JAMZ Climate Sensor: Prototype III

Deliverable H

GNG 1103

March 28th, 2021

Ogechi Ahunanya, Olivia Carnegie, Michaela Jones, Zach Léger, Julio Midence

Table of Contents:

Introduction

In this deliverable, we made what was supposed to be our last prototype, but based on our testing and some unfortunate errors we will need to make another one before design day. Over the past two weeks, our team has upgraded the code for the sensor and the sensor casing. For the code, we added time stamps that can range from seconds to hours, since the drone will be flying in rural areas it is possible that trips could last around an hour. For the physical prototype, we made a casing for our arduino, as well as added some improvements to our sensor casing. In this deliverable, we will show everything that we worked on and provide explanations as to why we need to make more modifications to our prototype before design day.

Physical Prototype & Code

Physical Prototype III

For our third prototype, we added some modifications to our sensor casing and we made a casing for the arduino. In our last deliverable, we made our first sensor casing prototype by 3D printing. Unfortunately, the holes for the pegs were too small for the pegs themselves and the prototype was not very stable. In this deliverable, we made sure to fix those issues by making the pegs larger and stronger and making the holes for the pegs the appropriate size. As shown in *Figure 1*, the new sensor casing is very stable and the top casing fits perfectly into the bottom casing. The issue with this prototype is that it was a bit too large and the holes for the screws were way too big.

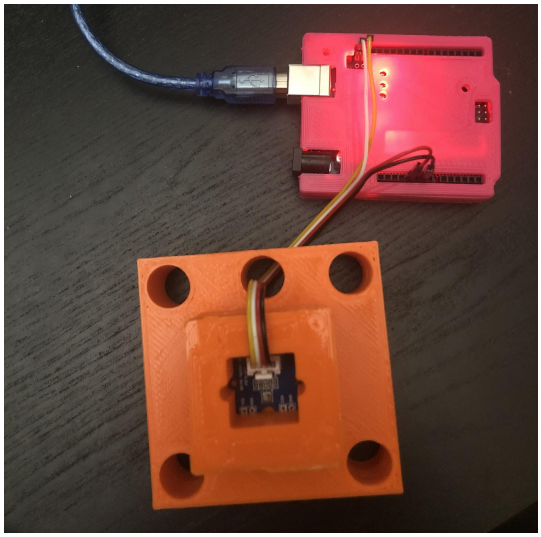

Figure 1. Picture of physical prototype III (first attempt)

In our second attempt, we fixed the issue of the hole size for the screws and we made the bottom casing less thick than our first attempt. The issue with this prototype is that when we made this prototype we accidentally made the peg holes too big, so the top casing does not stay in the peg

holes when flipped upside down. Also, when printing the casing the hot plate of the printer was not hot enough making our second attempt a little warped.

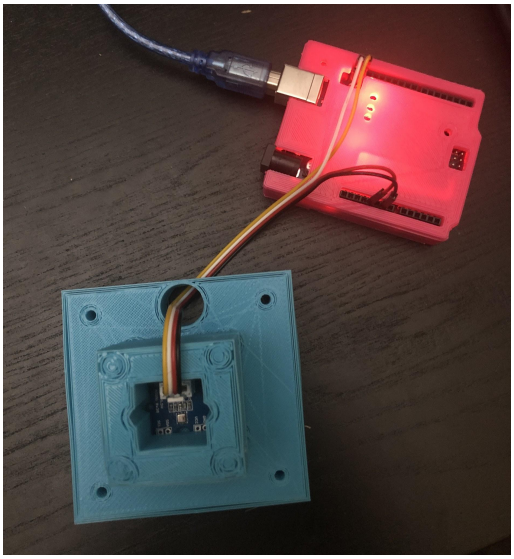

Figure 2. Picture of physical prototype III (second attempt)

As shown in figures 1 and 2, we can see the arduino casing. The case for the arduino has the perfect size however it is not very compact resistant nor water resistant (more information in the test plan section). All in all, we still have some things to work on, such as waterproofing the casing for the sensor and the arduino, fixing the fitting of the top and bottom cases for the sensor and improving the compact resistance of the arduino case. When making our final prototype before design day we will also make sure the holes will fit M3 screws since those are the screws that JAMZ will be using.

Current Functioning and Tested Code

The code has been tested, and it has passed all the required obstacles. The code has changed and gone through many different phases. During the first prototype, the values being printed were not realistic. The temperature was too high and the humidity was too low. After some debugging, the team realized the climate sensor outputs the humidity value first and then the temperature. When the values were being passed onto the created variables, the numbers were swapped. The first prototype gave the team a good insight on how the climate sensor needed to be programmed. The second prototype gave the team the insight on how to properly incorporate a warning system if any undesired value was being collected. The final prototype successfully prints accurate temperature and humidity values with time stamps and a warning system. The current functioning and tested code is given below.

```
/*
 * Make sure to include the AHT Library
```

```
 */
#include <Adafruit_AHTX0.h>

/*
 * Initialize the AHT library with the variables used for the Time Stamp
 */
Adafruit_AHTX0 AHT;
float seconds = 0.0;
int minutes = 0;
int hours = 0;


/*
 * Start with a Baud rate of 115200
 */
void setup()
{
  Serial.begin(115200);
  AHT.begin();
  /*
   * Create a loop iterating multiple times to have a line for the start of the table
   */
  int i = 0;
  for (i=0; i<60; i++)
  {
    Serial.print("-");
  }

 /*
 * Create the column headers for the table
 */
char firstColumn[] = "Time Stamp";
char secondColumn[] = "Temperature";
char thirdColumn[] = "Humidity";
char fourthColumn[] = "Warning";
Serial.print("\n|");
Serial.print(firstColumn);
Serial.print("\t|");
Serial.print(secondColumn);
Serial.print("\t|");
```

```
Serial.print(thirdColumn);
Serial.print("\t|");
Serial.println(fourthColumn);

}

void loop()
{

  float frequency = 0.5;
  float timeStamp;
  /*
    * Have time stamps showcase minutes and hours.
    */

  Serial.print(hours);
  Serial.print(":");
  Serial.print(minutes);
  Serial.print(":");


  /*
   * Have the variable timeStamp increase by 0.5 for each iteration
   */

   timeStamp = seconds + frequency;
   seconds = timeStamp;
   Serial.print(timeStamp);
   Serial.print("s");

  /*
   * Once the seconds reaches 60, add one minute to the minute variable. Repeat for the hours
variable.
   */
  if (seconds == 60.0)
  {
    seconds = 0.0;
    minutes = ++minutes;
  }
```

```
  if (minutes == 60)
  {
    minutes = 0;
    hours = ++hours;
  }

  /*
   * This comment is taken from learn.adafruit.com:
   * sensors_event_t - This type is used to encapsulate a specific sensor reading, called an 'event',
   * and contains data from the sensor from a specific moment in time.
   * Create a temperature and humidity value
   */
  sensors_event_t humValue, tempValue;
  /*
   * getEvent: Gets the temperature as a standard sensor event
   */
  AHT.getEvent(&humValue, &tempValue);
  /*
   * Print the temperature and humidity values in one row
   * Wait 500 milliseconds or the desired amount and print the next set of values in the next row
   */
  Serial.print("\t  ");
  Serial.print(tempValue.temperature);
  Serial.print("°C\t  ");
  Serial.print(humValue.relative_humidity);
  Serial.print("%\t  ");
  /*
   * Set up conditions to showcase a warning statement if temperature and or humidity values are
out of range
   */
  if (tempValue.temperature > 27.5 && humValue.relative_humidity > 60)
  {
    Serial.println("WARNING, TEMPERATURE AND HUMIDITY TOO HIGH");
  }
  else if (tempValue.temperature < 22.5 && humValue.relative_humidity < 45)
  {
    Serial.println("WARNING, TEMPERATURE AND HUMIDITY TOO LOW");
  }
  else if (tempValue.temperature > 27.5 && humValue.relative_humidity < 45)
  {
```

```
   Serial.println("WARNING, TEMPERATURE IS TOO HIGH AND HUMIDITY IS TOO
LOW");
  }
  else if (tempValue.temperature < 22.5 && humValue.relative_humidity > 60)
  {
   Serial.println("WARNING, TEMPERATURE IS TOO LOW AND HUMIDITY IS TOO
HIGH");
  }
  else if (tempValue.temperature > 27.5)
  {
   Serial.println("WARNING, TEMPERATURE IS TOO HIGH");
  }
  else if (tempValue.temperature < 22.5)
  {
   Serial.println("WARNING, TEMPERATURE IS TOO LOW");
  }
  else if (humValue.relative_humidity > 60)
  {
   Serial.println("WARNING, HUMIDITY IS TOO HIGH");
  }
  else if (humValue.relative_humidity < 45)
  {
   Serial.println("WARNING, HUMIDITY IS TOO LOW");
  }
  else
  {
   Serial.println();
  }

  delay(500);
}
```

CAD Prototype

During the past two weeks we had the opportunity to make lots of changes to our CAD. We also had the opportunity to print these designs. Through much trial and error we were finally able to design and print an effective sensor case. The biggest concern with this third prototype is that the core of the case happens to be way too large and dense. By the next deliverable we plan to print a new smaller case and test it. Not only do we want to reduce materials in general, we also want to ensure our total system meets JAMZ's weight/mass needs.

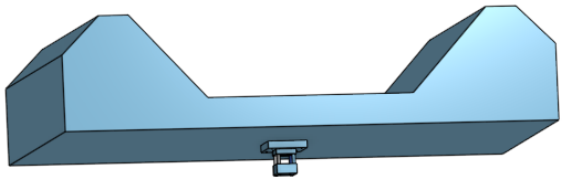

Figure 3. Sensor Casing on Drone Lid

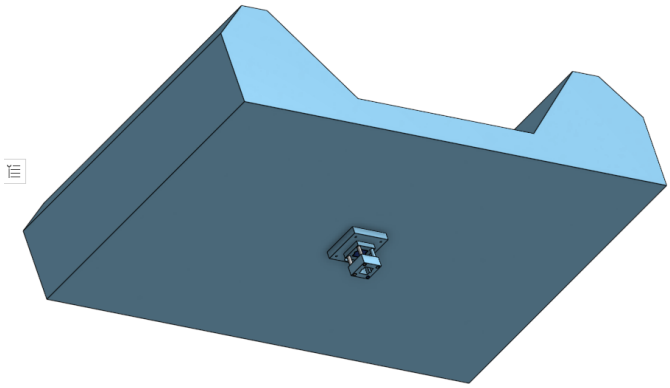

Figure 4. Sensor Casing on Drone Lid

Figures 3 and 4 demonstrate the placement of the Sensor and Casing on the interior of the drone lid. The Sensor and casing will be placed and screwed to the center of the lid and inside the box, but the sensor will not touch the box directly in order to provide optimal results. The CAD design of the sensor casing on the drone lid was improved by making the lid a replica of the one JAMZ has designed, making it easier for the client to identify the placement of our sensor on their drone.

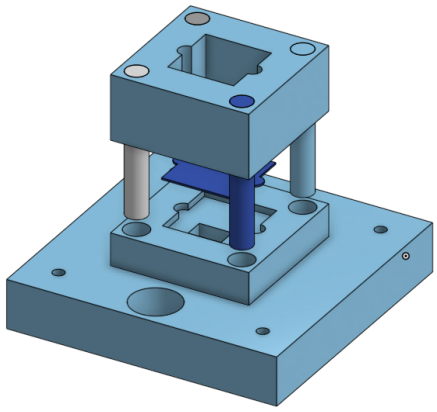

Figure 5. Sensor Casing

Figure 4 demonstrates the design of our sensor casing created to protect the climate sensor from damages. The casing was improved from the last sensor casing design to have a thicker handle so it is easier for the user to remove the lid of the climate sensor when needed. We noticed in the previous prototype that the pegs were too thin which made them susceptible to breakage. In this prototype, we made the pegs thicker so they are significantly less prone to easy breakage. Furthermore, we included a larger hole for the sensor wiring to go through for more efficient connection. To improve this prototype, we are looking to use M3 screws since those are the screws JAMZ is using.

In the last deliverable we decided to print an arduino case to include in our physical prototype. During this past week we took the time to perform several drop tests to check the effectiveness of the arduino casing. Unfortunately, at relatively low heights the top of the casing was coming off. From here we decided we would have to alter the way the bottom and the top of the casing will attach.

Test Plan III

Table 1: Climate Sensor Test Plan

Test ID	Test Objective (Why)	Description of Prototype used and of Basic Test Method (What)	Description of Results to be Recorded and how these results will be used (How)	Estimated Test duration and planned start date (When)	Analysis

1 (Code)	To ensure proper and accurate function of code with sensor.	Numerical and physical prototype. Test speed of data and response time reliability	Plug in sensor and change the temperature and humidity of the surrounding environment $0 (use materials we already have)	New code already tested	Prototype I and II gave the team all the knowledge needed to fully understand how to program the AHT20. Accurate temperature and humidity values were being printed with a warning system in place. The next big step was adding a timestamp. After some testing, a timestamp ranging from seconds to hours was implemented successfully.
2 (Sensor functionality within case)	To ensure that the sensor will work properly while within the protective casing.	Numerical and physical prototype. Test the abilities and accuracy of the sensor while it is in the protective casing.	Plug in sensor while it is in the protective casing and change the temperature and humidity of the surrounding environment	Test this while testing the newest version of the code (ASAP).	The sensor works just as well with the case on as it is without. In the same environment tested just minutes apart, the same temperature and humidity values were obtained.
3 (Weather proofing)	To ensure no weather elements can enter the box at any connectio n points.	Physical prototype. Test the ability to withstand different weather, such as rain, from entering the	Spray water from a bottle onto the prototype to see if any water enters. For the prototype, use rubber blades to	Test as soon as all materials are purchased and available for use.	No water is able to penetrate into the box, however, the waterproofing for the actual sensor and the Arduino will need more work. Currently, all solutions we have work theoretically,

		delivery box or the drone.	cover any openings and surround any components within those openings (other materials might need to be used for sooner testing).		but the materials will need to be purchased in order to understand what the best approach to the situation will be.
4 (Sensor protection)	To ensure durability and protection to the sensor.	Physical prototype. Test the strength of the casing. Test how casing reacts under different environments with and without the sensor inside.	Perform a drop test from varying heights both with and without the sensor in the protective casing. Perform this test multiple times.	Already tested.	The case was able to withstand the drop tests with the exception of the top detaching on impact. Moving forward, a redesign of the pegs holding in place will be implemented in order to fix this problem.
5 (Sensor Attachment)	To ensure the sensor does not disconnect during delivery.	Physical prototype. Test the strength and fitting of the attachment style under different environments.	Attach sensor casing to the delivery like box and test its strength by shaking to observe how it holds on to the lid.	Proper testing will be held as soon as the final case is printed. Prototype testing will be held as soon as the proper screws have been purchased and are available for use.	Using the proper screws in order to attach the sensor casing to the lid of the delivery box will ensure no separation between them will occur. The screws hold everything together very nicely.

6 (Wire connection)	To ensure nothing will come loose or disconnect during a delivery.	Physical prototype. Test the physical connection of the wires with the updated attachment system. Test for any kind of flight path and disruptions that may occur.	Plug wires into all other necessary components using new additions and move/shake the prototype as if to simulate a rough flight.	Test as soon as sensor attachment has been tested.	With the addition of the casing for both the sensor and the Arduino, The wire connection is proving to be quite strong. To guarantee the connection of the wires, soldering them in place would be the next step.
7 (Arduino protection)	To ensure durability and protection to the Arduino Uno.	Physical prototype. Test the strength of the casing. Test how casing reacts under different environmen ts with and without the Arduino inside.	Perform a drop test from varying heights both with and without the Arduino Uno in the protective casing. Perform this test multiple times.	Already tested.	The case was able to withstand the drop tests with the exception of the lid coming off on impact. Moving forward, a secure attachment of the base and top of the case will be implemented in the design.

Client Feedback

Looking back on the past input we have received from the clients, we have been able to make improvements to each of our prototypes. For this past prototype we were able to reach out to JAMZ directly and gain their input on our last design. We had many questions to ask the client in order to provide them with the best design possible. Once these questions were answered, we were then able to improve on all aspects of our system. There is still more that can be improved upon, but this design satisfies all and even surpasses some of the expectations set out by JAMZ.

Conclusion

In conclusion, our third prototype did not end up being our final prototype. Although many improvements were done on our casings, our tests revealed that we still had adjustments to make. The first one being, reducing the size of our sensor casing and the second one being, altering our arduino casing. Code wise our team has many positive additions, including adding time stamps. For the next deliverable and design day with minor adjustments we are confident that our testing will be complete and our prototype will be finalized.

Wrike Planning for Next Deliverable

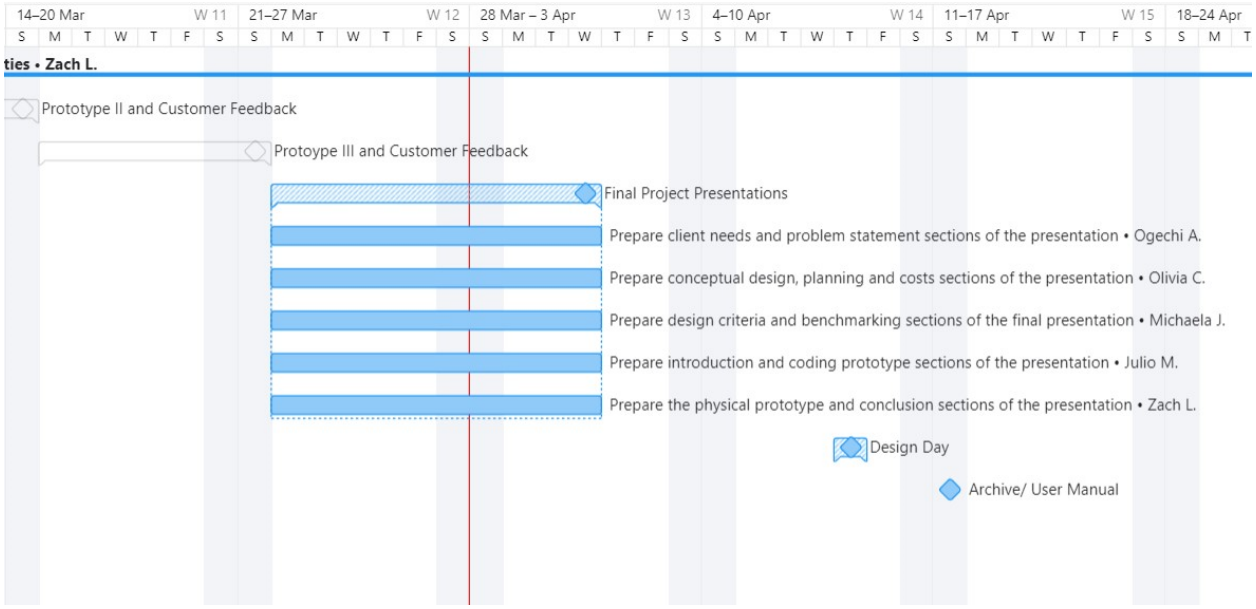

Figure 6. Screenshot of gantt chart for the upcoming deliverable

For the next deliverable, our team needs to prepare a final presentation of our design process for the climate sensor add-on. For the preparation of our presentation, there will be no dependencies because everyone is preparing their own section, which is independent of the other sections. To start the presentation, Julio will prepare and present the introduction and the coding sections of our project. Julio was chosen to work on these sections due to his high energy when presenting (very important for the introduction) and his strong knowledge of the project and code. Following Julio, Ogechi will prepare and present the client needs that we have gathered throughout the emphasizing stage of our project as well as the problem statement that our team has defined. Ogechi was chosen for this task because they have a very strong understanding of the client's needs and strong verbal communication skills which is crucial for the explanation of the problem statement (viewers of the presentation must fully understand what we set out to accomplish with our add-on). Next, Michaela will prepare and present the design criteria and

benchmarking sections of the presentation. Michaela was chosen to work on these sections because she was responsible for benchmarking and came up with our design criteria in earlier deliverables, making her the ideal candidate to prepare these sections of the presentation. Moving on, Olivia was chosen to present and prepare the conceptual design, planning and cost sections of our presentation. Olivia was chosen to work on these sections because she has a strong understanding of all the conceptual designs that our team has come up with (including the ones that she came up with). Also, she was responsible for the cost analysis in an earlier deliverable, so she has a strong knowledge of the costs of our add-on. Olivia was not responsible for planning the team's tasks throughout the semester, however every team member understands and knows the planning system very well, so she is able to present that section. Lastly, Zach was chosen to prepare and present the physical prototyping section of the presentation. Zach was chosen for this task due to his heavy involvement of the fabrication of all the physical components. In conclusion, the tasks assigned for this deliverable were carefully planned out so that everyone was able to contribute equally as well as utilize their strengths, in order to ensure success.