EcoPower Track:

Sustainable Room Energy Monitoring Project

Group members: Ruxandra Burian, Zichen Jing (Nancy), Jiwoo Lee, Afra Alhammadi, Renata Espinosa

### **Problem definition**

With increasing greenhouse gas emissions, global warming is a worldwide problem this decade. The more the temperature rises, the more people rely on air conditioners that use electricity as fuel. Furthermore, the unconscious opening of lights in households also leads to the overuse of electricity. Both lead to the vicious cycle of increasing electricity usage and greenhouse gas emissions.

Within the NYUAD community, all students are covered by specific amounts of financial aid. Moreover, partial-aid students are not required to pay electricity fees. The unconsciousness of daily electricity usage and its corresponding prices leads to wasteful lifestyles and unhealthy environments. Our project aims to create awareness regarding electricity usage from air conditioners and lights for the students and to create a more habitable environment. To achieve this we developed EcoPower Track: Sustainable Room Energy Monitoring Project. The device will help students with data analysis on their energy usage and how much it costs for the amount of energy they consume. Additionally, in order to have a bigger impact, the device will be meant to serve competition between students to promote having a sustainable lifestyle. Prizes such as a single room for the next academic year for the students who live the most sustainably will ensure that rather than just creating awareness, there will also be action taken.

### **Objective**

The objective of this project is to measure the energy consumption (electricity) from the use of air conditioners and lights in NYUAD campus dorms by students. With the use of devices including the M5Stamp-C3[[1]](#footnote-0) as the host controller, LED light (WS2812C-2020-V1[[2]](#footnote-1)), accelerometer/gyroscope (TDK MPU-6500[[3]](#footnote-2)), light sensor (VEML7700-TR[[4]](#footnote-3)), humidity, barometric pressure, and ambient temperature sensor (BME280[[5]](#footnote-4)). This will help students encourage healthy consumption, prepare them for future living when they have to pay for electricity and remind them about the importance of sustainability. This aims to be achieved without the need of having to make a huge investment in changing the infrastructure of the campus buildings.

### **Overview**

The device will collect the temperature of the room and will process it to provide an estimate of the length of time the AC has been turned on by keeping track of the change in temperature. The humidity will be collected with the objective of determining how long the windows have been open. From the light intensity, the period of time the light has been turned on will be estimated. From all this data, the approximate value of kilowatts used throughout the day will be calculated, and later on, the electricity will be obtained. The average time in an hour that the AC was on, and the light was turned on will be uploaded to a database online, and the kilowatts and the price of electricity will also be uploaded daily.

When the temperature and humidity of the room are within the acceptable range, the LED of the device will show a green color. Otherwise, the LED will turn red.

The device is an oval-shaped box with protruding legs that will be placed around the ceiling area. More specifically, the legs of the device will allow it to be attached to a wood grill design on top of the wardrobe in every dorm room.

Assumptions:

* The kWh used by the AC will be calculated assuming a cooling capacity of 5500 BTU per hour.
* It will be assumed that keeping the window open leads to inefficient use of energy, as the AC from nearby rooms will need to use more energy to cool the room.
* It is assumed that an ideal temperature range for the AC to be on is 20 degrees and above, as different sources state that having the AC on at a temperature 20 degrees below the outside temperature leads to inefficiency.
* The temperature sensor will measure the difference in temperature between the initial and final states when there is a temperature change, assuming that the AC turns off once the desired temperature is reached and turns on again once the temperature starts to rise.
* We also assume that the kWh is 0.070 AED for the green band and 0.075 AED for the red bar.
* A humidity with a range below the average humidity of Abu Dhabi will be considered acceptable as this will be used as an indicator of whether the room window is open or closed.
* The software of FireBase will be used to upload the information processed by the device.

Process

* Looking for an idea
* Researching /documentation
* Writing a draft persuade-code
* Designing the circuits
* Designing the shape for the circuit to fit in
* Writing the code
* Finalizing the design and circuit
* Ordering the 3D model from JCLB

Research

We researched the prices of kWh for energy consumption for AC to get an idea of how much we are consuming and how much it costs. We found out that in the UAE, the price of electricity is 0.079 US dollars per kWh for households and 0.110 US dollars for businesses. It can also vary depending on the size of the area. A general air conditioner will consume between 3000 and 3500 watts per hour. While light bulbs consume between 2 and 100 watts of electricity, it also depends on the size and type.

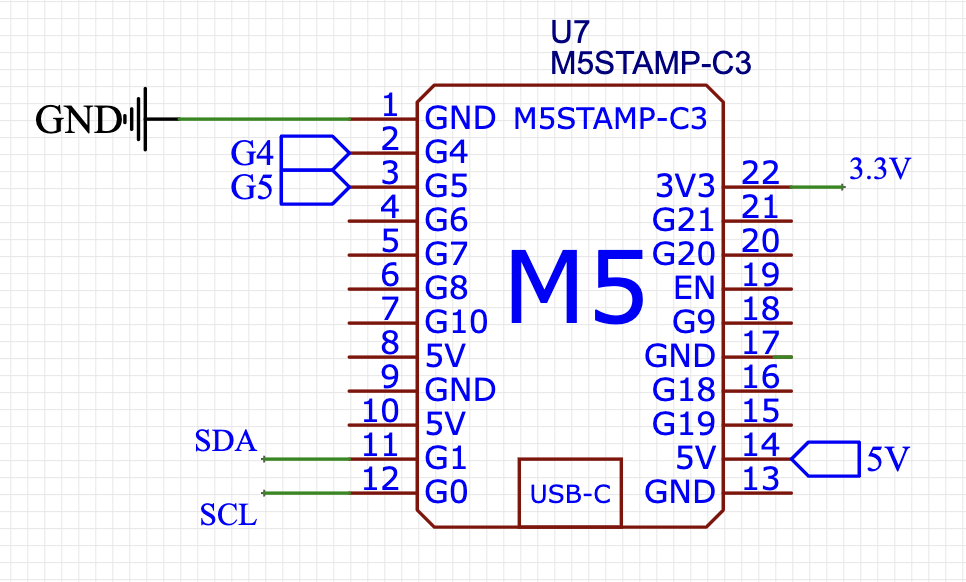
The relationship between humidity and AC affects the temperature in the room. High humidity cancels out the air conditioner’s cooling effect and warms your area. Higher humidity outdoors and in your area can both negatively affect the cooling potential of your HVAC system, HVAC experts recommend a humidity level below 60% for your area to be comfortable. The ideal temperature during summer is between 22 and 26 degrees Celsius, and in winter, it is between 10 and 20 degrees Celsius.

The cooling capacity of air conditioners refers to their ability to remove heat from the environment. It works as the conversion factor that provides the average amount of kilowatts used by AC per hour. To convert from its unit BMU per hour to kilowatts, the cooling capacity needs to be multiplied by the hours and then multiplied by 0.000293 to obtain kilowatts. The average number of kilowatts used by a lightbulb is 0.06 kilowatts per hour.

### **Datasheets and Circuit Design**

We used datasheets for each sensor and LED from their websites. The datasheets are pasted below, specified for each device, along with a brief description of connections. The final circuits for each device are built in EasyEDA.

* M5Stamp-C3



There is no datasheet for M5Stamp, it depends on the application.

GPIO 2, 3, 8, and 9 are already connected to some devices and therefore left out of our circuit design.

We connected GND to the GND pin so GND from all sensors could be connected here;

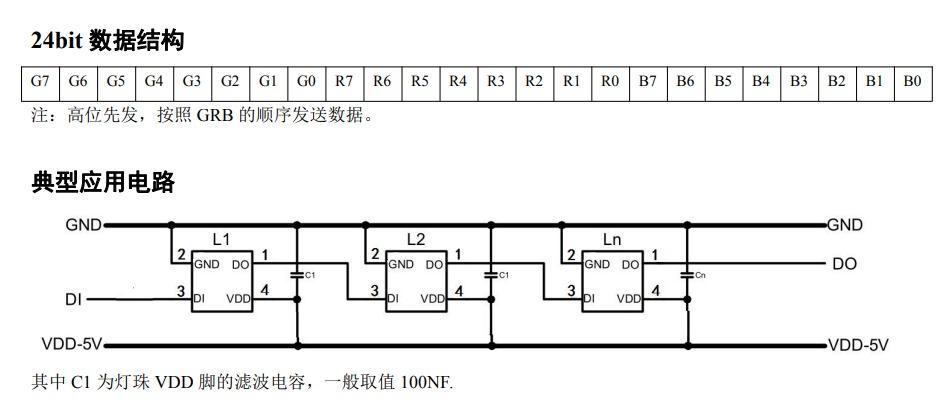
We decided to use G4 pin and G5 pin for MPU6500 and LED respectively;

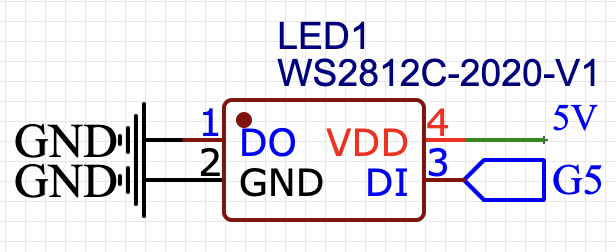
We connected 3.3V to the 3V3 pin so 3.3V from all sensors could be connected here;

5V was connected to 5V pin for LED connection;

SDA and SCL can only be connected to G1 and G0 pins respectively, SDA and SCL from all sensors could be connected here;

* LED light (WS2812C-2020-V1)



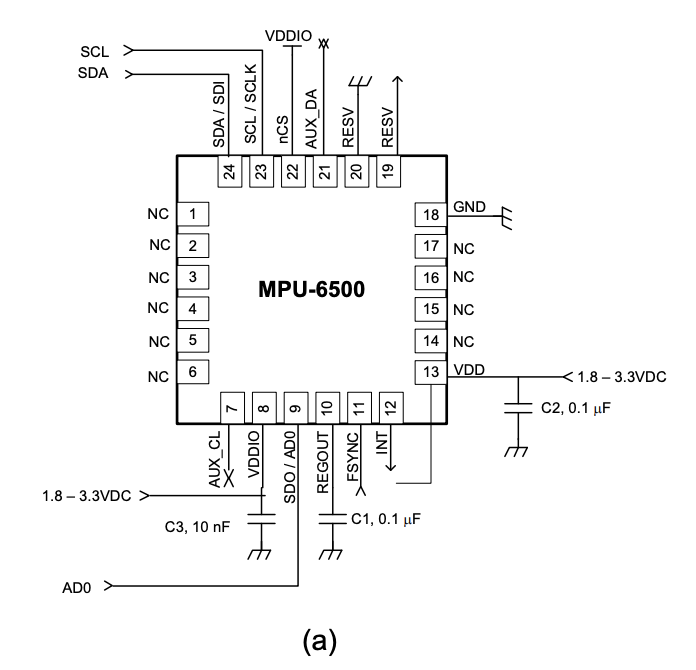


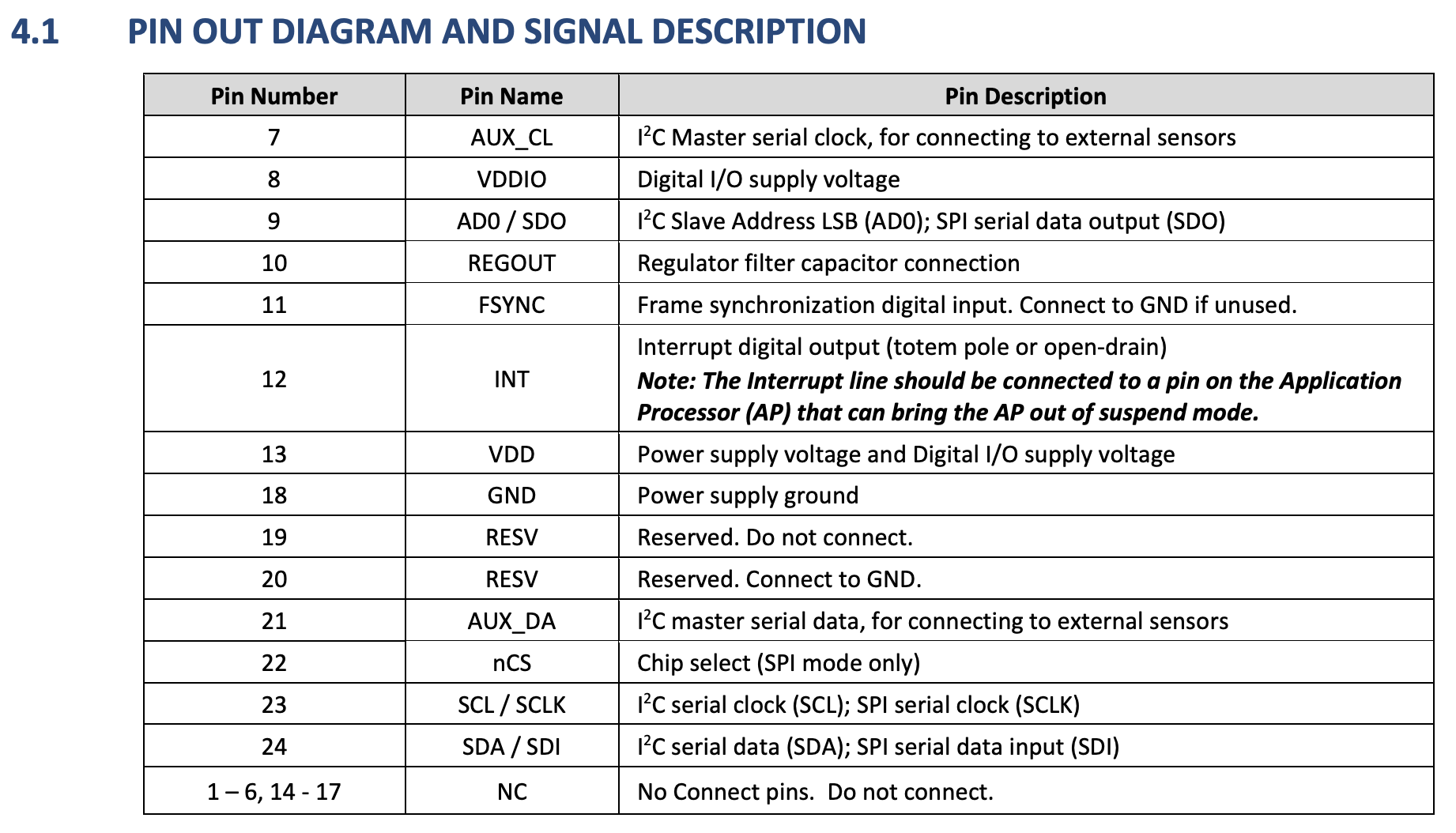
Since we are only using one LED in our device, both DO and GND pins are connected to ground;

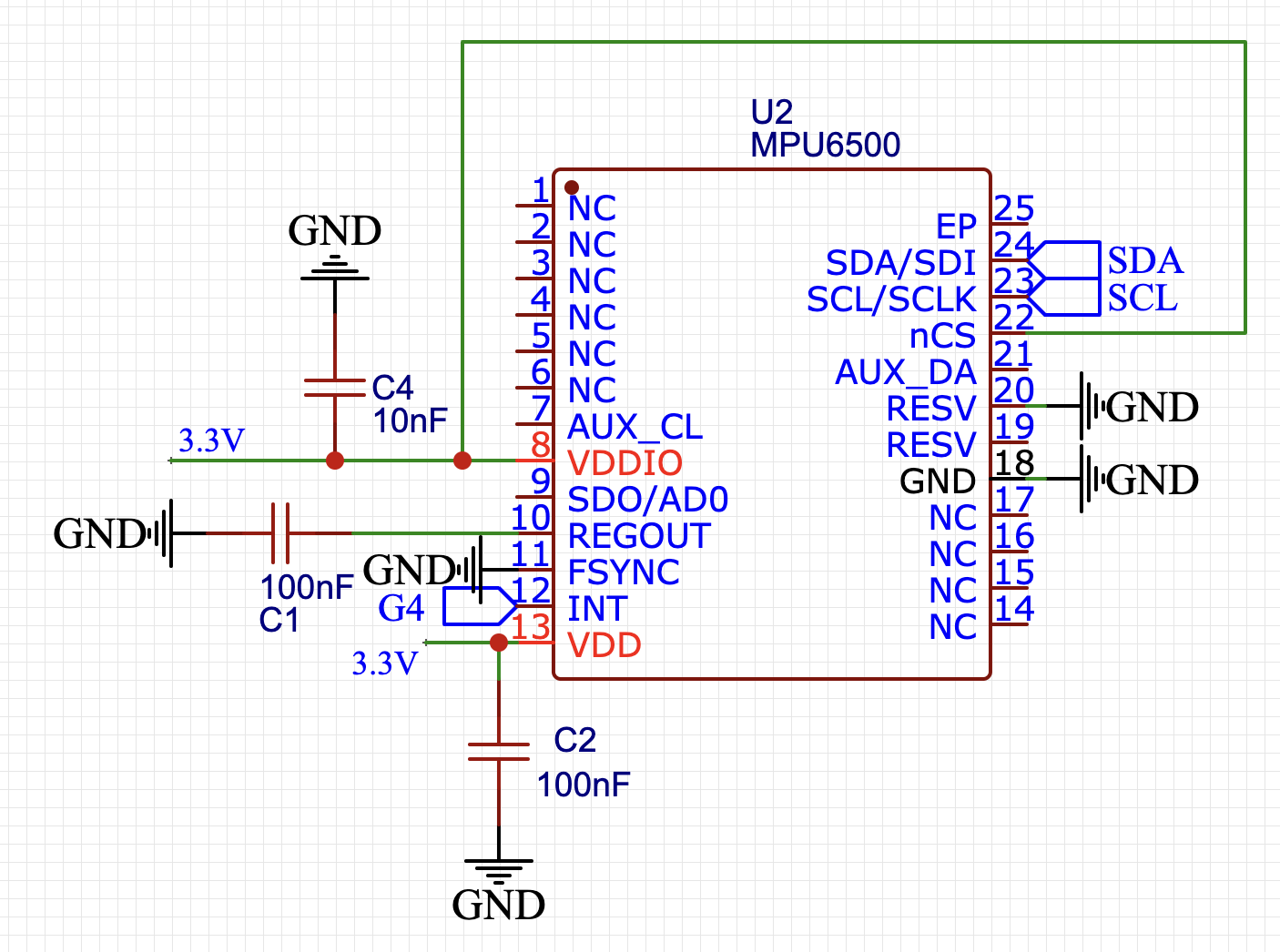
VDD pin was connected to 5V;

GPIO5 on the M5 stamp is free to connect therefore we connected DI to it.

* Accelerometer/gyroscope (TDK MPU-6500)







We followed both the datasheet and pin descriptions while connecting MPU6500.

We connected SDA to SDA/SDI pin;

SCL to SCL/SCLK pin;

VDDIO to nCS pin;

GND to RESV(20), FSYNC, and GND pins;

3.3V, capacitor FN15X104J160PNG (C2, 100nF), and GND to VDD pin;

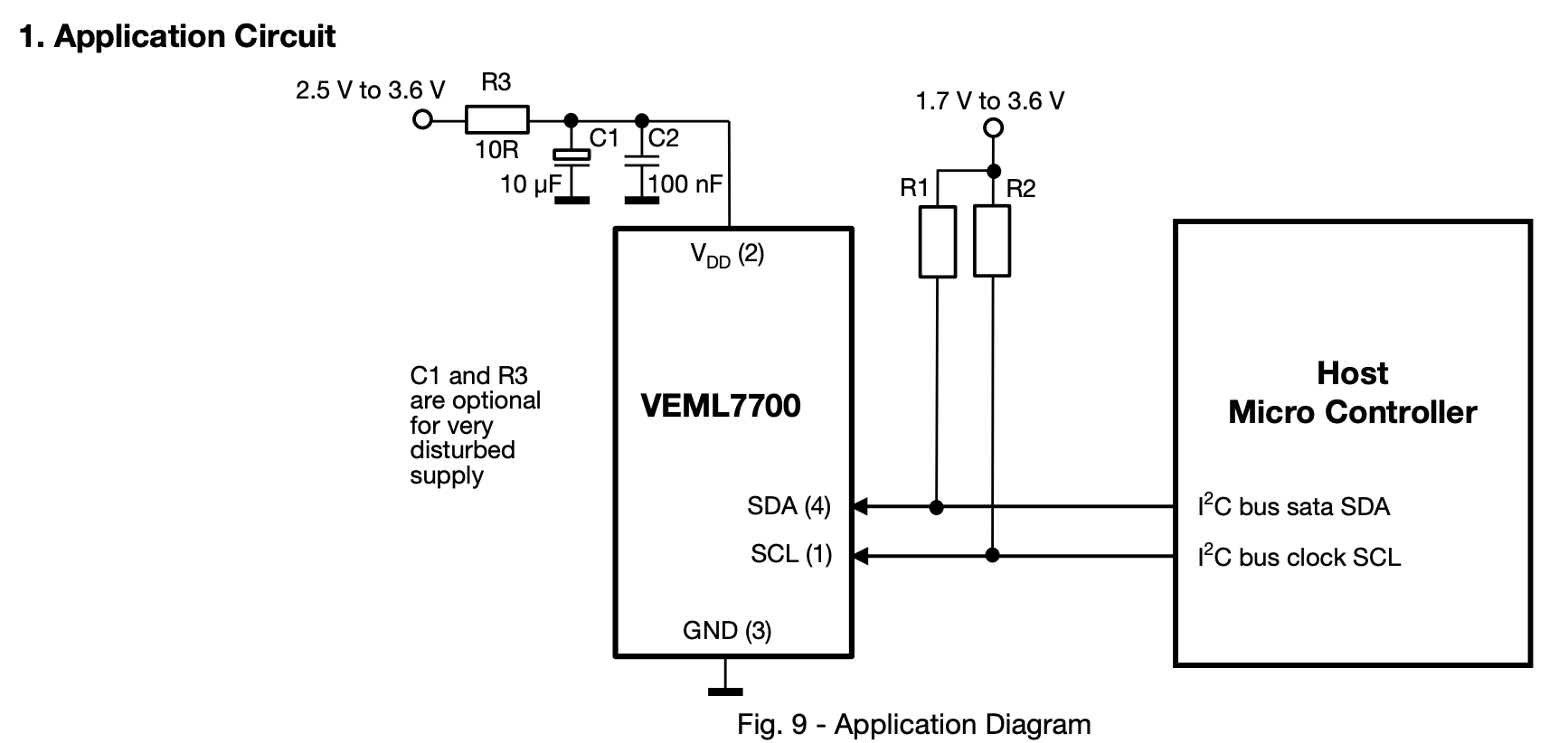
GPIO4 is free on the M5 stamp therefore connected to INT pin;

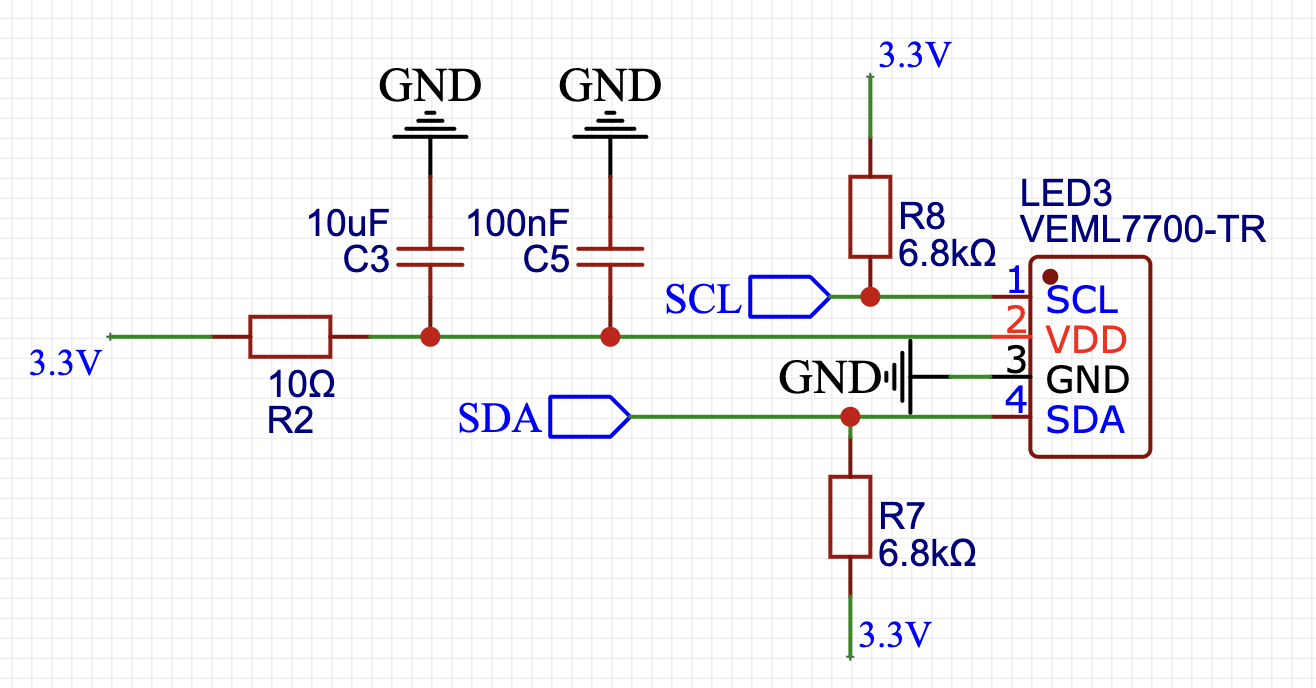
Capacitor FN15X104J160PNG (C1, 100nF), and GND to REGOUT pin;

3.3V, capacitor V103K0402X7R500NBT (C4, 10nF), GND, and nCS to VDDIO pin;

The rest pins are left out either because it is designed so or because they were unused.

* Light sensor (VEML7700-TR





The host microcontroller in our case is the M5 stamp.

We connected SDA to SDA pin;

SCL to SCL pin;

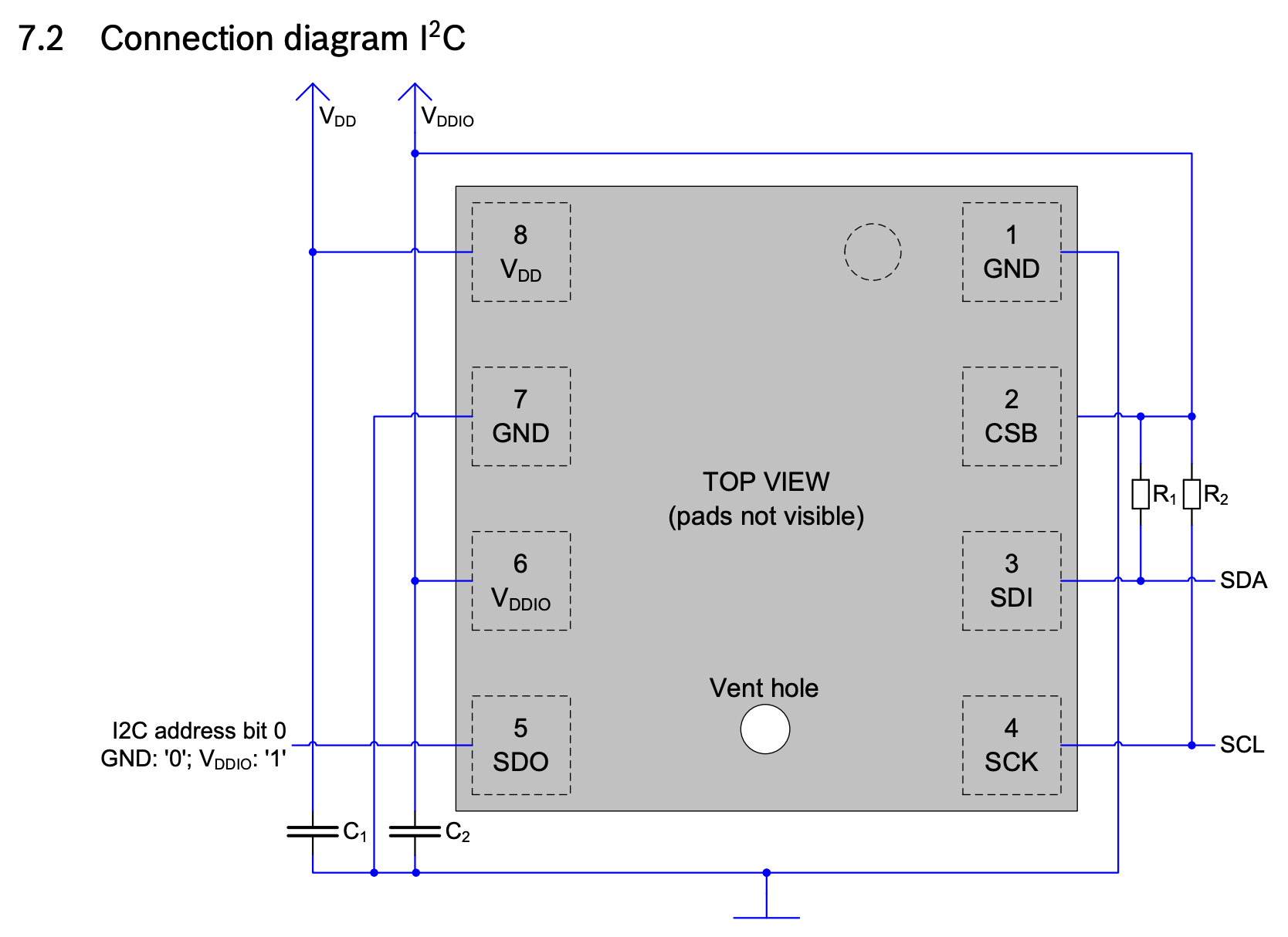
Two resistors CR0402J6K80Q10Z (R7 and R8, 6.8㏀) to 3.3V, to SDA and SCL pins;

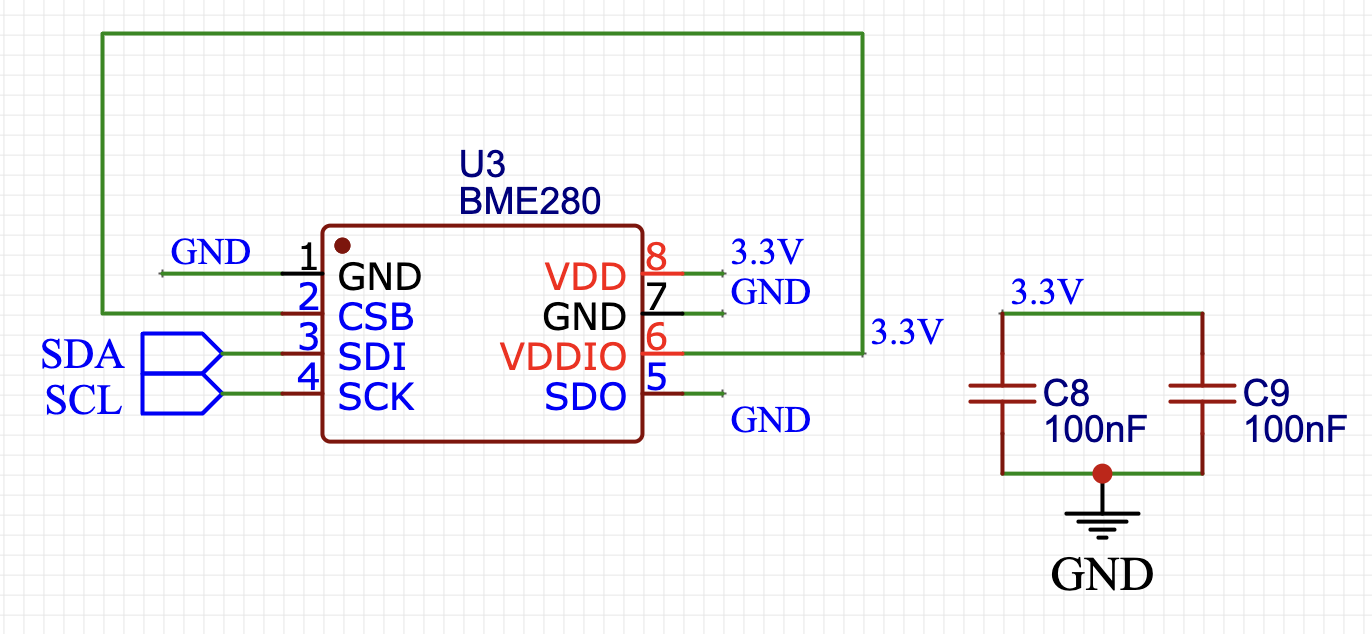
GND to GND pin;

3.3V, capacitor GRJ155R60J106ME11D (C3, 10uF), capacitor FN15X104J160PNG (C5, 100nF), resistor CR0402J10R0Q10Z (R2, 10Ω), to VDD pin;

GND is connected to both C3 and C5 capacitors.

* Humidity, barometric pressure, and ambient temperature sensor (BME280)





After interpreting the datasheet carefully, we decided to connect our circuit as two to avoid the crossing of wires and therefore avoid confusion.

We connected GND to both GND and SDO pins;

VDDIO pin to CSB pin;

SDA to SDI pin;

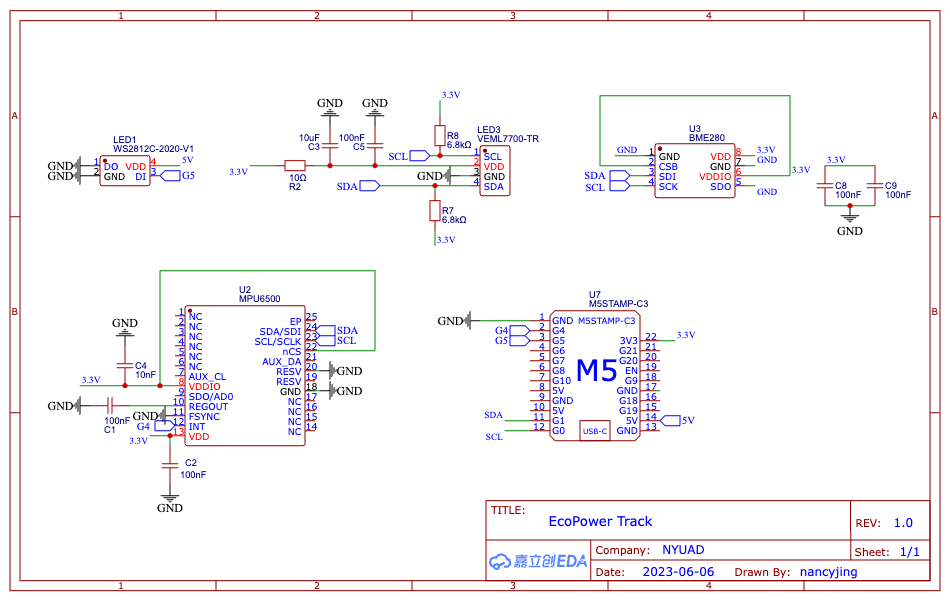
SCL to SCK pin;

3.3V to both VDDIO and VDD pins;

3.3V connected to VDDIO and VDD pins was extended out to two capacitors FN15X104J160PNG (C8 and C9, 100nF), and GND;

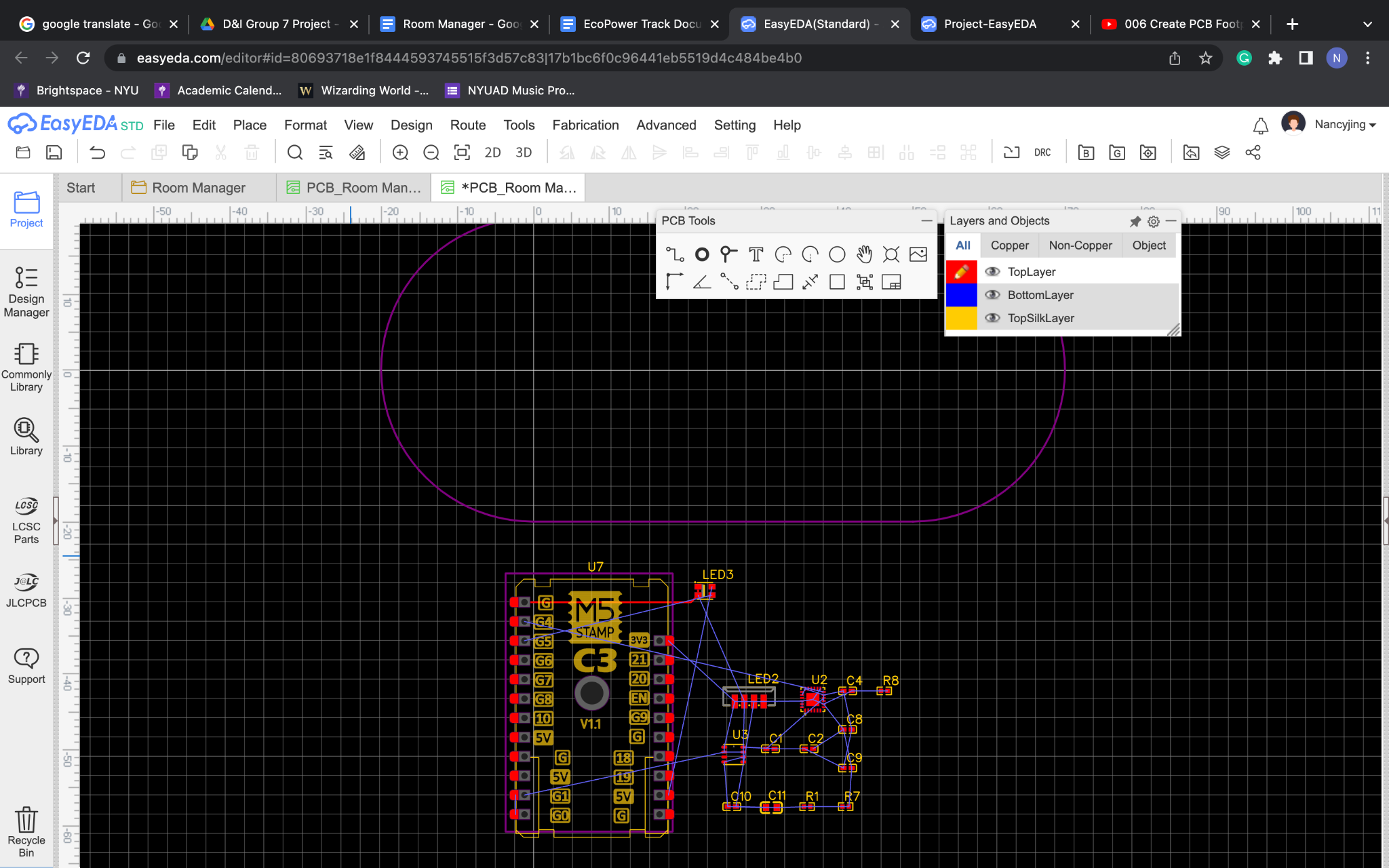
Since we have already connected the specified R1 and R2 shown on the datasheet for the VEML7700-TR (using 6.8㏀), we did not include them in the circuit design for BME280.

**Final circuit:**



This schematic of the circuit is then converted to PCB through EasyEDA.

### **PCB**

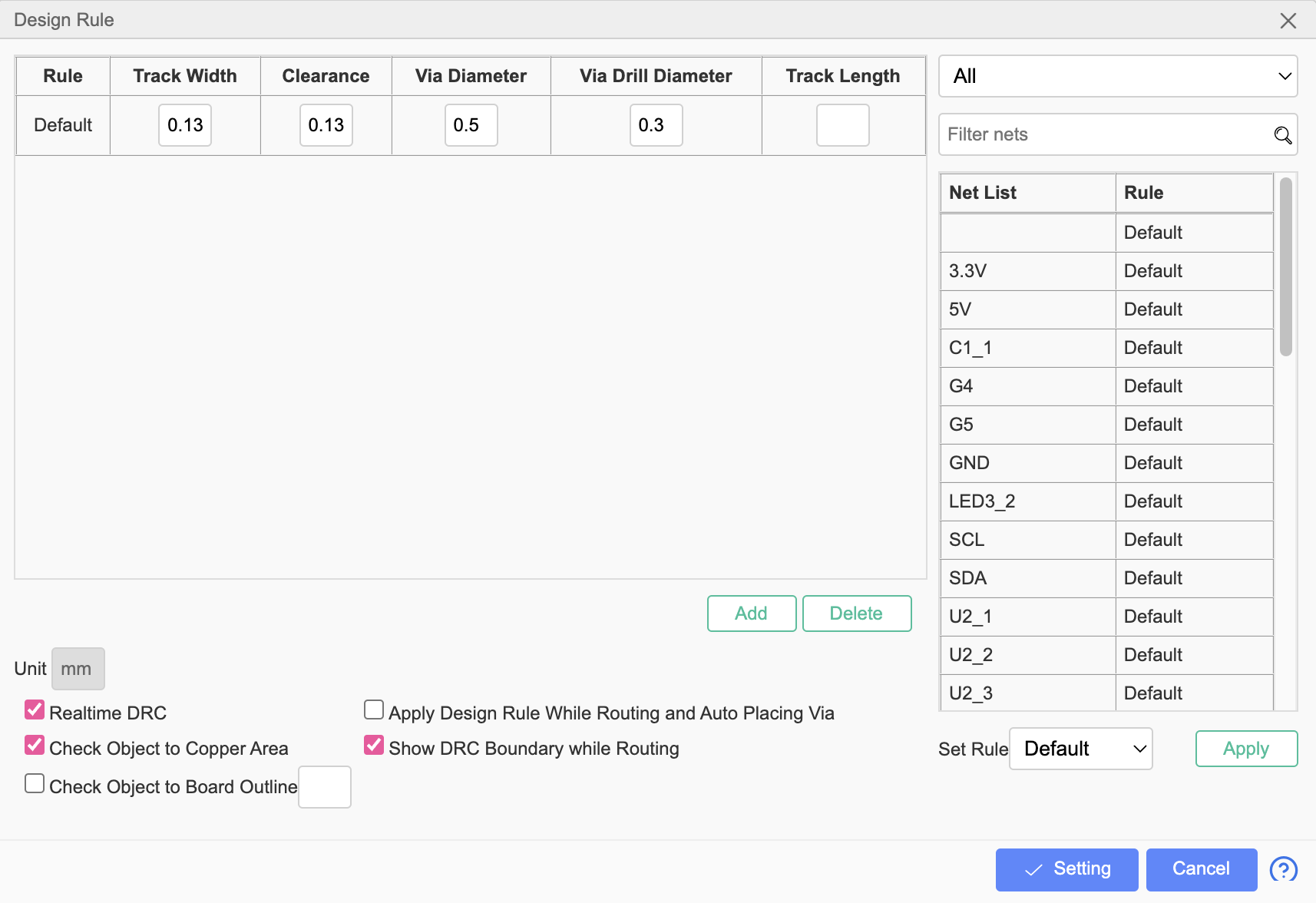


Before placing the sensors, LED, and M5 stamp, we need to choose the board outline of the circuit board. Aligning with our 3D design, we choose to make the board outline an oval shape. In EasyEDA, we constructed the oval shape by creating two semi-circles and connecting their ends by drawing two lines between them. The initial dimensions were two semi-circles with a radius of 20mm and lines being 50mm.

However, we considered including the holes for screw placement in the design of the board outline but we found out that it will impact the suitability of the circuit board in our 3D design, so we decided to increase the outline of the 3D design while decreasing the board outline of the circuit board. So that it can fit inside the outline of the 3D design and the screw holes instead of overlapping. The new dimensions of the circuit board outline are a radius of 19mm for each semi-circle and a length of 50mm for the lines connecting the semi-circles.

We placed the devices into the circuit board outline with capacitors and resistors placed near the sensor it is connected to. Also, we placed the LED away from VEML7700-TR so that it would not affect the accuracy of light detection from the VEML7700-TR. We placed BME280 away from the M5 stamp so that the heating up of the M5 stamp during usage will not affect the accuracy of temperature detection from BME280.

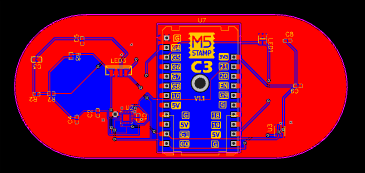
According to the PCB capabilities[[6]](#footnote-5) for 1-2 layers, we set the design rule as follows



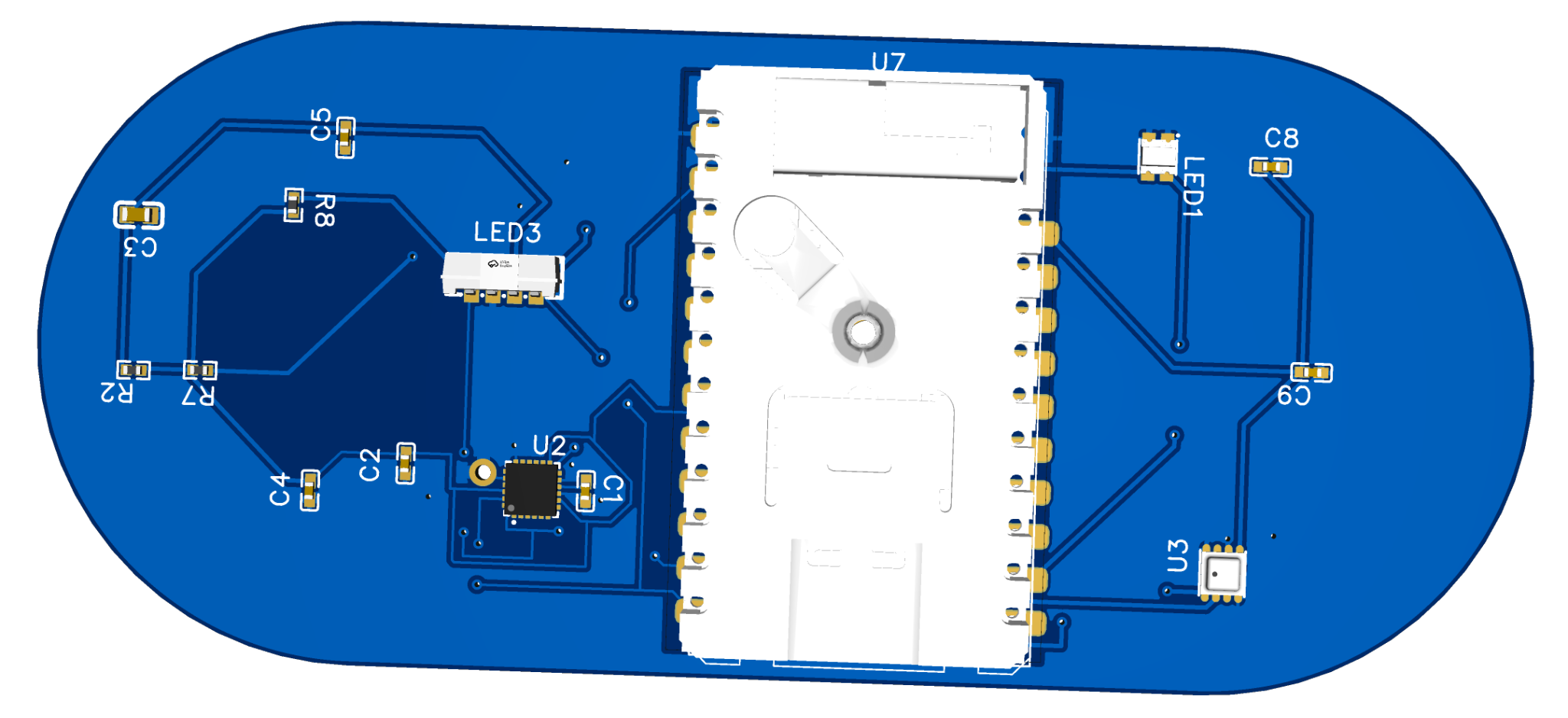
Since the wires on the top layer cannot cross each other, we used via to switch to the bottom layer for our wire connection. After the connection, we checked DRC to make sure there is no errors thus we can continue to the next step.

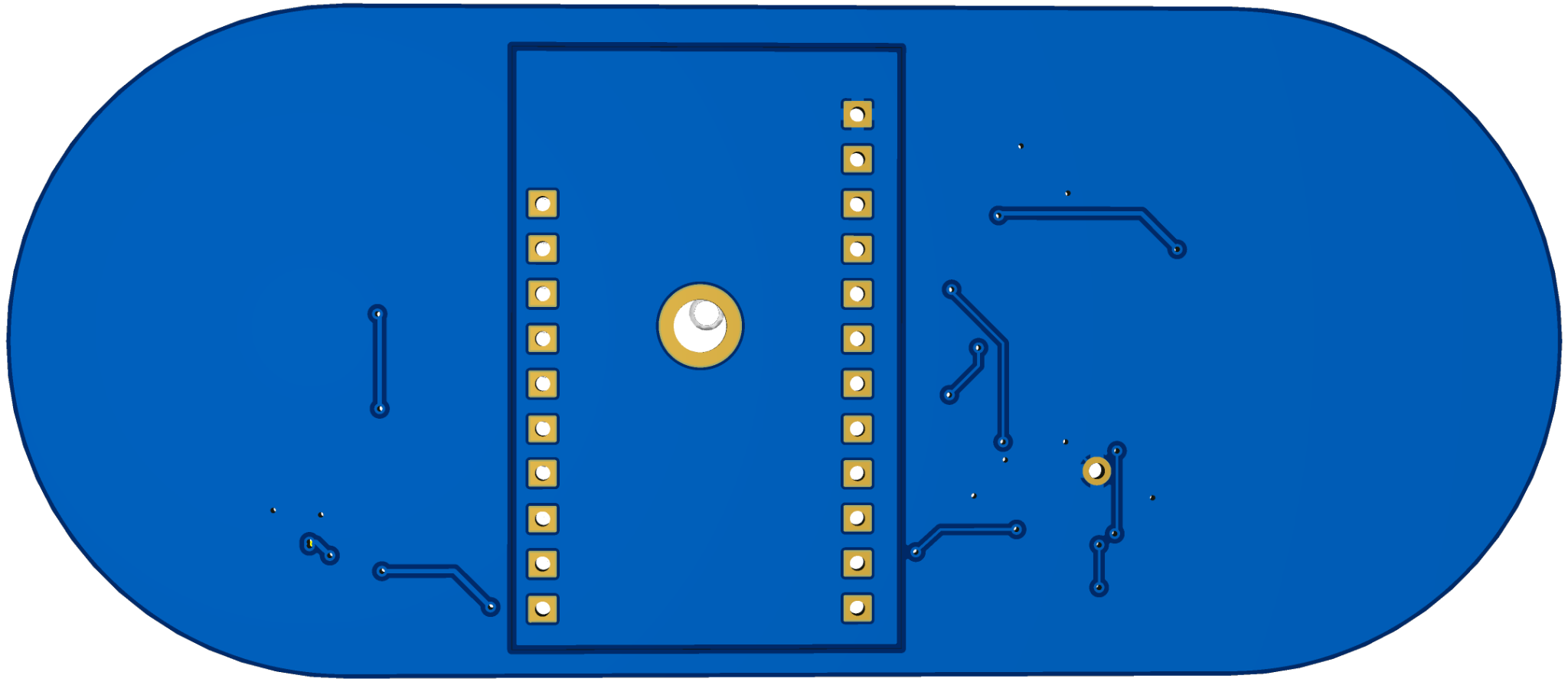
**Final PCB:**

We added copper areas of GND to both the top and the bottom layers.

****

It is then converted to 3D to assure the suitability of the 3D models of the sensors and LED with the circuit board.





The PCB design is then converted from EasyEDA standard to EasyEDA pro, it was exported as a step file to be imported to Autodesk Fusion 360 to check the suitability (width, length, height, and thickness) of it with our 3D model of the device.

On the day of submission, our PCB design has been approved for production and has progressed as shown below



### **Formulas**

* Calculate the kilowatts used by AC from the minutes it has been on:

KW=AC Cooling Capacity per minute \* minutes\*0.000293.

* Calculate the kilowatts used by light from the minutes it has been on:

KW= Milliseconds/3600 \* Kilowatts used by lightbulb per minute

* Total Kilowatts used:

TKW= AC kilowatts + Light Kilowatts

* Calculate the price of electricity from the total number of kilowatts:

Price = 0.079 \*KW

### **Pseudo Codes** Include Arduino library

### if ESP32 is defined,

### Include WiFi library

### otherwise if ESP8266 is defined,

### Include ESP8266WiFi library

### Include Firebase\_ESP\_Client library

### Include Wire library

### Include Adafruit Sensor library

### Include BME280 library

### Include VEML7700 sensor library

### 

### Include the file TokenHelper.h to provide the token generation process info

### Include the file RTDBHelper.h to provide the RTDB payload printing info and other helper functions.

### 

### Assign "renata" to global variable WIFI\_SSID

### Assign "1234" to global variable WIFI\_PASSWORD

### 

### Assign Firebase project API Key to global variable API\_KEY

### 

### Assign "ecomanager11@gmail.com" to global variable USER\_EMAIL

### Assign "sustainable123" to global variable USER\_PASSWORD

### 

### Assign the firebase URL to global variable DATABASE\_URL

### 

### Assign port number 23 to global variable PIN\_RED

### Assign port number 22 to global variable PIN\_GREEN

### 

### Declare FirebaseData variable fbdo

### Declare FirebaseAuth variable auth

### Declare FirebaseConfig variable config

### 

### Declare string variable uid

### Declare string variable databasePath

### Declare string variable tempPath

### Declare string variable humPath

### Declare string variable lightPath

### Declare string variable pricePath

### Declare string variable kwPath

### 

### Declare Adafruit\_BME280 variable bme for humidity and temperature sensor

### Declare Adafruit\_VEML7700 variable veml for light sensor

### 

### //

### 

### In function initBME()

### If BME280 sensor fails to begin with the address 0x76,

### Print "Could not find a valid BME280 sensor, check wiring!"

### Infinite loop

### 

### In function initWiFi()

### Begin Wi-Fi connection using WIFI\_SSID and WIFI\_PASSWORD

### Print "Connecting to WiFi .."

### While Wi-Fi status is not connected:

### Print '.'

### Delay for 1 second

### Print Wi-Fi local IP address

### Print an empty line

### 

### In function sendFloat(path, value)

### If Firebase Realtime Database successfully sets the float value at the specified path:

### Print "Writing value: ", variable value

### Print " on the following path: ", variable path

### Print "PASSED"

### Print "PATH: ", fbdo.dataPath()

### Print "TYPE: ", fbdo.dataType()

### Else:

### Print "FAILED"

### Print "REASON: ", fbdo.errorReason()

### 

### Declare constant NUM\_Sample as 60

### Declare constant NUM\_DaySample as 1440

### Declare variable TEMPsum as 0.0

### Declare variable samplectr as 0

### Declare variable samplectr2 as 0

### 

### Declare variable iniTEMP as 0

### Declare variable kw as 0

### Declare variable timecounter as 0

### Declare variable coolingcapacity as 0

### Declare variable costkw as 0

### Declare variable totalTimeHL as 0

### 

### Declare constant thresholdPin as 2

### Declare constant LT as 1000.0

### Declare variable prevLI as 0.0

### Declare variable kwlight as 0

### Declare variable pricelight as 9

### Declare variable HLStartTime as 0

### Declare variable HLEndTime as 0

### Declare variable HLduration as 0

### Declare variable timetokw as 90

### Declare variable hourendtime as 0

### Declare variable lighthour as 0

### 

### Function isLIaboveT(currentLI),

### If currentLI is greater than LT,

### currentLI true

### Otherwise

### currentLI false

### 

### Function setup()

### Begin serial communication at a baud rate of 115200

### Begin Wire communication

### Set PIN\_RED as an output pin

### Set PIN\_GREEN as an output pin

### 

### If VEML7700 sensor fails to begin,

### Print "Could not find VEML7700 sensor. Check your wiring"

### 

### Call initBME() function

### Call initWiFi() function

### 

### Set config.api\_key to API\_KEY

### Set auth.user.email to USER\_EMAIL

### Set auth.user.password to USER\_PASSWORD

### Set config.database\_url to DATABASE\_URL

### Invoke Firebase.reconnectWiFi with the argument true

### Call fbdo.setResponseSize with argument 4096

### 

### Set config.token\_status\_callback to tokenStatusCallback

### Set config.max\_token\_generation\_retry to 5

### Invoke Firebase.begin with config and auth as arguments

### 

### Print "Getting user UID"

### repeat while auth.token.uid is " "

### Print "."

### Wait for 1000 milliseconds

### 

### Assign auth.token.uid.c\_str() to variable uid

### Print "User ID:", user ID

### 

### Update database path by concatenating uid to "/UsersData/"

### 

### Set tempPath to the concatenation of databasePath and "/temperature"

### Set humPath to the concatenation of databasePath and "/humidity"

### Set pricePath to the concatenation of databasePath and "/Price"

### Set kwPath to the concatenation of databasePath and "/Kw"

### Set lightPath to the concatenation of databasePath and "/Light"

### 

### In void function loop

### Set TEMP to the value returned by bme.readTemperature()

### Set HUMD to the value returned by bme.readHumidity()

### Set currentLI to the value returned by veml.readLux()

### 

### If iniTEMP is greater than TEMP

### set kw as kw + (iniTEMP - TEMP) \* coolingcapacity

### set TEMP to iniTEMP

### 

### if isLIaboveT(currentLI) function returns true

### if HLStarttime is 0

### Start counting time in milliseconds, record starting time

### Otherwise

### if HLStarttime is not 0

### Stop counting time, record stop time

### Calculate duration of High light by HLEndTime - HLStartTime;

### Add the HLDuration to the totaltimeHL

### Reset HLStartTime to 0

### Reset HLEndTime to 0

### 

### Add TEMP to TEMPsum

### increase samplectr every minute

### increase samplectr2 every minute

### 

### If samplectr is not equal to NUM\_Sample

### If HLduration is not equal to 0

### Add HLduration to lighthour

### Set HLduration to 0

### Else if samplectr is equal to NUM\_Sample and HLStartTime is not equal to 0

### Set hourendtime to the current time in milliseconds

### Set lighthour to hourendtime minus HLStartTime

### 

### If samplectr is equal to NUM\_Sample

### Assign TEMPsum divided by NUM\_Sample to float variable avgTEMP

### Call sendFloat function with arguments tempPath and avgTEMP

### Call sendFloat function with arguments lightPath and lighthour

### Set lighthour to 0

### Set TEMPsum to 0.0

### Set samplectr to 0

### 

### If samplectr2 is equal to NUM\_DaySample

### Assign kw+totalTimeHL \*timetokw to kw

### Call sendFloat function with arguments pricePath and price

### Call sendFloat function with arguments kwPath and kw

### Reset kw to 0

### Reset price to 0

### 

### Call analogWrite function with arguments PIN\_RED and 52

### 

### If TEMP is more or equal to 20, and HUMD is less or equal to 50

### Call analogWrite function with arguments PIN\_GREEN and 168

### 

### Wait for 600000 milliseconds

### 

### **Codes**

#include <Arduino.h>

#if defined(ESP32)

#include <WiFi.h>

#elif defined(ESP8266)

#include <ESP8266WiFi.h>

#endif

#include <Firebase\_ESP\_Client.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

#include <Adafruit\_VEML7700.h> //Include the VEML7700 library

// Provide the token generation process info.

#include "addons/TokenHelper.h"

// Provide the RTDB payload printing info and other helper functions.

#include "addons/RTDBHelper.h"

// Insert your network credentials

#define WIFI\_SSID "3fary"

#define WIFI\_PASSWORD "0987654321"

// Insert Firebase project API Key

#define API\_KEY "AIzaSyCloe5gtVpURkCzMKoKEhKW28zkRoxFI44"

// Insert Authorized Email and Corresponding Password

#define USER\_EMAIL "ecomanager11@gmail.com"

#define USER\_PASSWORD "sustainable123"

// Insert RTDB URLefine the RTDB URL

#define DATABASE\_URL "https://console.firebase.google.com/u/1/project/esp32-demo-9824d/database/esp32-demo-9824d-default-rtdb/data/~2F"

//LED

#define PIN\_RED 23 // GIOP23

#define PIN\_GREEN 22 // GIOP22

// Define Firebase objects

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

// Variable to save USER UID

String uid;

// Variables to save database paths

String databasePath;

String tempPath;

String humPath;

String lightPath;

String pricePath;

String kwPath;

// BME280 sensor

Adafruit\_BME280 bme; // I2C

Adafruit\_VEML7700 veml; // Create a VEML7700 object for light sensor

// Timer variables (send new readings every three minutes)

/\*unsigned long sendDataPrevMillis = 0;

unsigned long timerDelay = 180000;\*/

/\* Initialize BME280

void initBME(){

if (!bme.begin(0x76)) {

Serial.println("Could not find a valid BME280 sensor, check wiring!");

while (1);

}

}\*/

// Initialize WiFi

void initWiFi() {

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

Serial.print("Connecting to WiFi ..");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print('.');

delay(1000);

}

Serial.println(WiFi.localIP());

Serial.println();

}

// Write float values to the database

void sendFloat(String path, float value){

if (Firebase.RTDB.setFloat(&fbdo, path.c\_str(), value)){

Serial.print("Writing value: ");

Serial.print (value);

Serial.print(" on the following path: ");

Serial.println(path);

Serial.println("PASSED");

Serial.println("PATH: " + fbdo.dataPath());

Serial.println("TYPE: " + fbdo.dataType());

}

else {

Serial.println("FAILED");

Serial.println("REASON: " + fbdo.errorReason());

}

}

//TEMP constants

const int NUM\_Sample = 60; //60 samples of temperature data per hour

const int NUM\_DaySample = 1440;

float TEMPsum = 0.0;

int samplectr = 0;

int samplectr2=0;

//Variable needed for temperature data

float iniTEMP = 0;

float kw = 0;

float timecounter = 0; // per minute

float coolingcapacity = 91.6; //should be kw per minute

float costkw = 0.079;

float totalTimeHL=0;

//LIGHT constants

const int thresholdPin = 2; // Define the pin number for the light threshold

const float LT = 1000.0; //Define the light threshold (adjust as needed)

float prevLI = 0.0; // previous light intensity reading

float kwlight =0;

float pricelight=0.079; //WRITE DOWN THE PRICE FOR THE LIGHT

unsigned long HLStartTime = 0; //start time of high light intensity (aka light on)

unsigned long HLEndTime = 0; // end time of light on (when light is off)

unsigned long HLduration = 0; //variable that shows the duration of light on in milliseconds (later converted to minutes)

float timetokw=0.06/60; //kilowatts per minute

float hourendtime=0;

float lighthour=0;

bool isLIaboveT(float currentLI) {

if (currentLI > LT)

return true;

else

return false;

}

void setup(){

Serial.begin(115200);

Wire.begin();

pinMode(PIN\_RED, OUTPUT);

pinMode(PIN\_GREEN, OUTPUT);

/\*if (!veml.begin()) {

Serial.println("Could not find VEML7700 sensor. Check your wiring"); //Error

//while (1);

}\*/

// randomSeed() will then shuffle the random function.

randomSeed(analogRead(0));

// Initialize BME280 sensor

//initBME();

initWiFi();

// Assign the api key (required)

config.api\_key = API\_KEY;

// Assign the user sign in credentials

auth.user.email = USER\_EMAIL;

auth.user.password = USER\_PASSWORD;

// Assign the RTDB URL (required)

config.database\_url = DATABASE\_URL;

Firebase.reconnectWiFi(true);

fbdo.setResponseSize(4096);

// Assign the callback function for the long running token generation task \*/

config.token\_status\_callback = tokenStatusCallback; //see addons/TokenHelper.h

// Assign the maximum retry of token generation

config.max\_token\_generation\_retry = 5;

// Initialize the library with the Firebase authen and config

Firebase.begin(&config, &auth);

// Getting the user UID might take a few seconds

Serial.println("Getting User UID");

while ((auth.token.uid) == "") {

Serial.print('.');

delay(1000);

}

// Print user UID

uid = auth.token.uid.c\_str();

Serial.print("User UID: ");

Serial.println(uid);

// Update database path

databasePath = "/UsersData/" + uid;

// Update database path for sensor readings

tempPath = databasePath + "/temperature"; // --> UsersData/<user\_uid>/temperature

humPath = databasePath + "/humidity"; // --> UsersData/<user\_uid>/humidity

pricePath = databasePath + "/Price"; // --> UsersData/<user\_uid>/pressure

kwPath = databasePath + "/Kw"; // --> UsersData/<user\_uid>/pressure

lightPath = databasePath + "/Light";

}

void loop(){

//Reading temperature, humidity, light intensity

float TEMP = random(10, 50); //temperature in celcius

float HUMD = random(10, 80); //humidity in percentage (0-100)

float currentLI = random(10, 100); //light in lux

// Calculating kw of temperature

if (TEMP < iniTEMP) {

kw = kw + (iniTEMP - TEMP) \* coolingcapacity\*0.000293;

}

iniTEMP = TEMP;

//Check if light intensity above threshold

if (isLIaboveT(currentLI)) {

if (HLStartTime == 0) {

// Start counting time if this is the beginning of high light intensity

HLStartTime = millis() ;

}

}

else {

// Light intensity is below the threshold

if (HLStartTime != 0) {

// End the counting of time if this is the end of high light intensity

HLEndTime = millis();

// Calculate the duration of high light intensity in milliseconds

unsigned long HLduration = HLEndTime - HLStartTime;

// Add the duration to the total time spent with high light intensity

totalTimeHL += HLduration;

// Reset the start and end times for high light intensity

HLStartTime = 0;

HLEndTime = 0;

}

}

//calculating average temperature per hour

TEMPsum =+ TEMP; //add temperature to the sum

samplectr++; //increase sample counter every minute

samplectr2++;

if (samplectr!=NUM\_Sample){

if (HLduration!=0){

lighthour = lighthour + HLduration;

HLduration=0;

}

else if (samplectr==NUM\_Sample && HLStartTime!=0){

hourendtime=millis();

lighthour=hourendtime - HLStartTime;

}

}

if (Firebase.ready()){

if(samplectr == NUM\_Sample) {

float avgTEMP = TEMPsum/NUM\_Sample;

sendFloat(tempPath, avgTEMP);

sendFloat(lightPath, lighthour);

lighthour=0;

TEMPsum = 0.0;

samplectr = 0;

//Reset to 0 for next hour.

}

//Database Price of electricity or not

if (samplectr2==NUM\_DaySample){

kw= kw+totalTimeHL \*timetokw;

//Price

float price = kw \* costkw;

sendFloat(pricePath, price);

sendFloat(kwPath, kw);

kw=0;

price=0;

}

}

//Send new readings to database

/\*if (Firebase.ready() && (millis() - sendDataPrevMillis > timerDelay || sendDataPrevMillis == 0)){

sendDataPrevMillis = millis();

// Get latest sensor readings

temperature = bme.readTemperature();

humidity = bme.readHumidity();

pressure = bme.readPressure()/100.0F;

// Send readings to database:

sendFloat(tempPath, temperature);

sendFloat(humPath, humidity);

sendFloat(presPath, pressure);

}\*/

//LED COLOR SETTING

//color code #34A853 (R = 52, G = 168, B = 83)

analogWrite(PIN\_RED, 52);

//Conditions for LED green

if (TEMP >= 20 && HUMD <= 60 ) //REPLACE with reasonable conditions for LED to turn green!

{

analogWrite(PIN\_RED, 52);

}

delay (60000);

}

### 

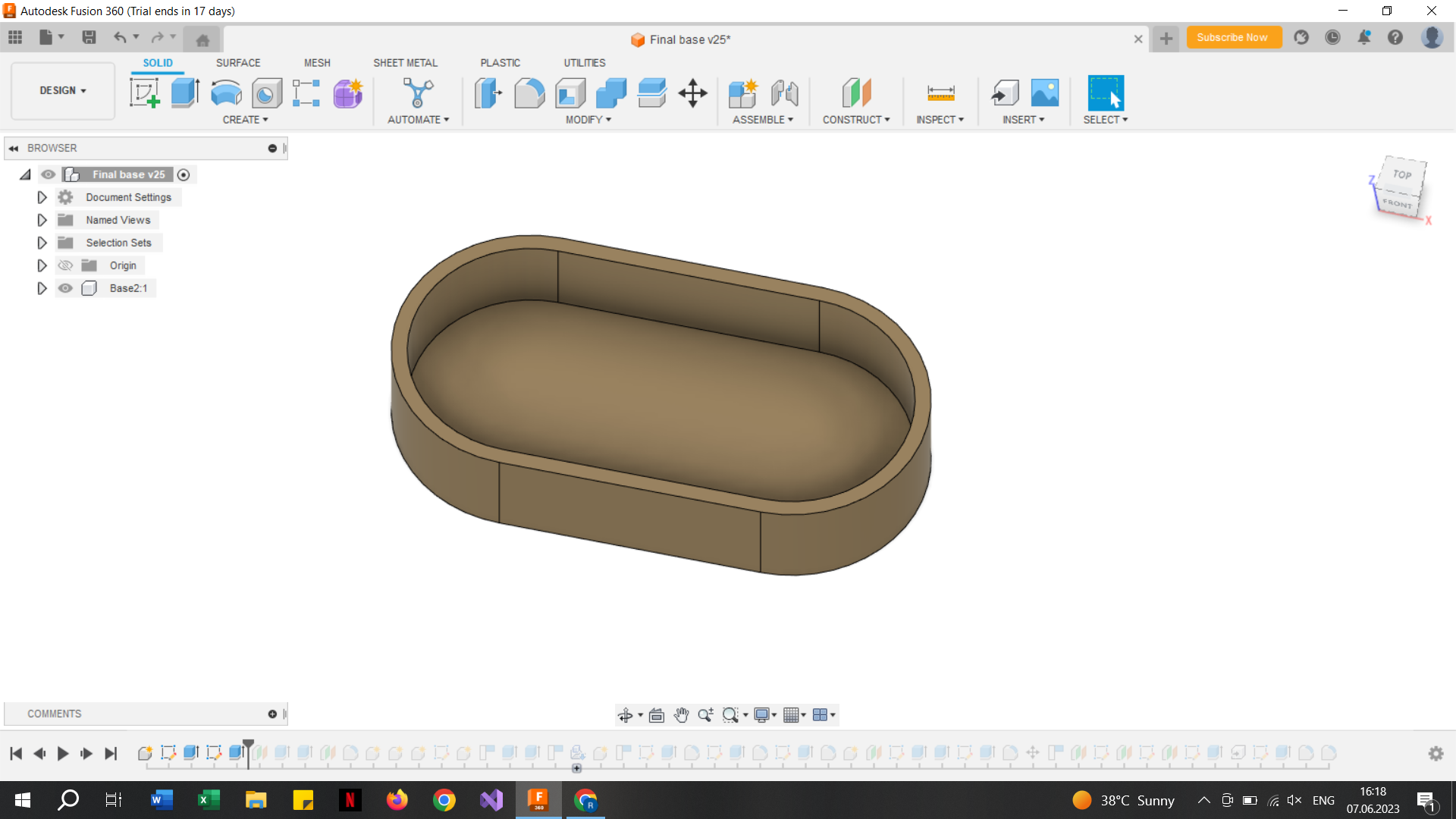
### 

### **Enclosure Design**

We used Autodesk Fusion 360 to create our 3D model for the enclosure design.

We started with a dumbbell-shaped box in order to have the temperature sensor away from the circuits so that the sensor does not detect the heat of the M5Stamp. However, it turned out that the shape’s irregularity made it hard to place it in the designated space.

Therefore, we moved on to a more common shape: an oval. The enclosure will fit the circuit board that contains all the necessary devices and sensors.



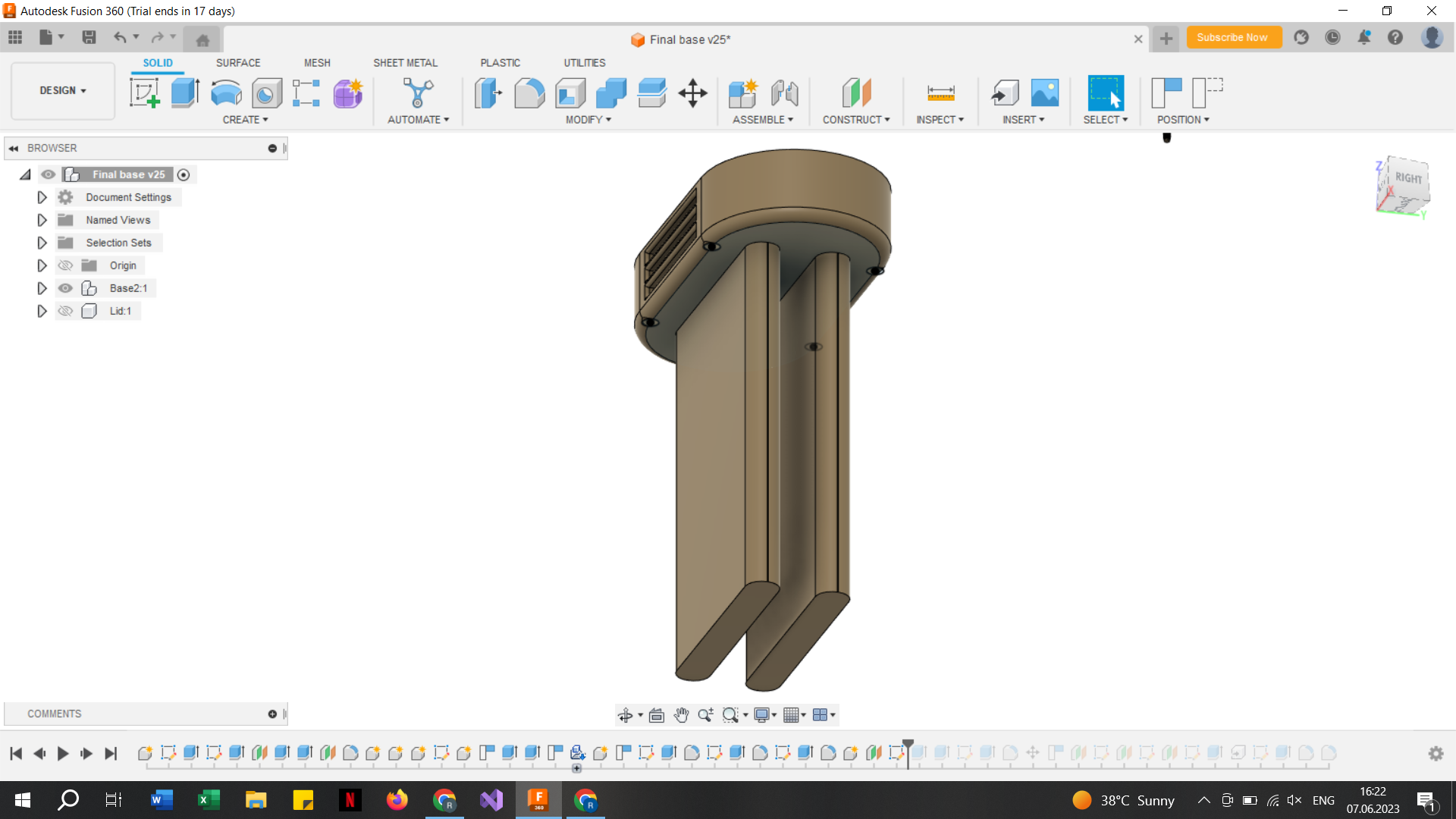
The design has two parts: the base and the lid.

The base has an oval shape with two long legs coming out of its back in order to be able to place it in the designated place: above the wardrobe in the rooms.

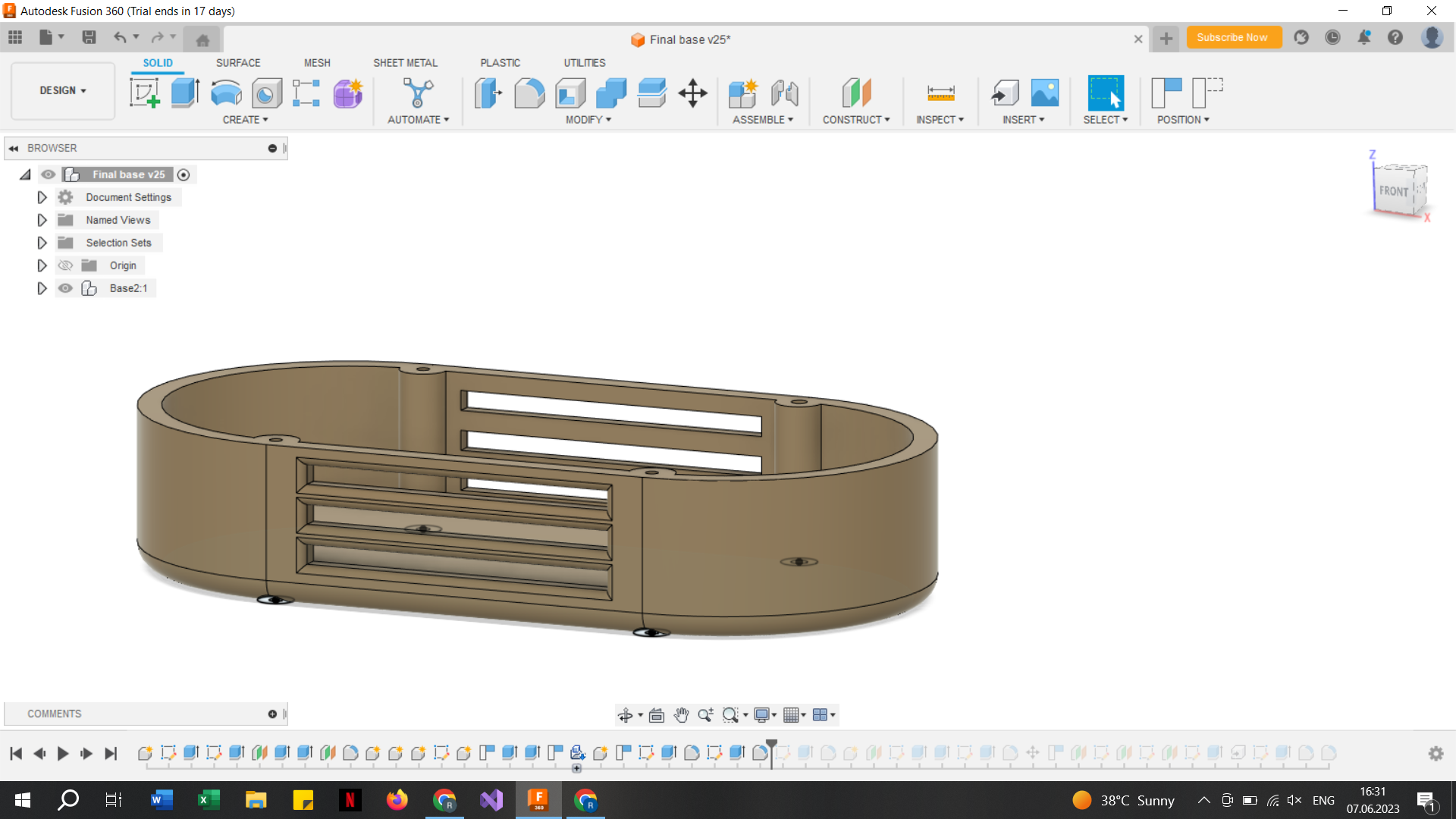


We measured the gap between the holes and the edges to be 2.5cm;

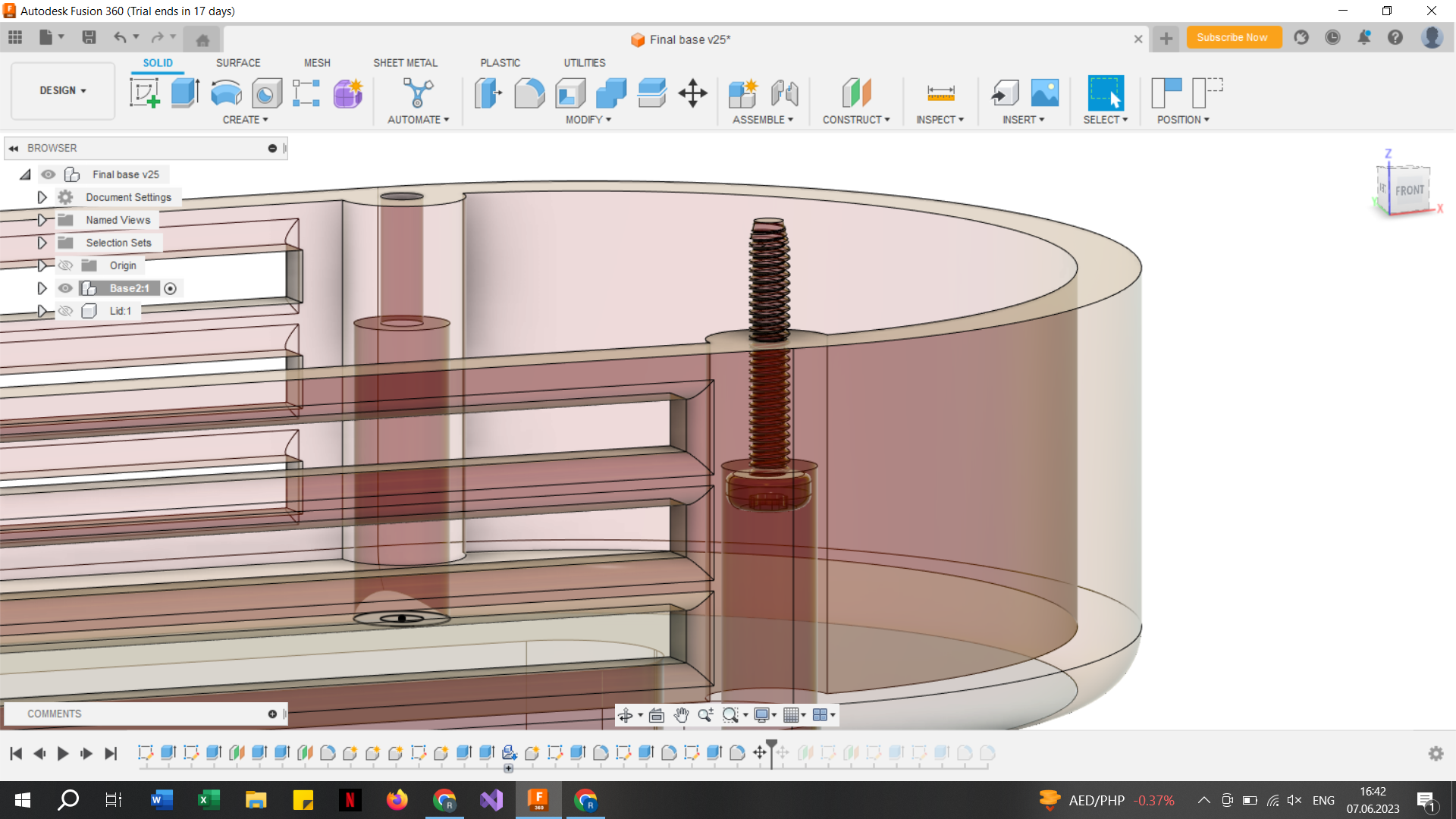
The gap between the holes themselves and the width of the holes is 1cm, in order to create the pattern on the back of our enclosure.



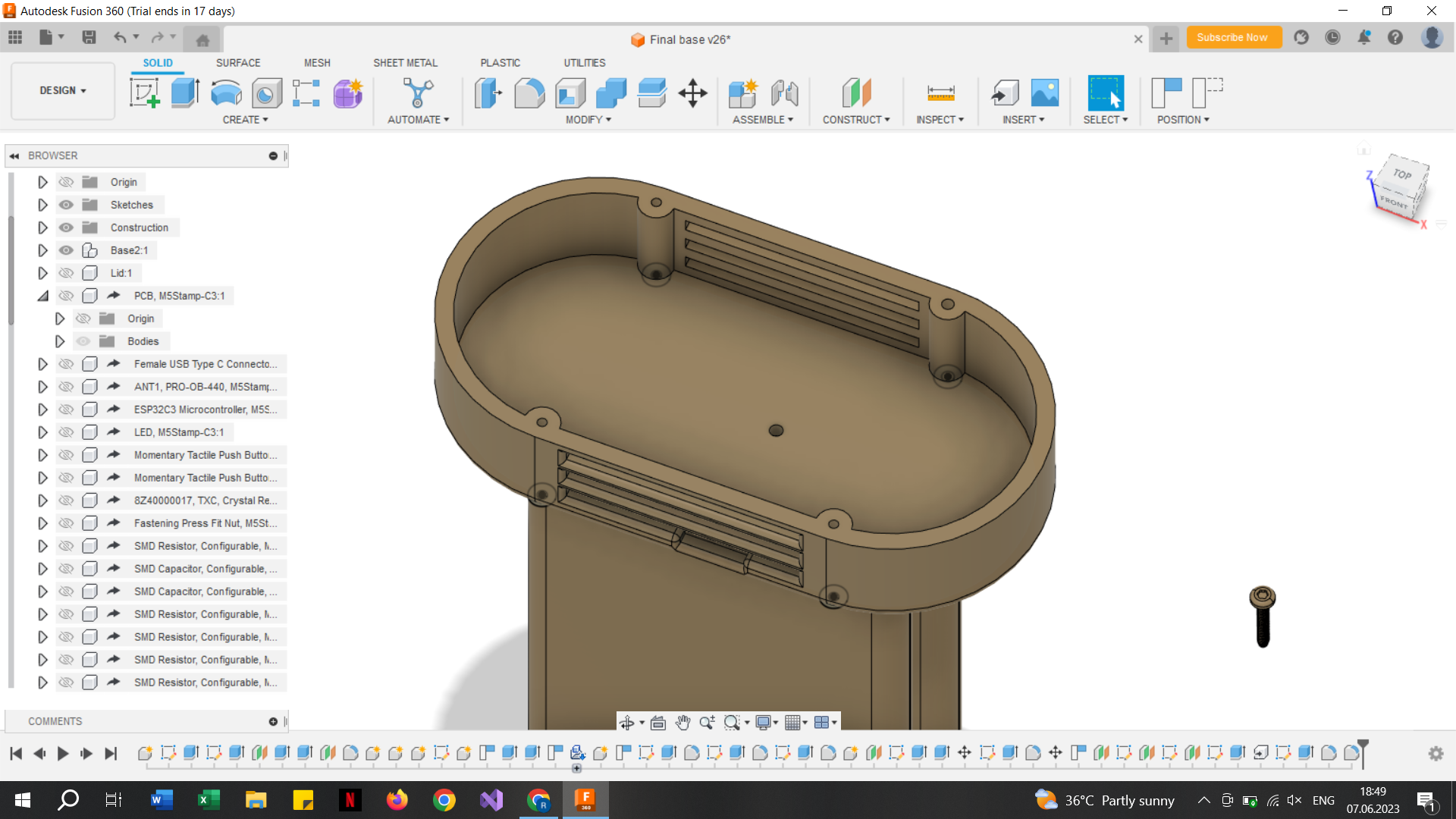
In addition to the legs, we added ventilation holes to our design to facilitate air flow for the circuit board and ensure that the room temperature is registered correctly by the temperature sensor.



In addition, we added screw holes in order to be able to attach the lid to the bottom. We decided to use 12mm M2 screws in order to piece the two parts together. Because the bottom part was taller than 12mm we had to make a bigger hole for the head of the screw to go in. The screw hole is 1.6mm in diameter to be able to put threads in it later.

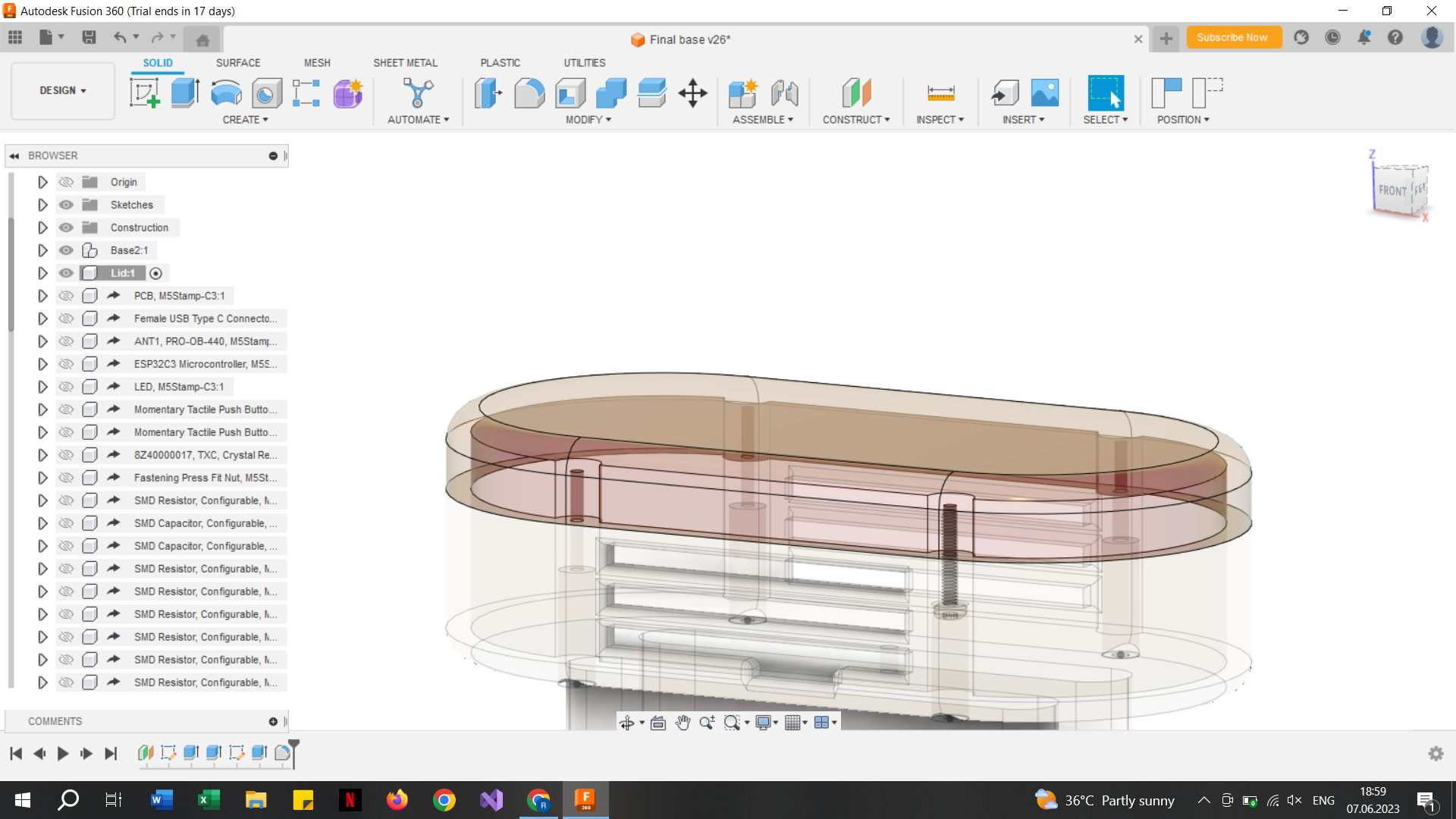


Because our enclosure will rest vertically, we had to come up with a solution to fix the circuit board in place. Therefore, we decided to add a hole in the bottom part of our design, between the two legs, which coincides with the hole that already exists in the M5Stamp. This will allow us to add another M2 screw that will fixate the circuit board onto the bottom part of our enclosure.



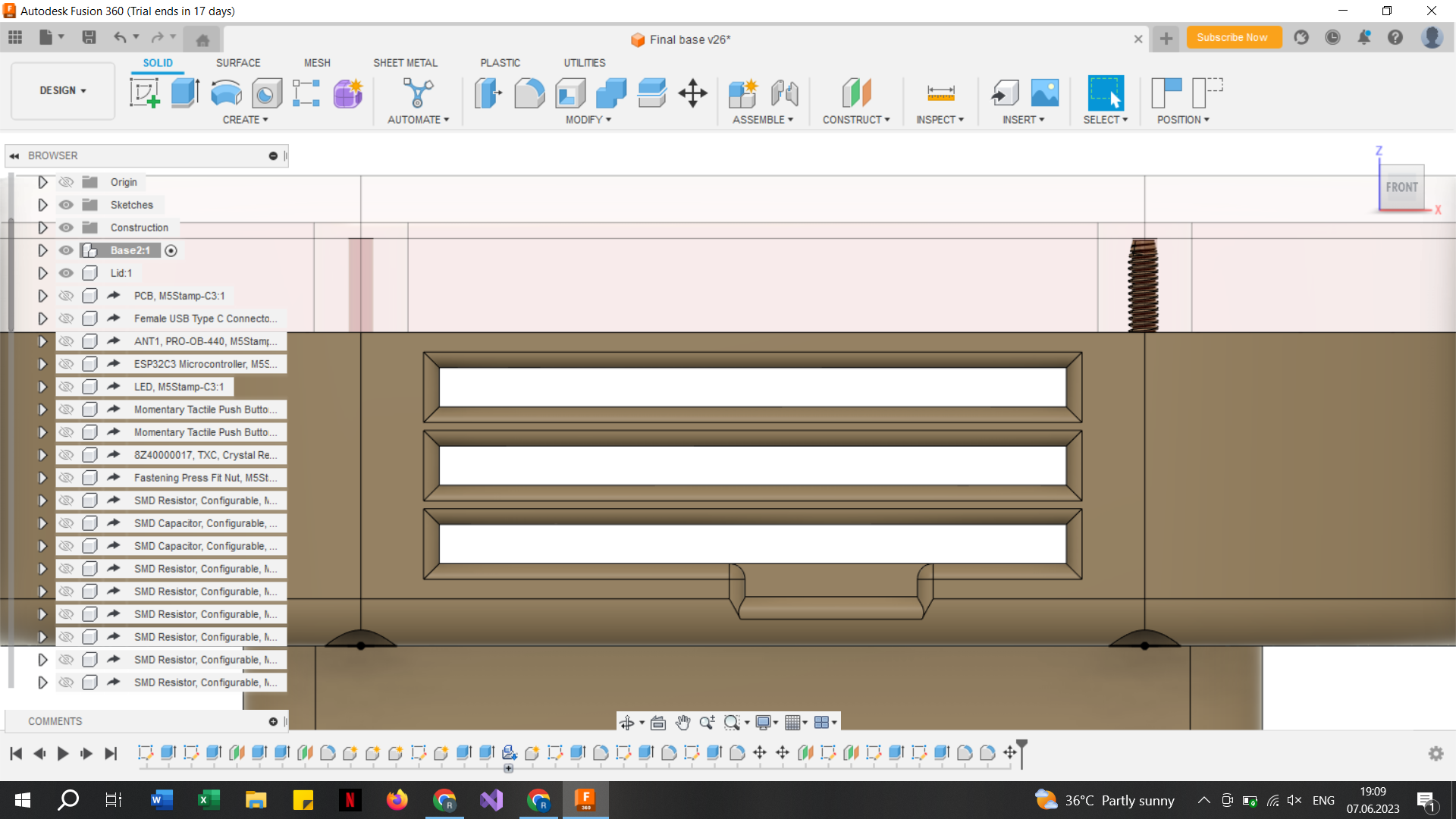


For the lid, we chose a transparent material in order to facilitate a way for the light to reach the light sensor without exposing it. The lid has the same shape as the main body, however, it is less tall. We made sure that the lid fits and that the holes for the screws aligned with the ones on the base of our design.

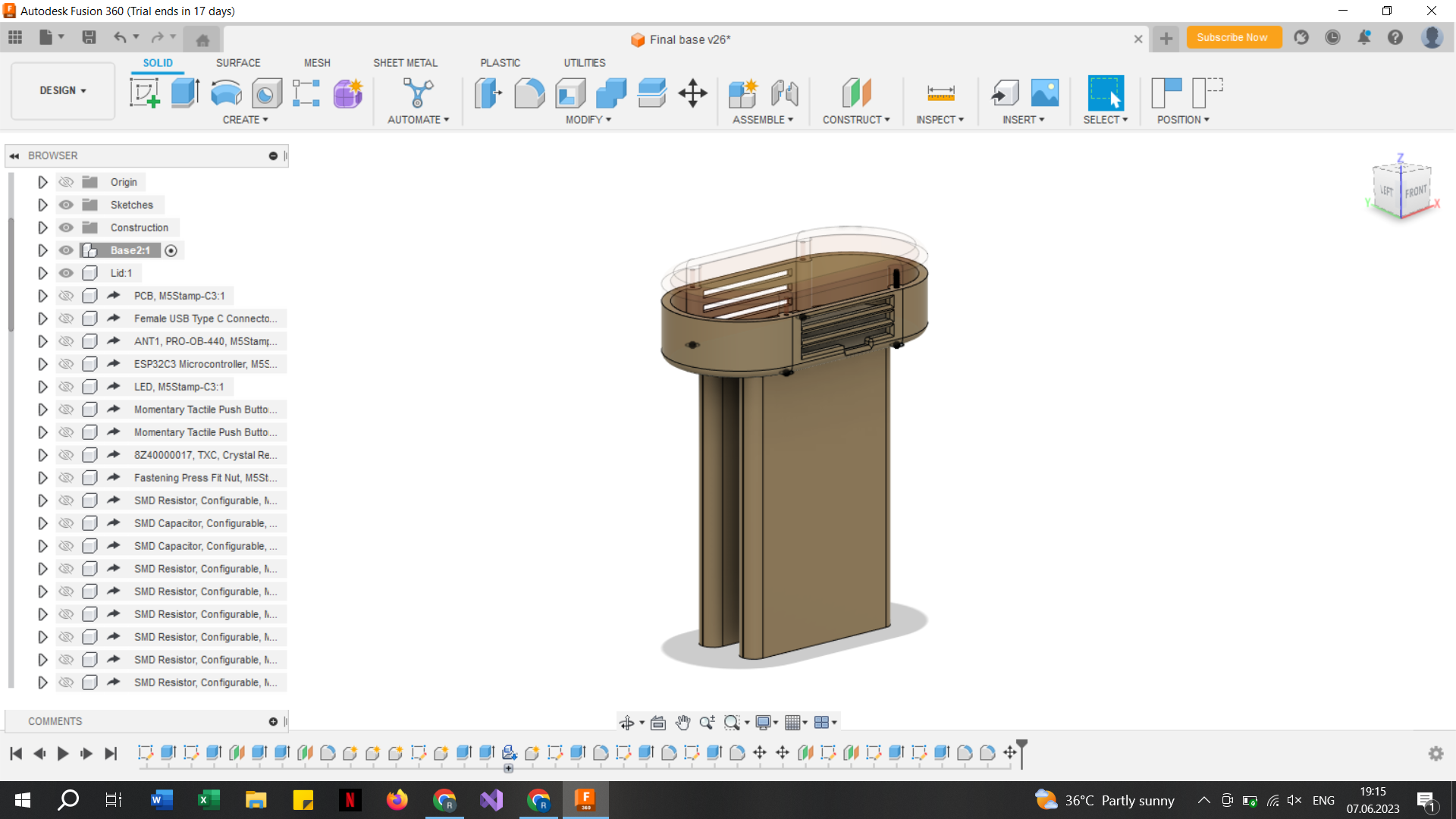


During our designing process, we faced a few challenges. One of them was the shape of the box and the placement of it in the room. We decided on this specific place as it made the most sense for the purpose that our device was designed for.

It was particularly hard to coordinate with the circuit board design from EasyEDA (as it is not so easy). The process of importing the design with the right dimensions was not straightforward and it took a while to figure out. But in the end, we managed to import it, without the 3D version of the M5Stamp. The 3D model of the M5Stamp was imported from grabcad.com in order to find the correct placement for the USB-C hole.



In the end, we managed to finalize the design with all the basic components that it needs to function properly and serve its purpose.



There are definitely aspects of the design that could be improved or refined such as a mechanism on the legs that would facilitate the design to better fixate into the designated place, more adequate dimension for the circuit board (the gap in between the bottom part and the top part might be considered too big compared to the size of the actual circuit board), etc. However, our design should allow the device to be able to hang onto the bands above the wardrobe. And it will be able to detect the light, temperature, and humidity accurately in the room since it is located high above exposing the whole room and the lid was made transparent to light.

All in all, the design should work just fine.

### **References**

<https://docs.m5stack.com/en/core/stamp_c3>

<https://datasheet.lcsc.com/lcsc/2202221130_Worldsemi-WS2812C-2020-V1_C2976072.pdf>

<https://www.vishay.com/docs/84286/veml7700.pdf>

<https://invensense.tdk.com/wp-content/uploads/2020/06/PS-MPU-6500A-01-v1.3.pdf>

<https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf>

<https://easyeda.com/>

<https://pro.easyeda.com/>

<https://jlcpcb.com/capabilities/pcb-capabilities>

<https://www.stovesonline.co.uk/btu-kw.asp#:~:text=Many%20heating%20engineers%20will%20be,central%20heating%20requirements%20and%20loads.&text=To%20convert%20BTU%20to%20kWh%20simply%20multiply%20by%200.000293>.

<https://waterbaths.net/blogs/blog/cooling-capacity-of-recirculating-chillers#:~:text=Cooling%20capacity%20measures%20the%20ability,tons%20of%20refrigeration%20(RT)>.

<https://news.energysage.com/how-many-watts-does-a-light-bulb-use/#:~:text=Taking%20an%20average%2060%20W,and%2043.8%20kWh%20per%20year>.

<https://www.energy.gov/energysaver/room-air-conditioners#:~:text=The%20required%20cooling%20capacity%20for,is%2012%2C000%20Btu%20per%20hour>.

<https://www.electronicssimplified.in/feed/esp32-esp8266-realtime-database-with-firebase/>

<https://esp32io.com/tutorials/esp32-rgb-led>

1. <https://docs.m5stack.com/en/core/stamp_c3> [↑](#footnote-ref-0)
2. <https://jlcpcb.com/partdetail/Worldsemi-WS2812C_2020V1/C2976072> [↑](#footnote-ref-1)
3. <https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6500/> [↑](#footnote-ref-2)
4. <https://www.vishay.com/en/product/84286/> [↑](#footnote-ref-3)
5. <https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/> [↑](#footnote-ref-4)
6. <https://jlcpcb.com/capabilities/pcb-capabilities> [↑](#footnote-ref-5)