1");
```

For current part, its ok to use either for comparison. Both variables required for part 3 when setting temperature threshold

Used temperature chart display as in part 1, temp is also used to compare with alert_temp

Used for alert indicator and set to 1 when conditions of motor_oplvl > 80.0 and IMU temperature > threshold temperature, and alert indicator is not activated yet.

Note that when IMU temperature is below threshold, alert1 will be deactivated (set to 0) if it was activated (previously set to 1).

Used to set value for LED Control Slider to be used in motor_oplvl computation.

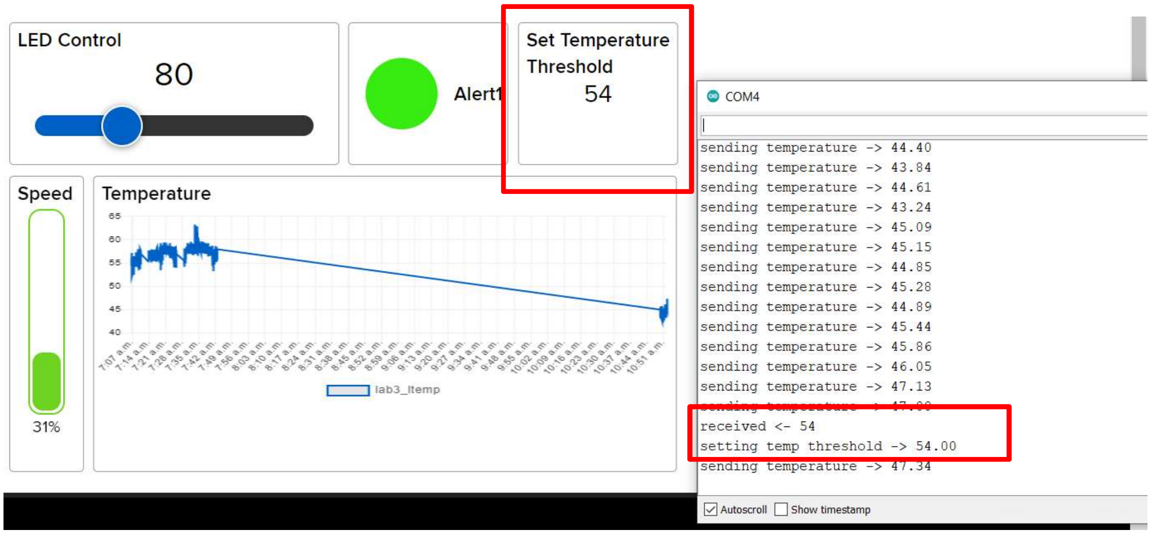
Computation for motor_oplvl:

```
motor_oplvl = current/255.0*100;
```

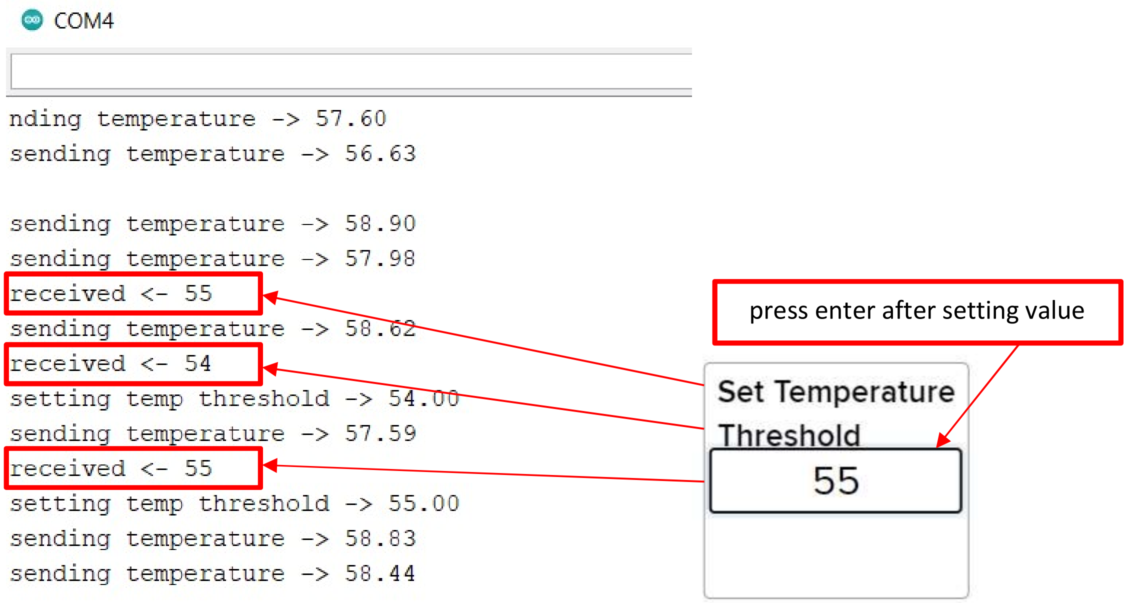
4. Items in 3 are hints. It is meant as a guide and not compulsory to follow. However, the explanations for the variables provided, are required conditions for the application to work.

Part 3: Create a Text Block UI to enable setting of Temperature Threshold setting for User

1. Create a Text Block for setting of Temperature Threshold. This is an enhancement to part 2, where the program sets the threshold value in the program.



```
sending temperature -> 44.40
sending temperature -> 43.84
sending temperature -> 44.61
sending temperature -> 43.24
sending temperature -> 45.09
sending temperature -> 45.15
sending temperature -> 44.85
sending temperature -> 45.28
sending temperature -> 44.89
sending temperature -> 45.44
sending temperature -> 45.86
sending temperature -> 46.05
sending temperature -> 47.13
sending temperature -> 47.00
received <- 54
setting temp threshold -> 54.00
sending temperature -> 47.34
```



```
nding temperature -> 57.60
sending temperature -> 56.63

sending temperature -> 58.90
sending temperature -> 57.98
received <- 55
sending temperature -> 58.62
received <- 54
setting temp threshold -> 54.00
sending temperature -> 57.59
received <- 55
setting temp threshold -> 55.00
sending temperature -> 58.83
sending temperature -> 58.44
```

press enter after setting value

2. Hint: Here are some of the variables and Feed that can be used for the Arduino application

```
float alert_temp = 55.0;
float last_alert_temp = 55.0;
```

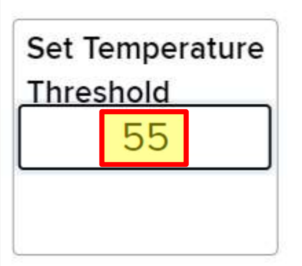
```
AdafruitIO_Feed *set_th = io.feed("lab3_set_th");
```

Values typed into the Text Block will be used to set the alert_temp.

```
set_th->onMessage(set_th_pt);
```

When new values are keyed into the Text Block, data is sent from cloud to device. Device on receiving the message will execute "set_th_pt" function.

```
void set_th_pt(AdafruitIO_Data *data) {

  // convert the data to integer
  int new_set_th = data->toInt();

  Serial.print("received <- ");
  . . . .
  . . . .

  Serial.print("setting temp threshold -> ");
  . . . .
  . . . .
}
```


Set Temperature Threshold

55

Data received from Text Block converted to integer

3. Items in 2 are hints. It is meant as a guide and not compulsory to follow. However, the explanations for the variables provided, are required conditions for the application to work.