GNG1103

Design Project User and Product Manual

Temperature and Humidity Sensor

Submitted by:

SAAND, group D8

Amanda Beraldo Brandao de Souza, 300211045

Nada El Rayes, 300143185

DongYu Wang, 300114760

Aaron MacNeil, 300199522

Shayleen Ghanaat, 300198724

04/11/2021

University of Ottawa

Table of Contents

List of Figures

List of Tables

1 Introduction

This User and Product Manual (UPM) provides the information necessary for food delivery companies and restaurants to effectively use the Temperature and Humidity Sensor and for prototype documentation.

The product was designed with the objective of monitoring the humidity and temperature inside the food delivery package and sending a warning to the user when these factors are not in the ideal conditions. The ideal temperature and humidity range was defined based on tests related to the best quality of the food.

This document is in the public domain and can be used by any student or company.

2 Overview

The purpose of SAAND is to design a temperature and humidity sensor that monitors the temperature and humidity of the food package. It is a main feature to have in order to ensure the customer's food does not get cold during transportation. In case if the temperature or humidity fall out of the approved range, an alert would be sent to the user.

SAAND implemented a cooling fan, in order to regulate the internal temperature of the housing. In regards to the housing, the junction box was chosen because it's waterproof and dustproof features, instead of a 3D-printed housing that is only water-resistant. Another key feature is the two DHT22 sensors that provide more accurate data. Lastly, I2C communication protocol was used to ensure that the warning would be sent to the user.

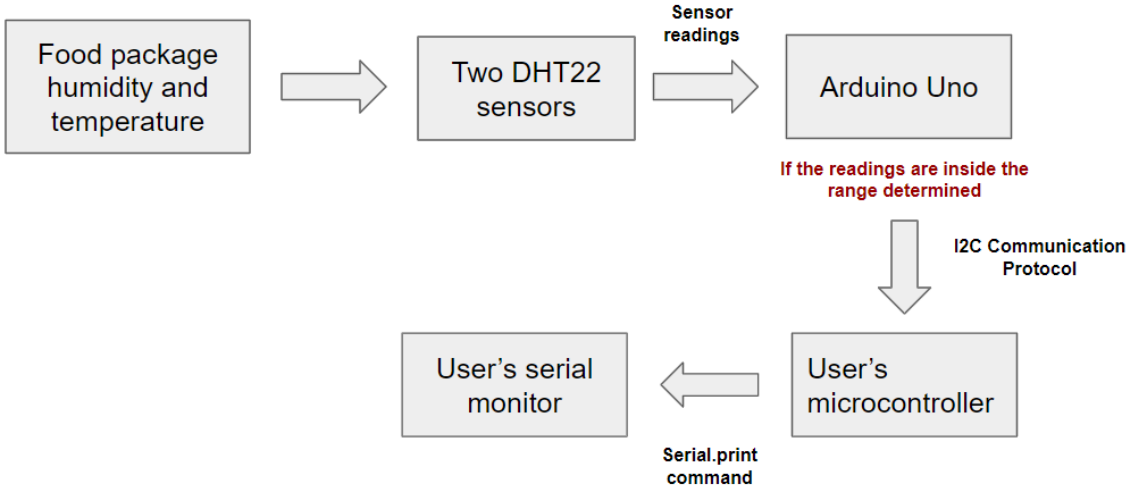

Figure 2.1 Temperature and humidity sensor system block diagram

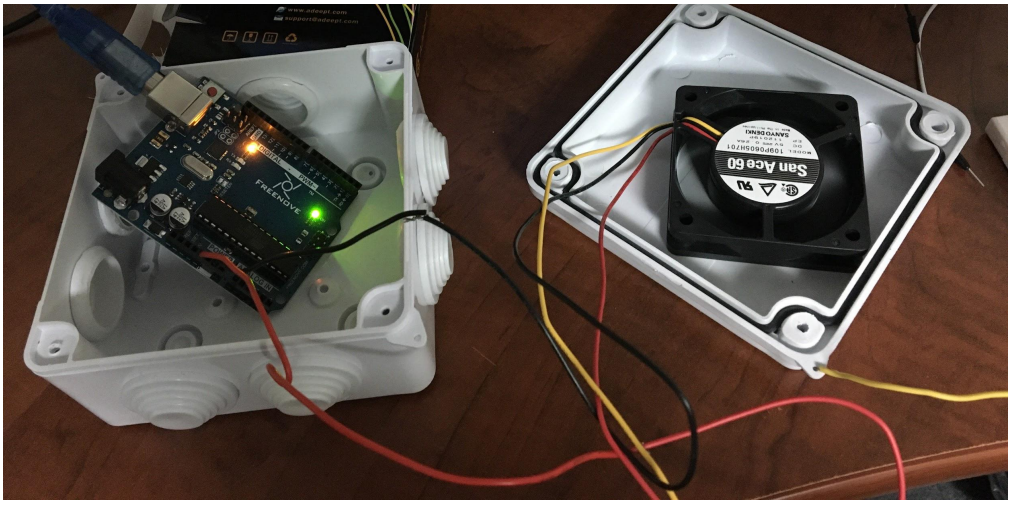

Figure 2.2 Final prototype for fan

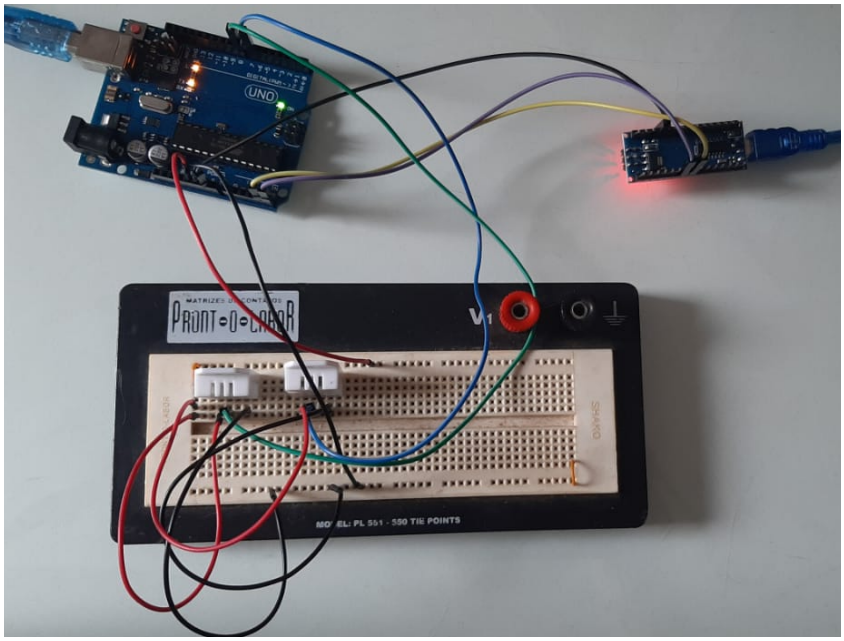

Figure 2.3 Final prototype for sensors and communication between microcontrollers

2.1 Cautions & Warnings

The user should be cautious when working with the product, ensuring that fingers are kept away from the spinning fan blade in order not to injure themselves.

3 Getting started

3.1 Set-up Considerations

The design is composed of two climate sensors, an Arduino Uno, junction box, and a cooling fan.

Figure 3.1 - [From left to right] Sensor DHT22, Arduino Uno, Junction Box and Cooling Fan

The purpose of the two sensors, which are input devices, is to continuously collect temperature and humidity data that will be transmitted to Arduino Uno inside the junction box for further analysis. The user's microcontroller, which is the output device, will receive a warning from Arduino Uno. If the temperature is between 0°C and 19°C or the humidity is above 60%. The cooling fan is used for cooling in the event of excessive internal temperature.

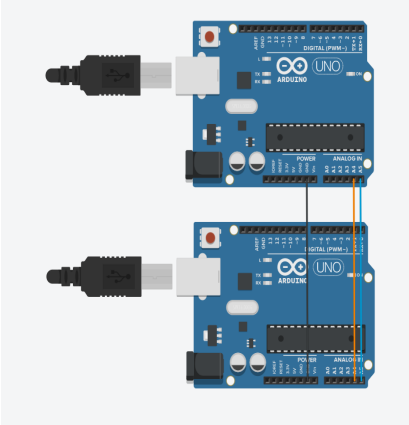

Figure 3.2 - Circuit for the communication of microcontrollers

3.2 User Access Considerations

The users of the product are individuals and companies that work with food delivery systems. One possible restriction is that the microcontroller used to receive information from the Arduino Uno should be provided by the client. The product is tested only with Arduinos and Raspberry Pi devices, therefore it may not work properly when connected to other microcontrollers.

3.3 Accessing the System

The system is turned on when the Arduino Uno is connected to a power source.

3.4 System Organization & Navigation

The following sub-sections provide detailed, step-by-step instructions on how to use the various functions or features of the Temperature and Humidity Sensor.

3.4.1 Description of the System

Two sensors stay outside the housing to properly produce average reading values of humidity and temperature. Then, the collected data is transmitted to the Arduino Uno through wires for further analysis. The data received will be interpreted with the code, and a warning will be sent to the user's microcontroller if the temperature and/or the humidity is not in the accepted range. The fan will be on to control the temperature of the components inside the housing box.

3.4.2 Functions/Features I

Two DHT 22 sensors are linked to Arduino Uno, so that we could have an averaged reading value, not just one sensor value.

3.4.3 Functions/Features II

The cooling fan prevents the inside of the housing from overheating when exposed to sunlight for long-term.

3.4.4 Functions/Features III

The housing box is made of water-proof materials. After the waterproof test, it is concluded that our design has high-quality waterproof and sealing properties.

3.4.5 Functions/Features IV

If the temperature and/or the humidity are not in the accepted range, Arduino Uno will send a warning to the user's microcontroller.

3.5 Exiting the System

The product can be turned off by disconnecting it from the power source.

4 Using the System

4.1 Powering the system:

The arduino is supplied by 5volts DC and can be powered by the drones power supply or by connecting the usb cable to the comport.

Connect 5volts DC and GND to the Arduino uno's 5V input and GND pins.

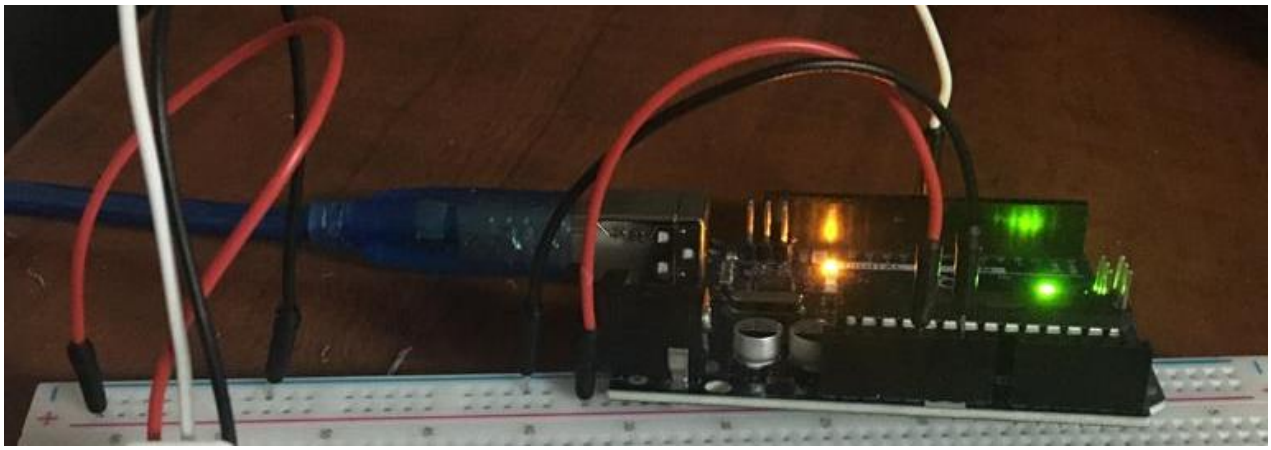

Figure 4.1 Powering the system

4.2 Mounting Arduino in the junction box

The arduino sits inside the junction box, fastened to the bottom floor of the junction box by M3x0.5 bolts.

1. Place the Arduino uno inside the junction box (see picture 1)
2. Fasten it to the bottom of the junction box(5) by threading M3x0.5 threaded bolts through the mounting holes in the Arduino (6).

4.3 Fastening the lid:

The junction box consists of 2 pieces: a main body and a lid.The assembly is fastened together with the included screws at 4 mounting holes as shown.

1. Place lid (1) of junction box on top of main body (2) of junction box
2. line and up the 4 mounting holes (3);
3. thread screws (4) through mounting holes (3) , fastening junction box top to junction box main body.

4.4 USB computer connections:

In order to view the sensor data readings , you will have to connect both arduinos to your computer via usb cables.

Connect both arduinos to the computer via usb cables to the usb port on each individual Arduino (see pic 2).

4.5 Arduino IDE:

The Arduino IDE allows the user to view the sensor data from the arduinos , as well as make any necessary edits to the code.This is useful when debugging the code, and validating sensor readings with another instrument.

Open Arduino IDE on your computer (see Figure 4.1.2).

Figure 4.2 Arduino IDE

1. Open the master code: CTRL +O > click file “master code” (highlighted in grey)> open
2. Open slave code: CTRL +O > click file “master code” (highlighted in grey)> open
3. Select the comport you will be using for the master Arduino : tools> port > com#(Arduino uno) . Note the comport number will be different for your computer, just choose one that shows the Arduino uno is available).
4. Press upload(red circle, see pic 4)

5. Select the comport you will be using for the slave Arduino : tools> port > com#(Arduino uno) . Note the comport number will be different for your computer, just choose one that shows the Arduino uno is available).
6. Press upload(red circle, see pic 4)

NOTE: Ensure that you do not select the same comport for both the slave code as the master code. For example: if you select port8(Arduino uno) for the master code then press upload, then you must select a different port- ie port7(Arduino uno)- for the slave code then press upload.

4.6 Viewing sensor data on the serial monitor:

The serial monitor allows the user to view the sensor data from the arduino. This is useful when debugging code

1. On the master code in arduino IDE, open the serial monitor using the command CTRL+SHIFT+M to see the temperature and humidity data.

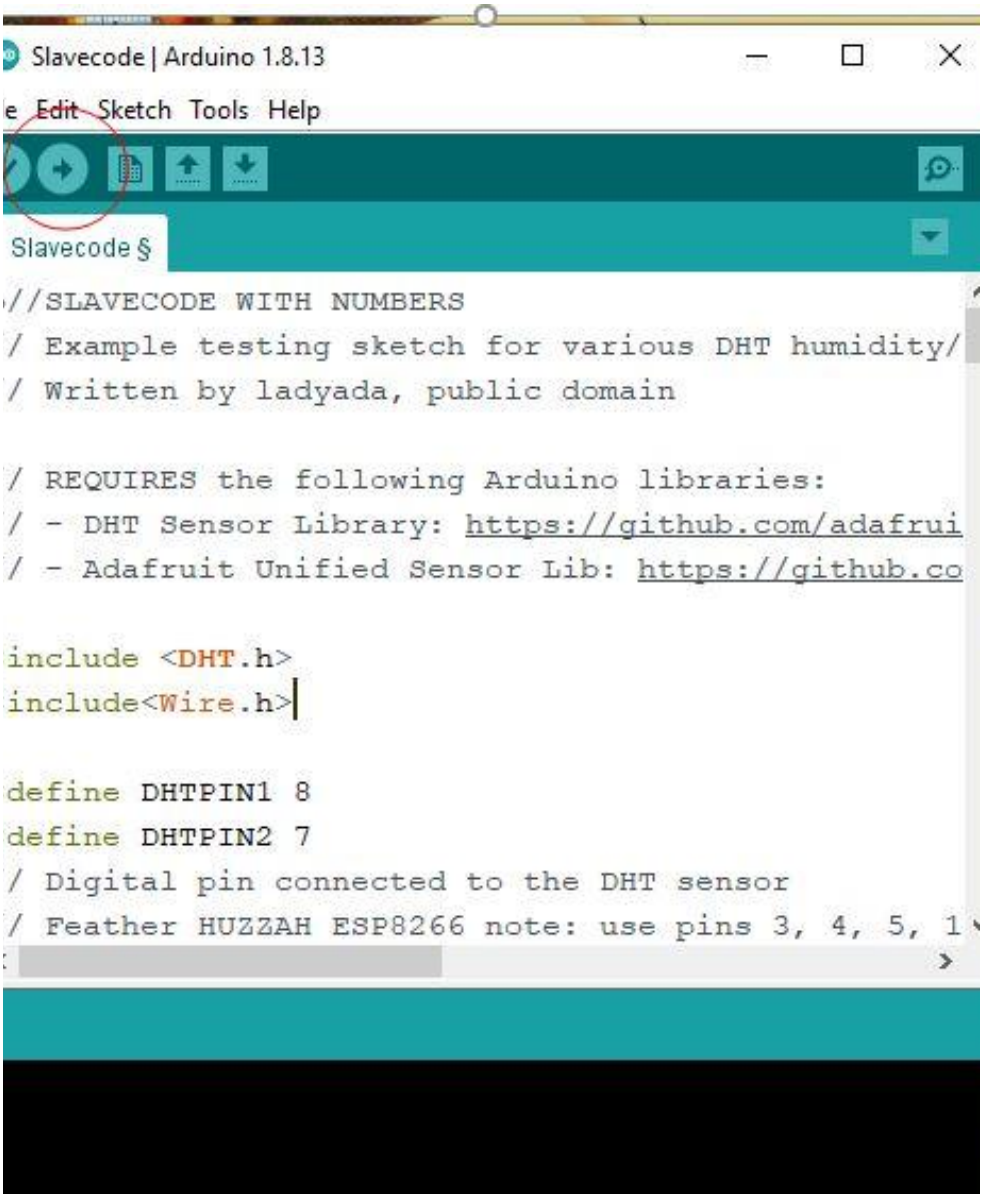

Figure 4.3 Uploading code

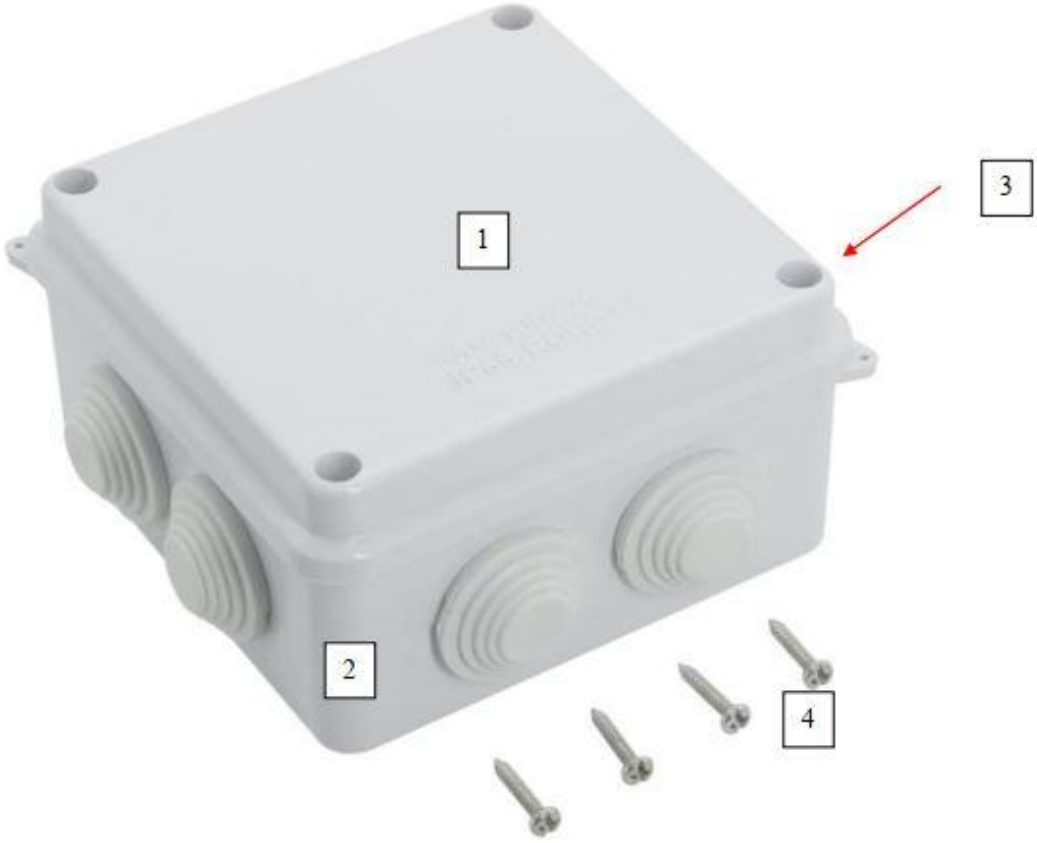

Figure 4.4 Assembling the junction box

Figure 4.5 Arduino in Junction box

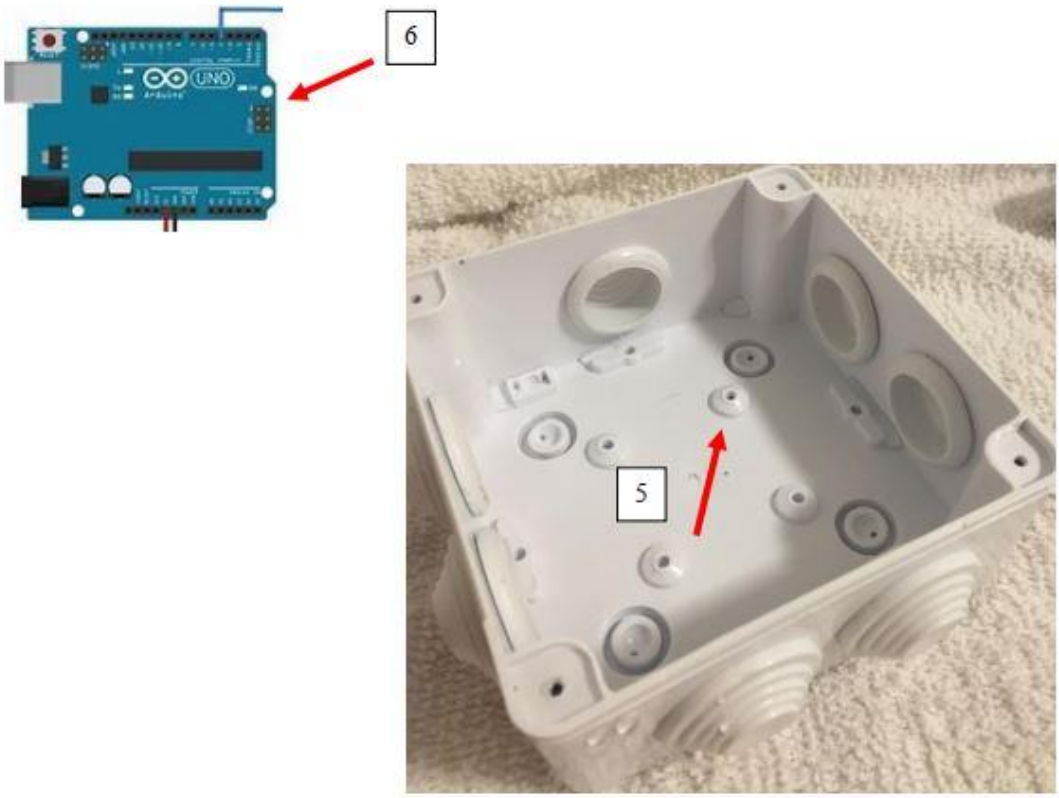

Figure 4.6 Fastening the arduino to the junction box

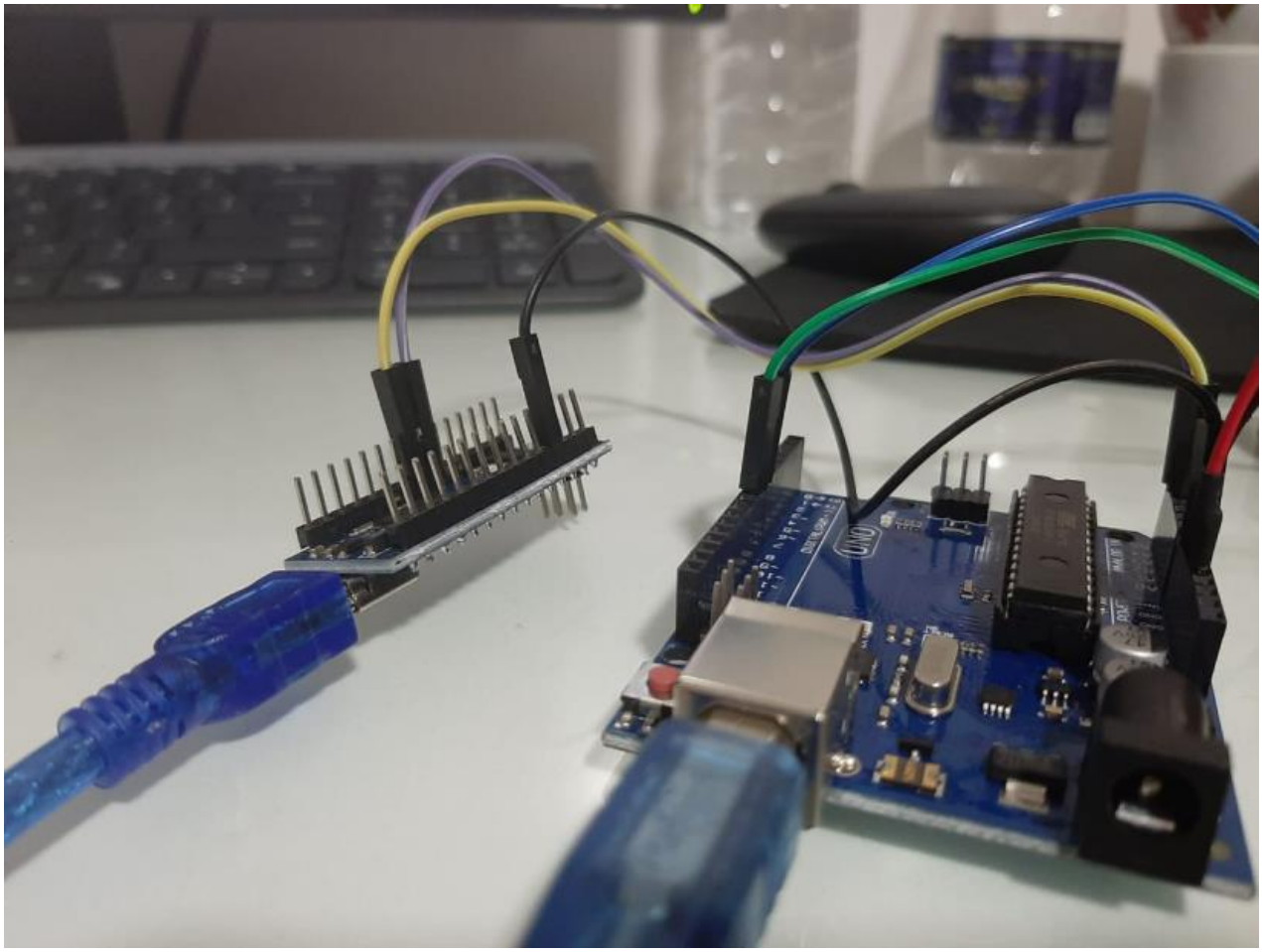

Figure 4.7 USB connected to computer

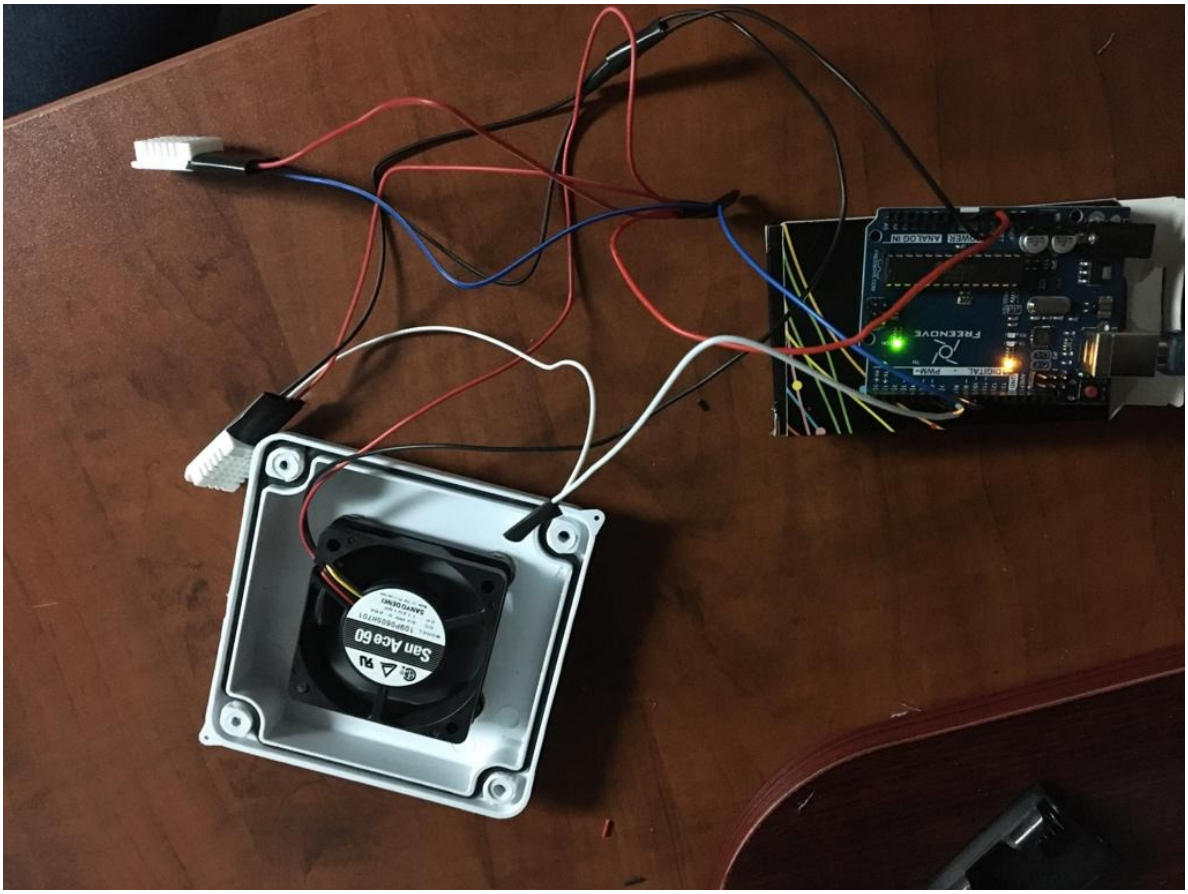

Figure 4.8 slave arduino unit in action

5 Troubleshooting & Support

5.1 Error Messages or Behaviors

5.1.1 Sensor readings

First, make sure that the sensors are properly connected to the Arduino Uno. Figure 5.1 shows the ideal connections of the pins.

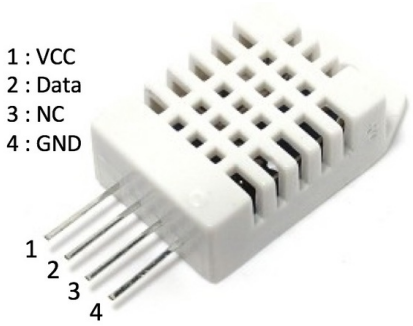

Figure 5.1 DHT22 Sensor pinout

The error message “Failed to read from DHT sensor!” is shown when the readings received from the sensors are not considered numbers by the Arduino Uno. This can occur if the libraries of the device are not correctly updated. It is recommended that the user update the libraries <DHT.h> and <Wire.h> to solve this error.

5.1.2 Communication between microcontrollers

Check if pins A5, A4 and GND of the Arduino Uno that receives information from the sensors are connected to pins A5, A4, and GND, respectively, of the second microcontroller. If one of the wires is disconnected or in the wrong position, unplug both Arduinos from the power source and connect again pins A5, A4 and GND of the Arduino Uno to pins A5, A4, and GND, respectively, of your microcontroller.

5.2 Maintenance

Avoid submitting the product to frequent mechanical stress (Eg, dropping, shaking). The components inside the housing should not get in contact with water.

5.3 Support

Emergency assistance can be obtained by contacting the system support coordinator using the following email:

Amanda Beraldo Brandao de Souza
abera050@uottawa.ca

Your email should contain your name, a detailed description of the problem and a picture of the subsystem with problems (the circuit and the serial monitor, the fan and its connections or the housing).

6 Product Documentation

6.1 Electrical components

6.1.1 BOM (Bill of Materials)

Table 6.1 - Bill of Materials (BOM)

No	Item	Qty	Unit cost ($)	Total cost ($)
1	DHT22 AM2302 Digital Temperature Humidity Sensor	2	6.95	13.9
2	Arduino Uno	1	32.99	32.99
3	LeMotech ABS IP55 Junction Box	1	14	14
4	Jumper wires	40	0.07	2.95
5	Resistors	5	0.16	0.79
6	Cooling fan 109P0605H701	1	14.32	14.32
	Total Cost (without tax and shipping):	78.95$		

6.1.2 Equipment list

Our product “SAAND” is a physical comprehensive prototype composed of two sensors DHT22, one microcontroller Arduino Uno, a housing LeMotech ABS IP65 Junction Box and a cooling fan San Ace 60 103P0605H701.

Critical components for the sensor

- 2×sensors DHT22

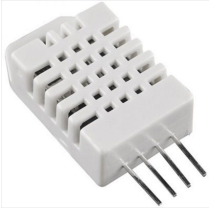

Figure 6.1 DHT22 Sensor

Critical components for the microcontroller

Arduino Uno

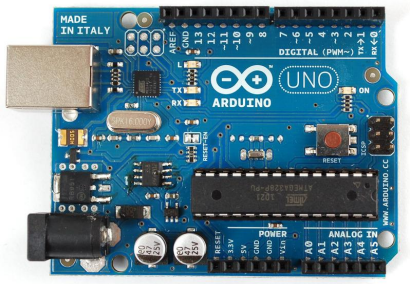

Figure 6.2 Arduino Uno

Critical components for the housing

1 Housing LeMotech ABS

Figure 6.3 Junction box

2 Cooling fan [San Ace 60 103P0605H701.]

Figure 6.4 cooling fan

6.1.3 Instructions

Mechanical:

The Arduino uno sits inside the junction box, fastened to the bottom floor. For the housing it has been suggested to use the 3D printer, it is true that using 3d printing is cost efficient. However the housing is not waterproof.
Choosing the Junction box as a housing has made the product dustproof, waterproof and super shock resistance.

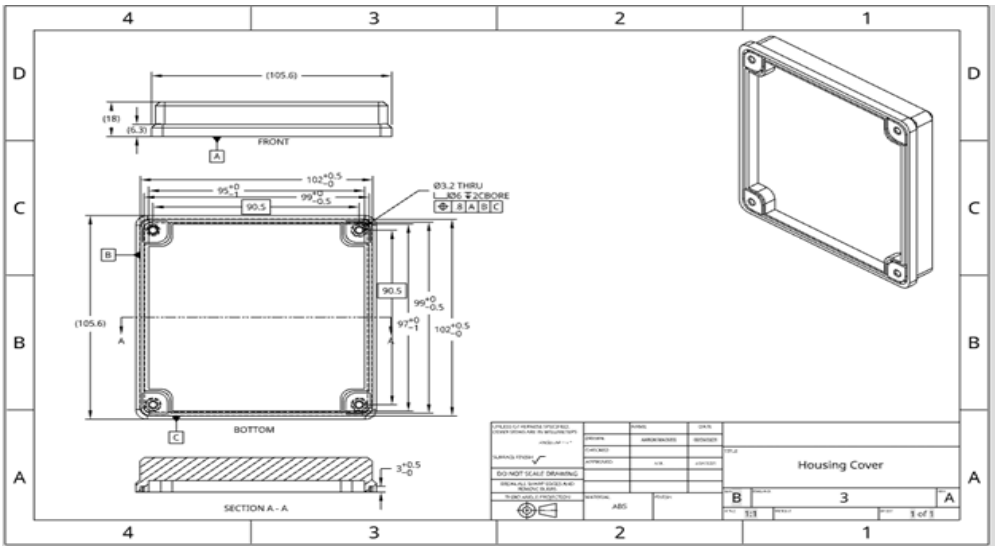

Figure 6.5 - Module Housing Cover technical drawing

Connecting DHT22 sensors to Arduino Uno:

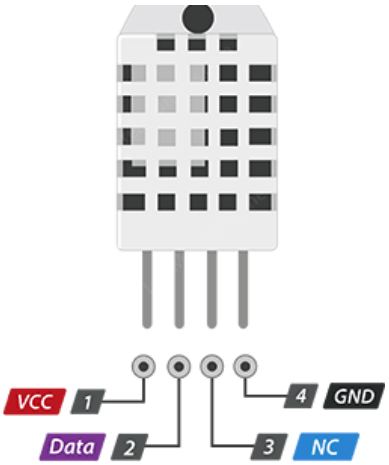

Figure 6.6 - DHT22 sensor

VCC pin supplies power for the sensor connecting it to 5V in the arduino.
Connecting the Data pin to a digital pin 8 and another data pin for the second sensor to pin7 and connect the GNG to the ground.

Join the Fan to Arduino Uno:

We need to supply +5V and GND to the power and ground rails on the breadboard. Tie the source pin to GND, the gate to Uno pin 2, and the drain to the black wire on the fan. The red wire of the fan gets connected to the positive rail on the breadboard.

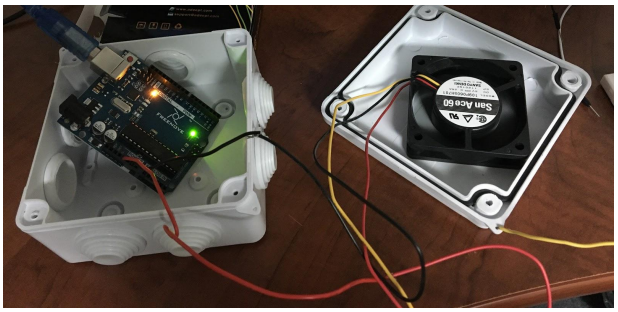

Figure 6.7 - Arduino Uno inside the housing linked with the cooling fan

The sensors will stay outside the housing to properly produce data of humidity and temperature, so it should be connected to the Arduino Uno through wires. A waterproof tube will connect the wiring from the microcontroller Arduino Uno that is inside the waterproof box to the sensor.

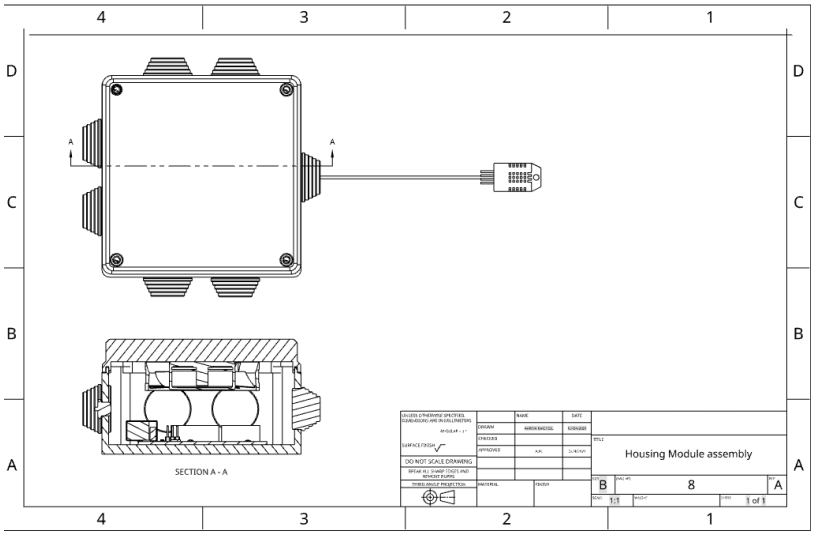

Figure 6.8 - Housing Module assembly drawing

A hole has been cut in the side of the housing and fed the enclosed wires out of the housing, and placed the sensor inside the takeout container.
The module has been mounted to the side of the locking mechanism so that the sensor/module can travel with the package throughout all steps of the delivery process and continuously monitor the temperature/humidity of the content.

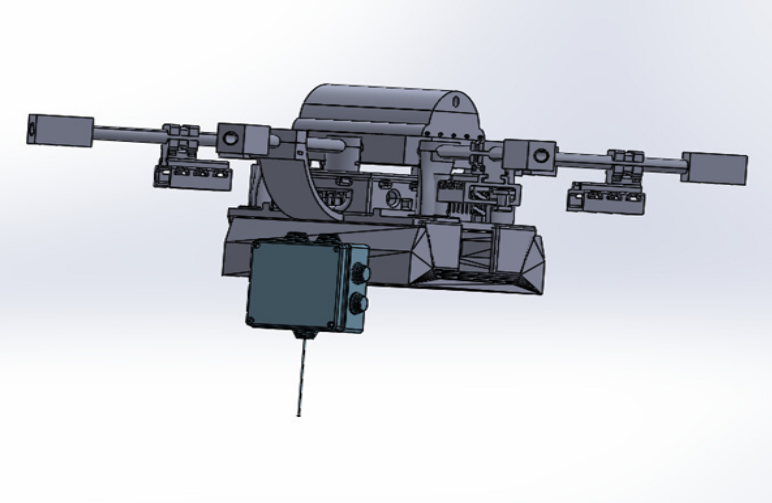

Figure 6.9 Module mounted to Drone Assembly - side view

Software:

The sensor that was initially used was Sensirion AG SHT31-DIHIH8120-021-001S-P2.5KS. When performing the tests, it was noticed that the sensor was not ideal for the project since the small size makes it hard to solder it with a regular soldering iron. To overcome this challenge it has been decided to use the sensor DHT22 that is less precise but is expected to be equally reliable and consistent in providing data.
The initial code was written for only one DHT22 sensor.
Working with one sensor gives the individual's opportunity to test the code for reliable data readings. Afterwards it can rewrite the code for two DHT22 sensors.

```ino
// Initialize DHT sensor.

DHT dht= DHT(7, DHT22);
DHT dht2 = DHT(7,DHT22 );

void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));

  dht.begin();
}

void loop() {
  // Wait a few secobetween measurements.
  delay(500);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

Figure 6.10 - Code used for the sensor DHT22

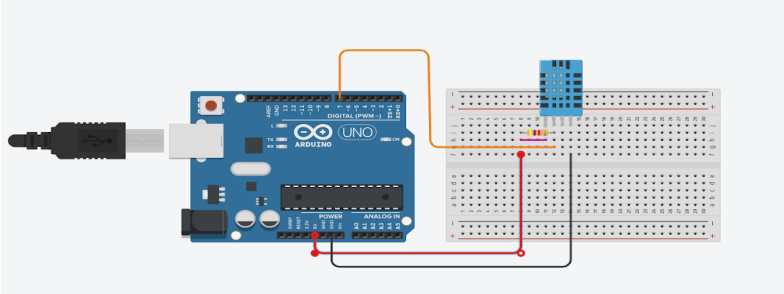

Figure 6.11 - Simulation of the circuit using an Arduino Uno, a breadboard, a resistor and one DHT22 sensor

The code is to implement 2 sensors, and to report back the average values of the temperature and humidity readings.

An average was defined between the readings of the two sensors as being the value that activates the communication between microcontrollers. This activation will only happen when this average reading is between 0°C and 19°C and when the humidity is above 60%.
For getting these values a sample of specific foods (examples: Ice cream, toast, tea etc) were measured using a thermometer. To define this range, most of the ideal temperatures tested were above 19°C and below 0°C. The humidity value was obtained by measuring the humidity when drinks would spill in a controlled environment.

FINAL_MASTERCODE_20210324

```
#include <Wire.h>

void setup(){
  Wire.begin();
  Serial.begin(9600);
}

void loop(){
  //request data from slave
  Wire.requestFrom(2,20);
  String string, string1, string2;// initialize strings
  do{
    char c = Wire.read();
    string = string + c; // whole string temp + hum
    string1 = string.substring(0,8); // string for temperature
    string2 = string.substring(9);// string for humidity
  } while(Wire.available());{ //print the warnig on serial monitor
    Serial.print("WARNING!");
    Serial.println();
    Serial.println();
    Serial.println();
    Serial.print("Temperature: " );
    Serial.print(string1);
    Serial.print("°C");


    Serial.println();
    Serial.print("Humidity: " );
    Serial.print(string2);
    Serial.print("% ");
    Serial.println("");
    Serial.println("");
    Serial.println("");

    delay(500);
  }
  delay(500);
  }
```

FINAL_SLAVECODE_20210324

```
//SLAVECODE WITH NUMBERS
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

// REQUIRES the following Arduino libraries:
// - DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor Lib: https://github.com/adafruit/Adafruit_Sensor

#include <DHT.h>
#include<Wire.h>

#define DHTPIN1 8
#define DHTPIN2 7
// Digital pin connected to the DHT sensor
// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
// Pin 15 can work but DHT must be disconnected during program upload.

// Uncomment whatever type you're using!
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321


// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 3 (on the right) of the sensor to GROUND (if your sensor has 3 pins)
// Connect pin 4 (on the right) of the sensor to GROUND and leave the pin 3 EMPTY (if your sensor has 4 pins)
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors.  This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht1(8, DHTTYPE);
DHT dht2(7, DHTTYPE);

#define SLAVE_ADDR 9  //define Slave I2C Address
 //Define Slave answer size


char wy[7];
char ix[8];

float tavg; //average temperature
float havg; //average humidity
```

Figure 6.12 - Code used for two DHT22 sensor

6.1.2 Feasibility analysis

All the components are readily available, usually in online stores or the local equipment store, therefore it will be easy and economical even for future production. Thus, the product is both economically feasible and accessible.

6.1.3 Risk mitigation

Table 2: Risk description and mitigation approach

Risk Description	Mitigation Approach
Temperature sensor fails to operate completely	To solve the issue temperature sensors output voltage should be continuously monitored by an ADC or redundancy can be introduced (i.e use of 2 temperature sensors) and better calibration can be done by comparing outputs of 2 temperature sensors.
Temperature sensor is enabled when calibration data is changing	There can be an issue as the chip can stay in reset or can be stuck in reset loop, thus not allowing the chip to boot and execute the application program
Syntax errors in code	Have another program open which you can refer to, as in most cases the syntax and formatting are the same between different programs.
Unable to connect the sensor to the Arduino	Purchasing another sensor that the individual is familiar with
Fan does not respond	Check the wiring Maybe the fan inner part system has a disconnection

6.2 Testing & Validation

Tests were performed to evaluate the parameters of the prototype previously described. In this section, we describe in detail how the tests were held and what are the results obtained.

6.2.1 Housing Water Test

Figure 6.13 Housing Water Test

Testing Method:

Physical Prototype test

Estimated Test Duration
1 hour

Description of Prototype
Internal convex plate for the circuit board waterproof, protection grade reaches more than IP56, water will not enter even when it is raining

Figure 6.14 Inside of the housing was not affected by the water.

6.2.2 Cooling fan

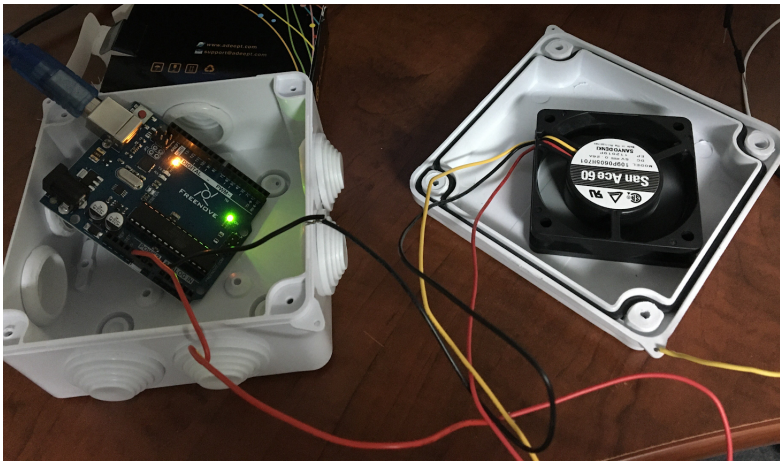

Figure 6.15 Arduino Uno inside the housing linked with the cooling fan

Testing method

Physical Prototype Testing

Estimate Duration Time

1 hour

Description of Prototype

Cooling fan connected to an Arduino Uno

Description of Results to be Recorded and how these results will be used

The Arduino effectively powers the cooling fan for the one hour period analyzed. This information will be used in the implementation of the fan on the final assembly. With the test we can see that the fan has extremely low power consumption and, therefore, can be used in the product's final assembly.

6.2.3 Housing Heating Test

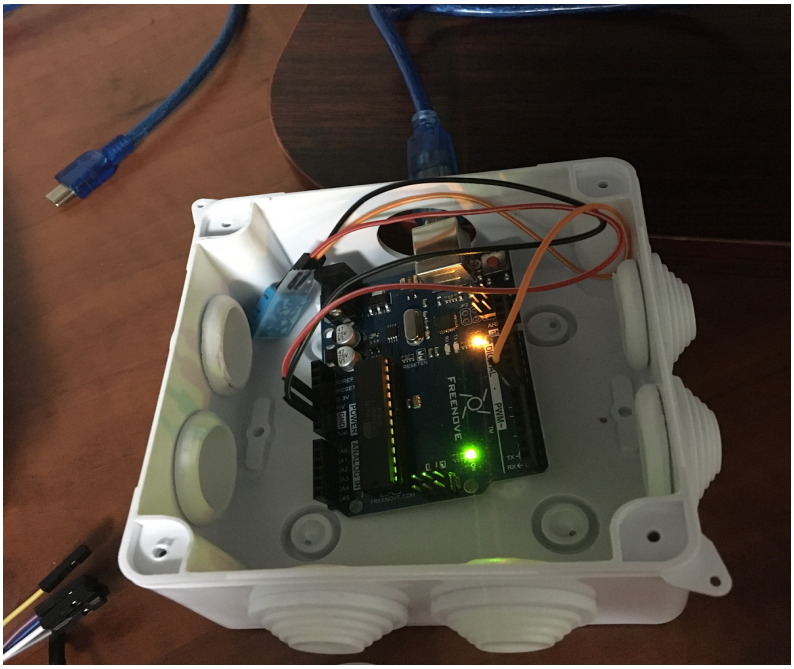

Figure 6.16 Arduino Uno inside the housing for the heating test

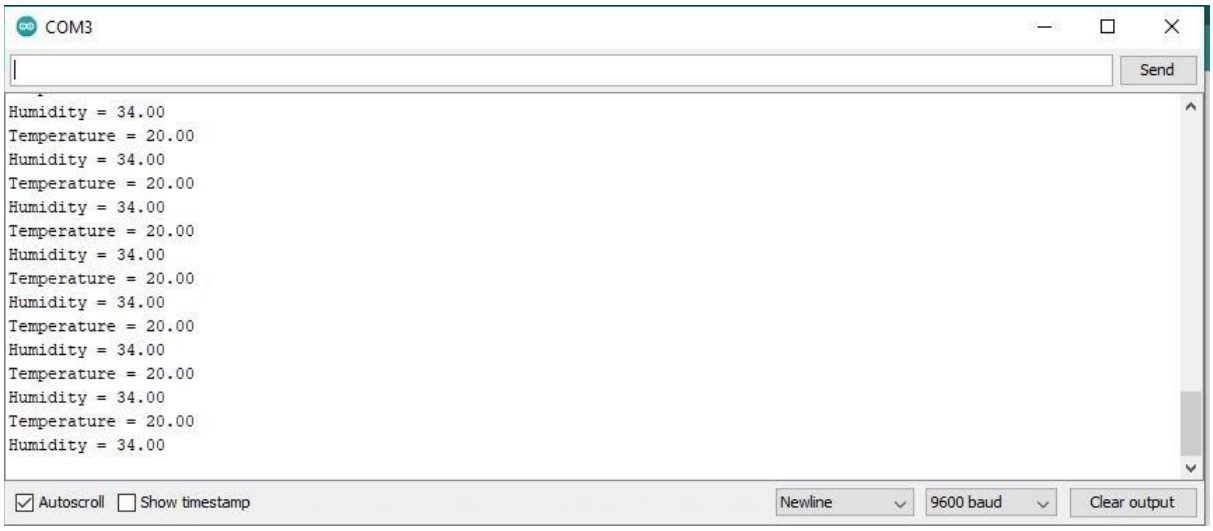

Figure 6.17 - Data obtained from the sensor inside the housing

Figure 6.18 Code used for the house heat test

Testing method

Physical Prototype Testing

Estimate Duration Time

1 hour

Description of Prototype

Conventional materials according to different use environments typically within five years or more,thermal deformation temperature: 40 to 85 degrees.

Description of Results to be Recorded and how these results will be used

After an hour in the closed housing, the sensor data has hardly changed. Therefore, we can see that the microcontroller itself does not overheat. Further testing will have to be done to see how the temperature deviates when the housing is left out in the sun for an extended period of time.
All software has been thoroughly debugged and fully integrated with all operational hardware and software systems.

6.2.4 Code for the sensors

The source code is available in https://makerepo.com/YoungEd/839.gng1103d8saand

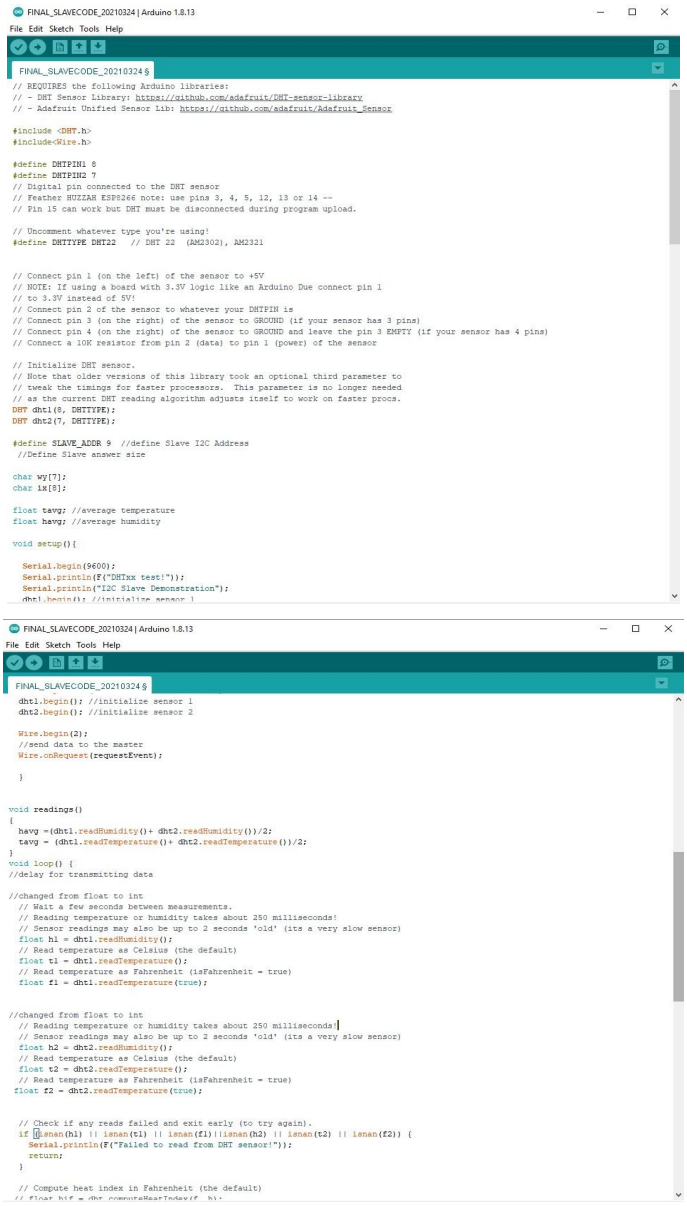

Figure 6.19 Code for the sensor

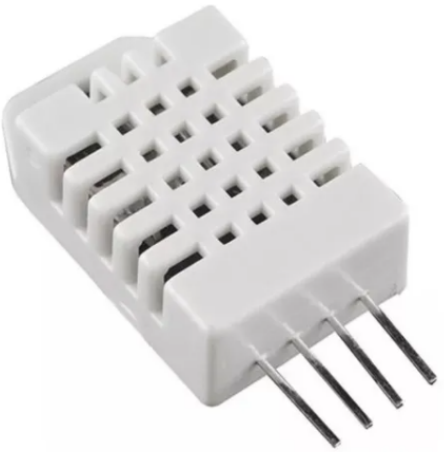

Figure 6.20 Sensor DHT22

Testing method

Analytical Prototype Testing

Estimate Duration Time

1 hour

Description of Prototype

Code that receives the information from the sensor and prints it on the serial monitor. For this code we don't have a specific condition for the transmission, so the communication happens all the time.

Description of results to be recorded and how these results will be used

Accurate readings from the two sensors DHT22. This data will be read in the first Arduino and transmitted to the other microcontroller.

6.2.5 Communication between microcontrollers

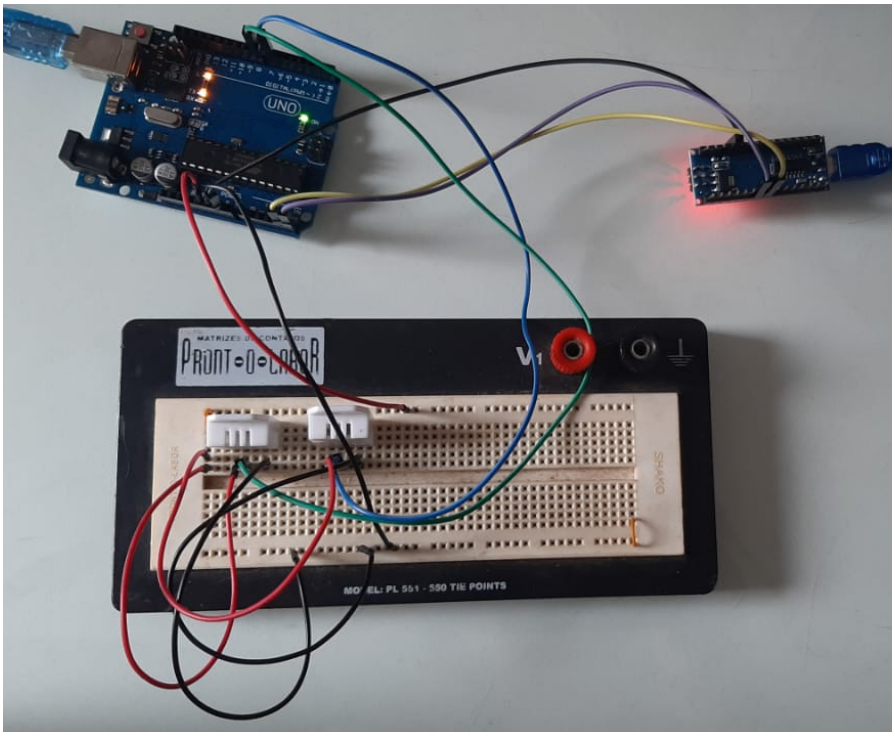

Figure 6.21 - Circuit using two DHT22 sensors, one Arduino Uno and one Arduino Mini

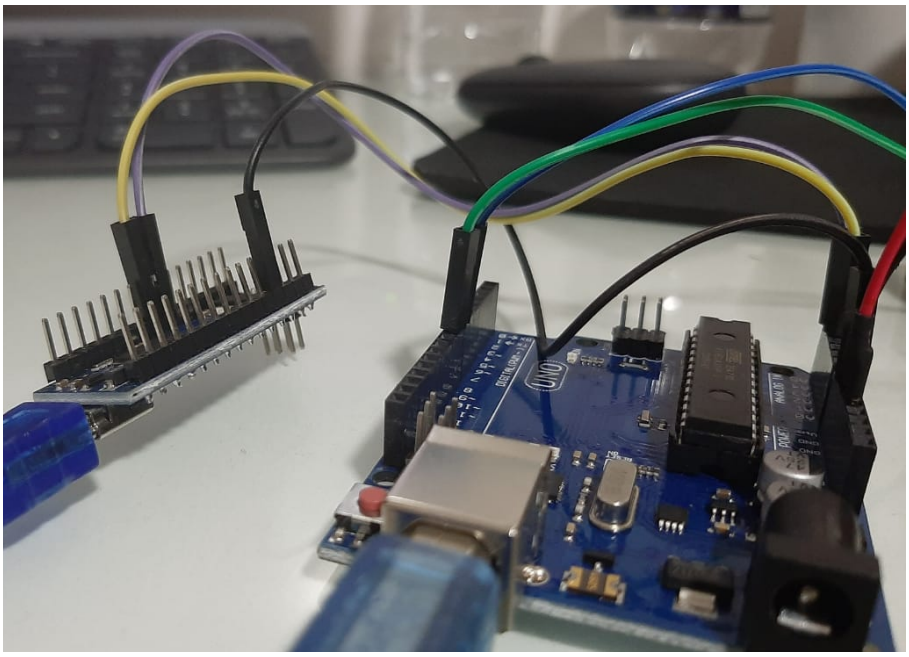

Figure 6.22 Arduino Uno connected to the Arduino Mini using the GND, A4 and A5 pins

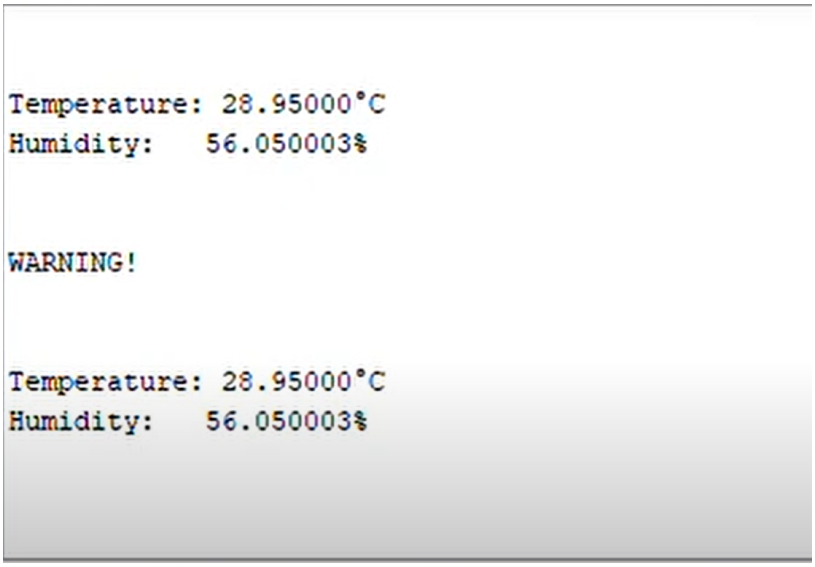

Figure 6.23 - Warning message and temperature and humidity readings on the Arduino's Mini serial monitor

Testing method

Analytical Prototype Testing

Estimate Duration Time

1 hour

Description of Prototype

Code that sends the information from the Arduino Uno to the Arduino Mini using I2C communication

Description of results to be recorded and how these results will be used

Warning and temperature and humidity readings on the Arduino's Mini serial monitor. These results will define if the data and the warning message are effectively transmitted from the Arduino Uno to the Arduino Mini.

7 Conclusions and Recommendations for Future Work

The team has learned a lot of key aspects since the beginning of the course.

The first obstacle we faced was a budget reduction. As a team, we had to see what components are truly worth purchasing since our budget was reduced in half from the initial 100$ to 50$.
The second obstacle we faced was to communicate effectively as a team to ensure everyone is on the same track. We also learned that before purchasing the components, research must be done to see the compatibility of the components. Finally, we have spent hours trying to run a code that was not giving accurate readings, but later we discovered that to run the code the libraries should be first cleared.

If the group had additional time to work on the project, having the circuit soldered to a printed circuit board would be the priority. In the case of a greater budget, a more precise sensor could replace the DHT22 sensors that were used in the design. Another aspect that could be enhanced in the product is the size of the components used since a smaller microcontroller, housing and fan would make the product lighter and more compact for use.

8 Bibliography

The links for components:

Arduino Uno R3 Microcontroller A000073:

https://www.amazon.ca/Arduino-A000073-Uno-REV3-SMD/dp/B00PUOVSYS/ref=sr_1_8?dchild=1&keywords=arduino+uno&qid=1614140517&sr=8-8

DHT22 AM2302 Digital **Temperature and Humidity Sensor** for Arduino:

https://www.buyapi.ca/product/dht22-am2302-digital-temperature-humidity-sensor-for-arduino/

LeMotech ABS Plastic Dustproof Waterproof IP65 **Junction Box** Universal Electrical Project Enclosure White 3.9" x 3.9" x 2.8":

https://www.amazon.ca/gp/product/B075DJDT99?pf_rd_r=XGD0WHWHJ8HAZCF2HT62&pf_rd_p=05326fd5-c43e-4948-99b1-a65b129fdd73&pd_rd_r=8d9ff3fb-768a-45e6-a56a-5be7841c7d08&pd_rd_w=B9NfW&pd_rd_wg=4waMC&ref_=pd_gw_unk&th=1

San Ace Cooling Fan:

https://www.digikey.ca/en/products/category/fans-thermal-management/16?s=N4IgjCBcoLQCxVAYygMwIYBsDOBTANCAPZQDaIArAGwDMADDSALqEAOALlCCAL59A

Reference List

DroneBot Workshop. (2019, June 24). I2C communications Part 1 - Arduino to Arduino. Retrieved April 11, 2021, from https://dronebotworkshop.com/i2c-arduino-arduino/

Lady Ada. (n.d.). DHT11, DHT22 and Am2302 Sensors. Retrieved April 11, 2021, from https://learn.adafruit.com/dht/using-a-dhtxx-sensor#

APPENDICES

9 APPENDIX I: Design Files

All documentation used for this project, from the empathize stage to the prototype testing, is provided on the following page:

https://makerepo.com/YoungEd/839.gng1103d8saand

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Deliverable B - Needs Identification and Problem Statement	PD B	01/31/2021
Deliverable C - Design Criteria	PD C	02/07/2021
Deliverable D - Conceptual Design	PD D	02/21/2021
Deliverable E - Project Plan Cost Estimate	PD E	02/28/2021
Deliverable F - Prototype I and Customer Feedback	PD F	03/09/2021
Deliverable G - Prototype II and Customer Feedback	PD G	03/14/2021
Deliverable H - Prototype III and Customer Feedback	PD H	03/28/2021
Deliverable I - Design Day	PD I	04/08/2021

Presentation Material		
Deliverable J - Project Presentations	PD J	03/26/2021