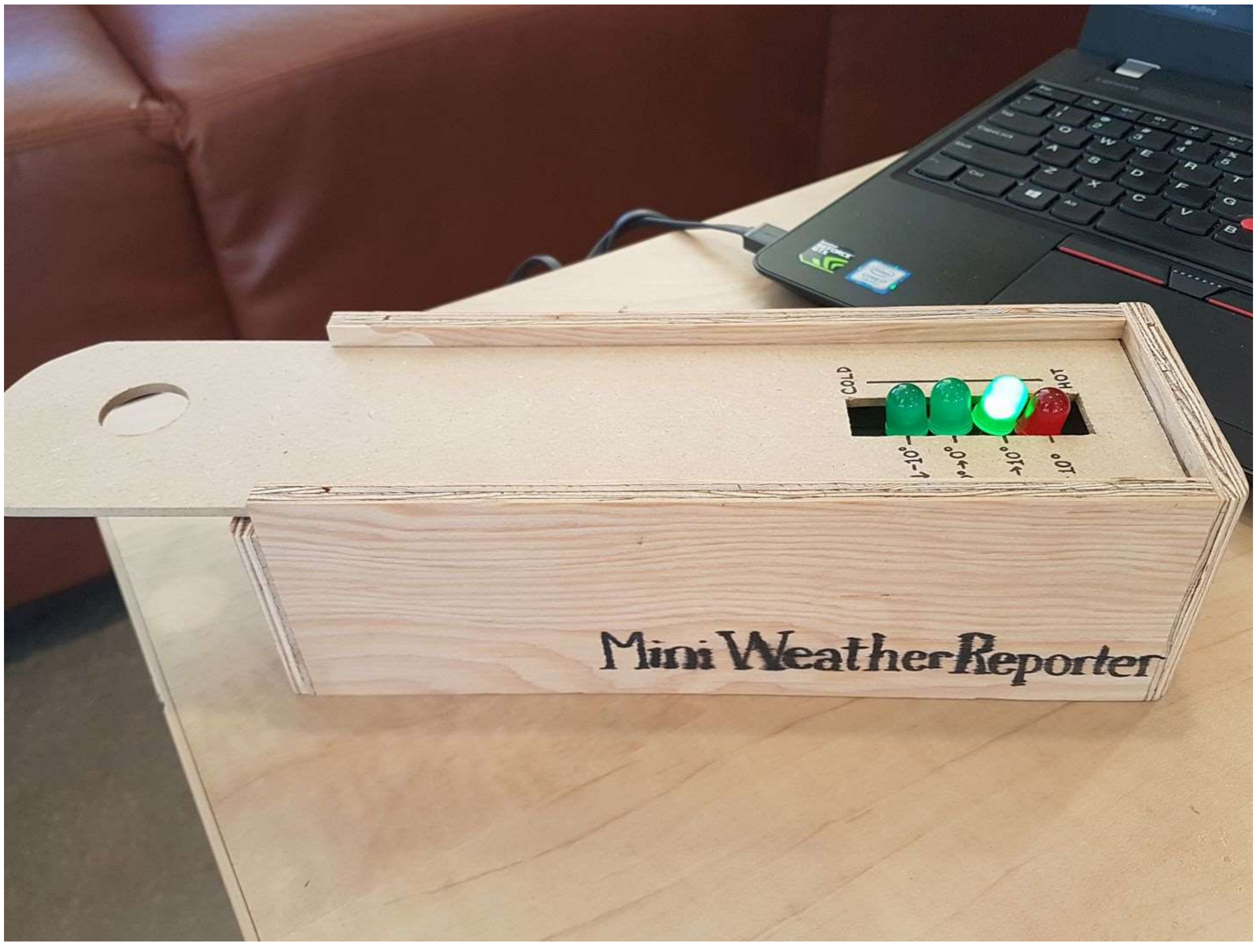
Final Project

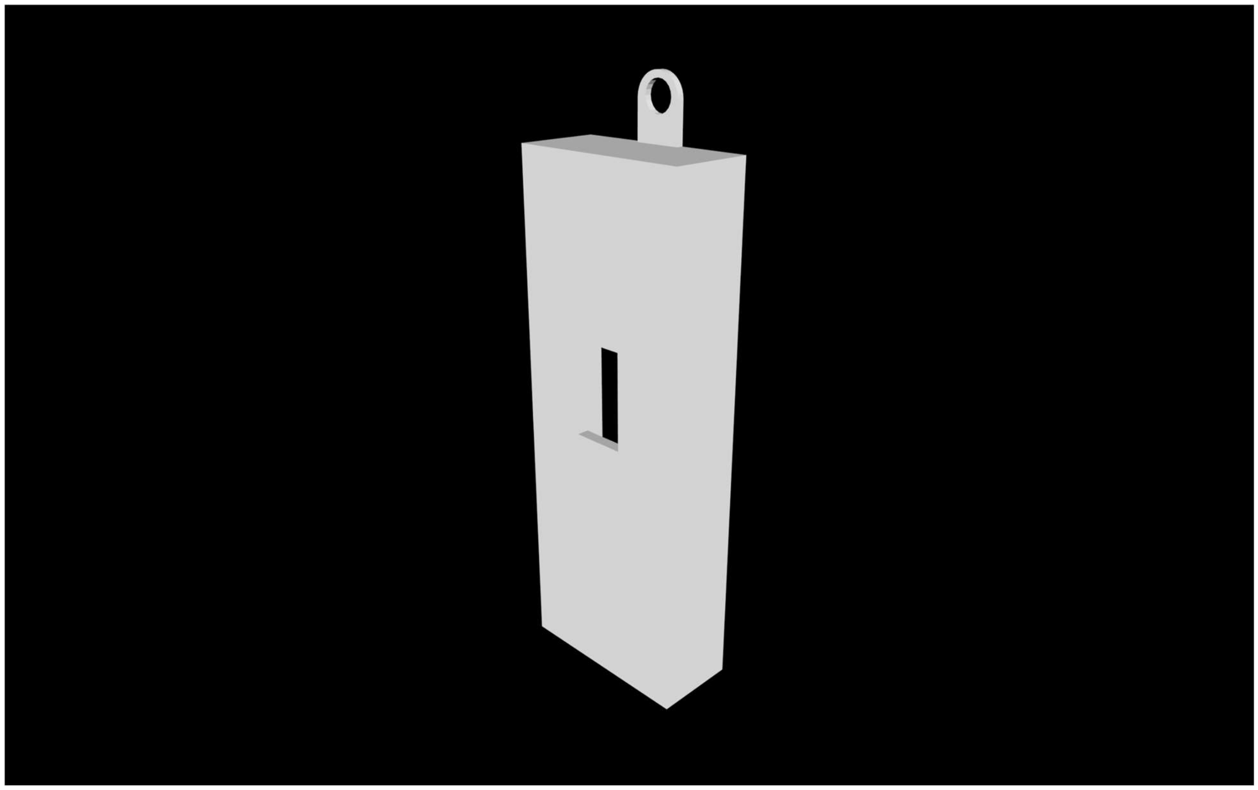
Group members: Eddie Minh, Swakhar Poddar, Anh Chau, Wing Hei Chang (Chloe).

- Project Title: Mini Weather Reporter

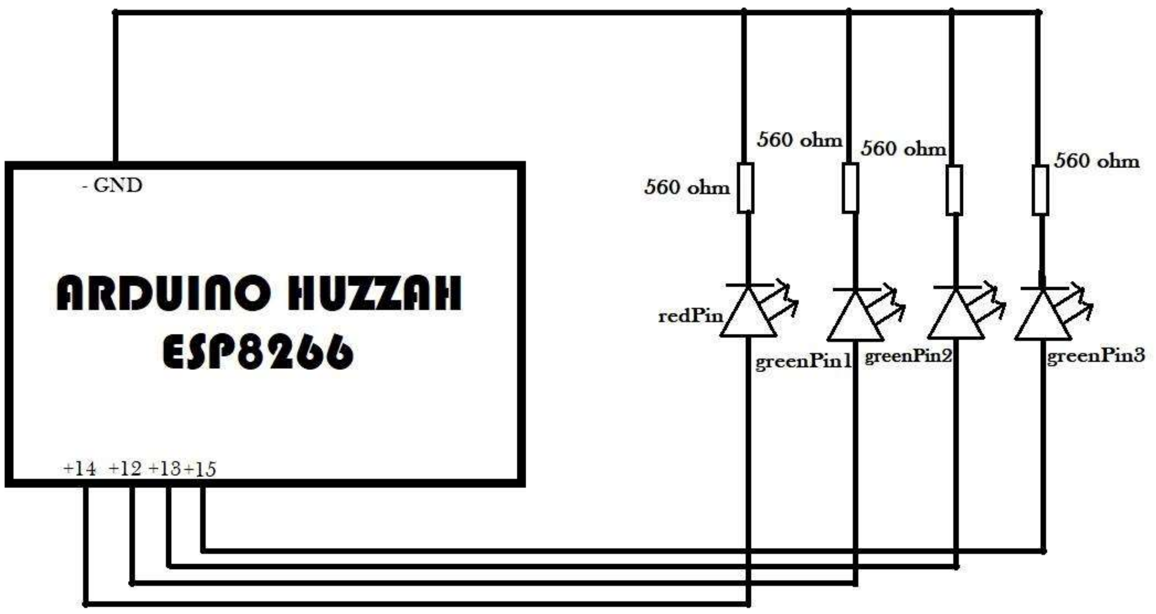
- **Project design**

- 

- **3D Model**

- 

- **Circuit Diagram**

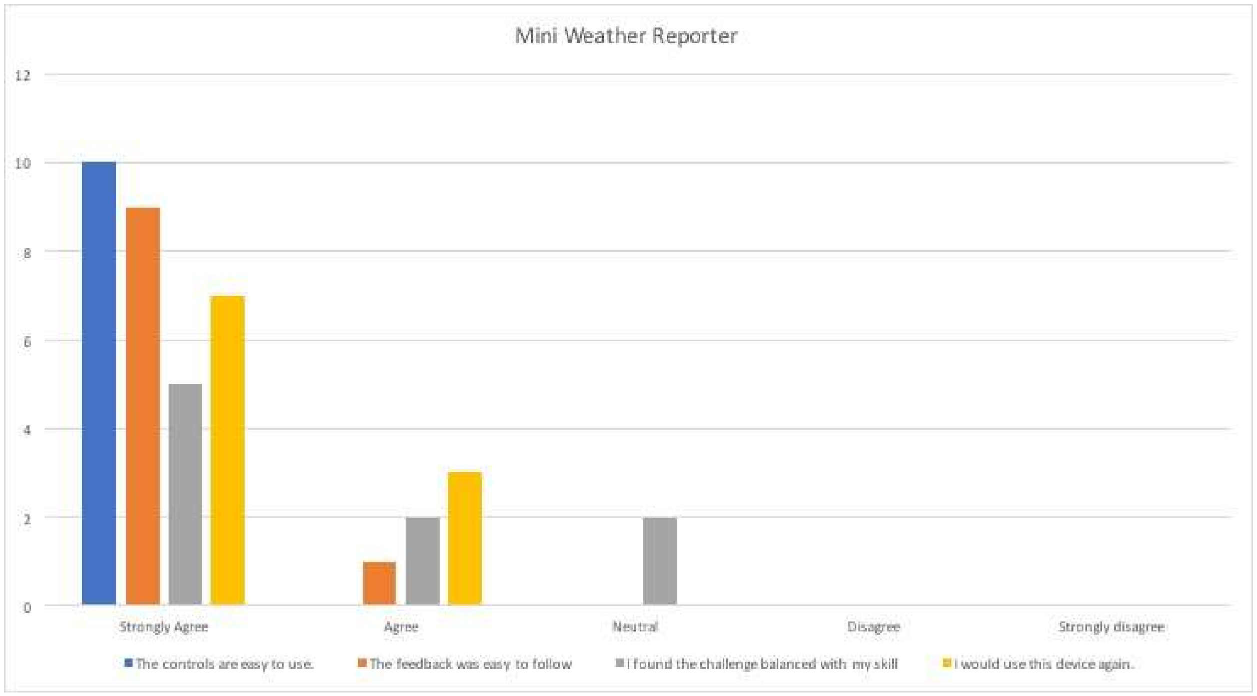


CIRCUIT DIAGRAM

MINI WEATHER REPORTER

-

- **<u>Survey</u>**

- 

We have given out surveys to 10 people during Show and Tell. The result and feedbacks we got are satisfying. All of them strongly agrees that the control of Mini Weather Reporter is easy to use.

- **Code**

```
 //libraries and definitions included in the program

#include <Arduino.h>
#include <ArduinoJson.h>


#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>


#include <ESP8266HTTPClient.h>


#define USE_SERIAL Serial
#define JSON_BUFF_DIMENSION 2500
#define COMMON_ANODE


ESP8266WiFiMulti WiFiMulti;

//declare LED values
int greenPin1 = 14;
int greenPin2 = 12;
int greenPin3 = 13;
int redPin = 15;


//declare other values
String payload;
float tempC;

void setup() {

USE_SERIAL.begin(115200);
// USE_SERIAL.setDebugOutput(true);
payload.reserve(JSON_BUFF_DIMENSION);

USE_SERIAL.println();
USE_SERIAL.println();
USE_SERIAL.println();

for(uint8_t t = 4; t > 0; t--) {
USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
USE_SERIAL.flush();
delay(1000);
}


WiFiMulti.addAP("maker", "ubcmaker"); // ID and Password of the Wifi connection used.

pinMode(greenPin1, OUTPUT);
pinMode(greenPin2, OUTPUT);
pinMode(greenPin3, OUTPUT);
pinMode(redPin, OUTPUT);




}

void loop() {
// wait for WiFi connection
if((WiFiMulti.run() == WL_CONNECTED)) {


HTTPClient http;


USE_SERIAL.print("[HTTP] begin...\n");
// configure traged server and url
//http.begin("http://api.openweathermap.org/data/2.5/weather?id=5990579&APPID=e5a8df70
83fafd542c11ba2facea8ea6", "7a 9c f4 db 40 d3 62 5a 6e 21 bc 5c cc 66 c8 3e a1 45 59
38"); //HTTPS
http.begin("http://api.openweathermap.org/data/2.5/weather?id=5990579&APPID=e5a8df7083
fafd542c11ba2facea8ea6"); //HTTP


USE_SERIAL.print("[HTTP] GET...\n");
// start connection and send HTTP header
int httpCode = http.GET();


// httpCode will be negative on error
if(httpCode > 0) {
// HTTP header has been send and Server response header has been handled
USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);


// file found at server
if(httpCode == HTTP_CODE_OK) {
payload = http.getString();
Colorresult(payload.c_str());  // The call of function Colorresult() parses c string
text “payload” and manipulate the LEDs accordingly.





} else {
USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
}


http.end();

}
delay (5000);
//prompt for data every 5 seconds to maintain accuracy.
}





}
```

```
//function to pars Json string from api.openweathernetwork.com
void Colorresult(const char * jsonString) {
StaticJsonBuffer<20000> jsonBuffer;

// FIND FIELDS IN JSON TREE
JsonObject& root = jsonBuffer.parseObject(jsonString);
if (!root.success()) {
Serial.println("parseObject() failed"); //Feedback if parsing fails.
return;
}
JsonObject& main = root["main"];
float temp = main["temp"]; //Parse temperature value
tempC= temp - 273.15; //Convert from K to C
Serial.println (tempC);
if (tempC >= 10) {
digitalWrite(redPin, HIGH);// red LED
}else if (tempC > 0 && tempC < 10){
digitalWrite(greenPin1, HIGH); //green LED 1
}else if (tempC > -10 && tempC <= 0){
digitalWrite(greenPin1, HIGH); //green LED 1 & green LED 2
digitalWrite(greenPin2, HIGH);
}else{
digitalWrite(greenPin1, HIGH);////green LED 1 & green LED 2 & green LED 3
digitalWrite(greenPin2, HIGH);
digitalWrite(greenPin3, HIGH);
}
}
```

-