

GNG 1103: Deliverable H

Group D12: Hot Solutions

Introduction

The following document contains the test plan and results for prototype 3 of the climate sensor addition to JAMZ drone delivery service. For the third prototype physical models were created of both the sensor case and the Arduino case. The code that was created during the second stage of prototyping was also tested. The sensor was also tested to determine how the AHT20 sensor responded to changes in the external environment. Also included will be future test plans to ensure that all the subsystems connect together and function as intended.

Past Prototypes: What was Accomplished and What was Learned.

The first prototype consisted of focused analytical models of both the sensor cases and the Arduino protective case. These designs were based off of dimensions obtained online to determine the size of the Arduino nano and the dimensions of the sensor. Tests for these designs included determining weight and volume of the cases. Both of these designs have since been modified due to design changes and errors in the dimensions. Also created during the first prototype was the basis of the code that was going to be used. The serial communication aspect of the code was tested. Serial communication is the method of communication between the Arduino nano used in the climate sensor design and the Raspberry Pi that is used in the drone. The code was tested with no sensors attached, so the message that was received was: sensor not detected. As no sensors were attached this was the expected response, showing that the code was functional. The exponential filter that will be used to smooth out spikes in the data. The code was tested against analytically produced data. Both the analytically produced data and the data obtained using the code matched. This proved the functionality of the code.

The second prototype consisted of a focused model of the most critical subsystems; the communication of data between the sensors and the Arduino, and between the Arduino and the raspberry pi. This prototype consisted of two parts. The first tested a spam prevention function so that data is not being transmitted unnecessarily. The spam prevention function uses the millis function to prevent new warnings from being transmitted to the raspberry pi until after a certain time has passed. This allows for the drone operator to correct the problem without being given a new warning every tenth of a second. The second part consisted of testing the communication between the Arduino and the Pi. An LED was used to show if the message was received by the pi. The code was run, and the LED turned on, indicating that the message had been received.

Following prototype II all the major components of the code had been created and tested. At this stage a tentative final code was created.

Prototype III

For prototype III, several different components of the design were tested. Both physical and analytical prototypes were used during this stage of testing. A physical model was developed for both the Arduino case and the sensor case. These were developed from the CAD models created during an earlier prototype. As all the necessary components had been gathered, the tentative final code was tested during this stage of prototyping. Tests were also run, using only one sensor, to observe how the sensor reacts to changes in the external environment.

Arduino Case

Two models of the Arduino case were created during the third stage of prototyping. These are shown in Figures 1 and 2. Both were printed using a 3D printer and made of PLA. The first case was made using dimensions found online. These dimensions turned out to be slightly small as the Arduino didn't fit inside

the case. This first printed case also had a hole in the front to allow for wires to exit. The hole at the front was deemed unnecessary as there were already areas to allow for the wires to exit along the sides of the case. The design was modified, and a second case was printed.

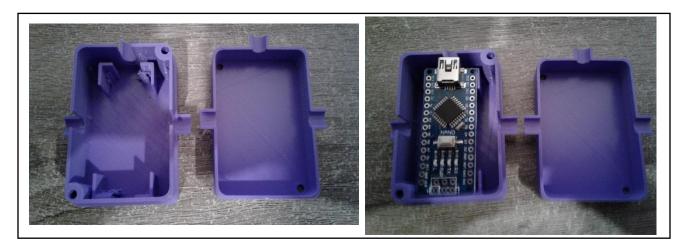


Figure 1: This is the first printed prototype of the Arduino case. As shown on the right the Arduino does not fit inside.



Figure 2: This is the second printed prototype of the Arduino case. The photo on the left shows that the Arduino now sits snugly inside the case. The photo on the right shows the side view of the case. The hole is for wires to exit the case.

These cases were tested to determine whether they were resistant to water and to low impact collisions. To simulate a low impact collision, the case was thrown a distance of 2.5m against a wall. Upon impact with the wall, the case would rebound and a second impact with the floor occurred. This test was repeated multiple times to see how the case design would fare after repeated collisions. Upon examining the case after the collisions, the case showed no indications that it had been involved in a collision. The surface of the case remained unbroken, and both halves of the case remained together during the collisions. The Arduino nano fits very snugly in the case. Even the collision did not displace the Arduino from where it sat within the case. This feature was requested by JAMZ because if the components are protected, they can be recovered even if the drone does crash.

Two tests were also done to test water resistance. The first test involved placing a small amount of water on the top face of the case. The purpose of this was to observe if water can get inside the case. Water getting inside the case could damage the Arduino. Damage to the Arduino would affect the entire function of the climate sensor system. The water sat on the top face of the case for an hour. When the water was removed the inside of the case was still completely dry. No water had passed through the top face. The second test involved sliding droplets of water along the wires that were exiting the case. Water could enter the case through these holes and damage the Arduino. The test showed that water will adhere to the wires and will move along them. A minimal amount of water does reach the case, a very small amount gets inside the case. To further waterproof the case a waterproof spray could be applied. A rubber sealant could also be applied around the wires to ensure that no water enters the case through the holes where the wires exit.

AHT20 Case





Figure 3: This shows the two halves of the sensor case.

shown on the left the sensor sits within the bounds in the bottom of the case. On the right is the incorrectly printed top half.

A physical prototype of the sensor case was created based off of the dimensions of the sensor chosen. This involved making changes to the original CAD model, as a different sensor was being used than had been chosen during stage one of prototyping. However, problems arose during printing as the top part of the case printed as a solid block rather than a shell. This means that no tests could be run using the case. As shown the sensor does fit nicely in the bottom half of the case.

Final Code Tests

Testing of the final code took place in two stages. First, a simplified model of the code was tested using only one sensor. The purpose of this was to establish how the sensor responds to temperature and humidity changes. The second stage of testing involved testing the full code using both sensors. This is used to establish that the code works as intended and accurate data is communicated to the Arduino.

Testing with One Sensor

A simplified version of the final code was tested using only one sensor. The purpose of these tests was to determine that the sensor functioned as it was supposed to, and to observe how the sensor responded to changes in the operating conditions. The sensor was tested above a mug of steaming water and outdoors.

The different locations allowed for observations to be made about the sensor responds to different environments. They also allowed for the accuracy of the sensor to be determined.

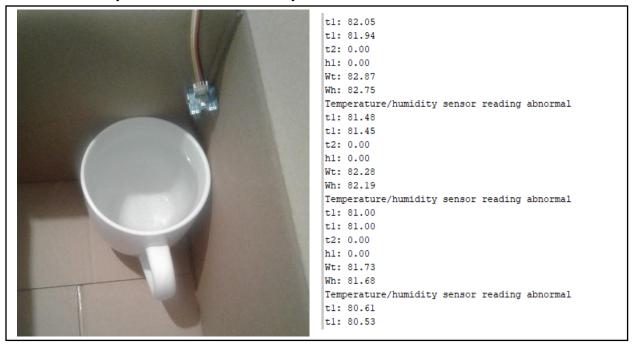


Figure 4: The photo on the left shows how the sensor was placed for the test. The photo on the right shows some of the data received from this test.

For this test the sensor was positioned above a mug of the water. The water in the mug had just been boiled so its internal temperature was roughly 100°C. When the sensor was placed above a mug of steaming water the temperature reading on the sensor increased quickly. As shown in figure 4 the temperature detected by the sensor was above 80°C. The temperature reding from the sensor rose higher than this over the course of the test. For the actual drone delivery the food or drink being delivered will be in a sealed container so the temperature change will not happen as quickly or change as drastically as it did during this test.

A test was also run to see what the serial monitor showed when the sensor was moved outside. The outside temperature when the test was run was 13°C. The values recorded by the sensor, shown below in Figure 5 do not show this value. Some reasons for this discrepancy include; warm, humid air from the previous test was still influencing the temperature reading or the outside location. The outdoor location was in a corner, close to the walls of the house. Heat from inside the house could have been escaping and warmed the air slightly. This would have given incorrect temperature values. Tests will be rerun outdoors with both sensors in the future.

```
ing abnormal
t1: 31.81
 t1: 31.63
 t2: 0.00
 hl: 0.00
 Wt: 31.63
 Wh: 31.52
 Temperature/humidity sensor reading abnormal
 tl: 31.86
 t1: 31.80
 t2: 0.00
 hl: 0.00
 Wt: 31.74
 Wh: 31.63
 Temperature/humidity sensor reading abnormal
 t1: 32.00
 t1: 31.95
 t2: 0.00
 hl: 0.00
 Wt: 31.84
 Wh: 31.76
 Temperature/humidity sensor reading abnormal
 t1: 32.08
 t1: 31.98
 t2: 0.00
 h1: 0.00
 Wt: 31.95
 Wh: 31.87
 Temperature/humidity sensor reading abnormal
t1: 32.24
```

Figure 5: This is the data from when the sensor was used outdoors.

```
ens Error: Sensor 1 not detected
T1: 0.00
H1: 0.00
T1: 0.00
H1: 0.00
T1: 0.00
H1: 0.00
T1: 0.00
H1: 0.00
Sensor disconnected
Sensor disconnected
Sensor disconnected
Sensor disconnected
T1: 0.00
H1: 0.00
Sensor disconnected
```

Figure 6: Illustrates that an alert is sent when the sensor is disconnected.

As a potential failure prevention, the code was also tested to ensure that an alert was triggered if a sensor was disconnected. With two sensors in the design, one sensor being disconnected would not stop data from being collected. One sensor being disconnected could affect the accuracy of the data that is gathered. The test showed that an alert was triggered when a sensor was disconnected from the Arduino. For the test the sensor was disconnected from the Arduino after data collected had begun. As shown in the above figure an alert was triggered when the sensor was disconnected.

Testing with Both Sensors

After both the sensors had been delivered a problem was identified. Both sensors used the same address on the Arduino and could not be run at the same time. This problem was solved through the addition of a multiplexer to the circuit. A multiplexer would take the inputs from both sensors and convert them into one output, which would then be communicated to the Arduino. However, upon implementing the multiplexer into the circuit additional problems arose. When the sensors were not directly connected to the Arduino, the sensors were not detected as an input.

Some proposed reasons for this problem were that one of the sensors or the multiplexer didn't work, the circuit was connected incorrectly, and that the code itself was faulty. Tests were created to determine if any of these problems was the issue. The sensors were tested by connecting them directly to the Arduino and running a code that reads the values from one sensor. The serial monitor was then used to read the incoming data. When connected directly to the Arduino both sensors communicated the temperatures and humidity. This test determined that the sensors were not the problem. The multiplexer was also tested using a multiplexer test code.

A third guess was that the circuit was connected incorrectly. This was later proved to be false as all the connections were correct with the circuit diagram that had been created. Both the SDA and SCL pins were tried on the Arduino, as well as using A4 for SDA and A5 for SCL. When the sensors were connected directly to the Arduino both connecting to the SDA/SCL and the A4/A5 pins yielded results. When the multiplexer is connected to the Arduino the A4/A5 pins give a response to the multiplexer test code. However, the sensors were still not detected when run through the multiplexer.

The final guess was that the code itself was faulty. To determine whether this was the case, several tests were run with example codes to see if this was the case. The I2C scanner example code was used. This code tests to see whether I2C devices are found. When the scanner code was run the output from the code showed that no I2C devices were found when the sensors were connected to the multiplexer. When the sensors were connected to the Arduino, the I2C scanner found the sensor. This indicates that the problem is not with the code, or at least not only with our code.

Upon further thought, a new reason for these issues has been proposed. The proposed solution is that a pull up resistor is required in the circuit. The resistors would connect to each of the connected SDA/SCL pins. This solution will be tested in the coming week.

Circuit Diagram

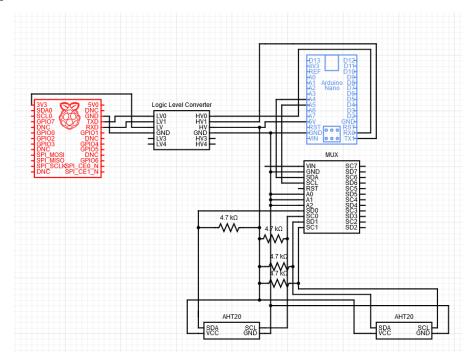


Figure 7: This is the circuit diagram that shows all the components in the climate sensor circuit.

The circuit diagram utilizes a raspberry pi, but during testing, the raspberry pi was not used. The circuit diagram has been updated to show the pull up resistors. If the 4.7K resistors are not available, a resistor near that value will be used, but it will be no less than 2K due to the 3mA max sink rate on the I2C pin.

Switch case test

```
void serialEvent() {
 while (Serial.available()) {
   message = Serial.read();
   switch (message) {
   case 'c':
     boxOpen=false:
     break;
   case 'o':
     boxOpen=true;
     break;
    case 'H':
     idealTU = 30;
     idealTL = 20;
     idealHU = 90:
     idealHL = 0;
     Serial.println(message);
     Serial.println((String)"Ideal temp: "+idealTL+" to "+idealTU);
    case 'C':
     idealTU = 15;
     idealTL = 0;
     idealHU = 90;
     idealHL = 0;
     Serial.println(message);
     Serial.println((String)"Ideal temp: "+idealTL+" to "+idealTU);
     break;
    case 'S':
     idealTU = 35:
     idealTL = 25;
     idealHU = 90;
     idealHL = 0:
     Serial.println(message);
     Serial.println((String)"Ideal temp: "+idealTL+" to "+idealTU);
     Serial.println(message);
      Serial.println("Error: Invalid input");
```

Figure 8: Script for the switch case test.

```
H
Ideal temp: 20.00 to 30.00
C
Ideal temp: 0.00 to 15.00
S
Ideal temp: 25.00 to 35.00
p
Error: Invalid input
```

Figure 9: Illustrates the different temperature values used with different types of food.

The switch case an additional feature that was created to allow for the operator to set different temperatures depending on the food type. When it receives a char from its serial communication, it goes through the switch statement to check if what it had received was to change its set temperature range. However, it has been mostly commented out for now as JAMZ mentioned they are not sure if they will be delivering cold foods. If they wish to do so in the future, the option is available.

Future Test Plans

Over the next two weeks, a new case will be printed for the sensors. The case will then be tested to ensure that it is resistant to low impact collisions. Tests will also be run to ensure that the case protects the sensor

without inhibiting its function. Further debugging of the code will continue until a prototype is created where both sensors function while being connected to a multiplexer and not directly plugged into the Arduino. The design with both sensors will then be tested. A physical box will be created to simulate actual running conditions. The sensors will be positioned inside the box where they would be positioned if they were actually on the drone. A mug of hot water will be placed inside the box. The box will then be closed off and the temperature and humidity will be monitored. This test will be repeated several times to ensure that the climate sensor system responds the same way each time. A comprehensive analytical model is also being crafted to show how each subsystem will be positioned on the drone and to test total size and weight.

Conclusion

The purpose of this prototype was to create physical components for several of the subsystems and to test the response of the sensor to different stimuli. All these features are necessary. The Arduino protection case is needed to protect the Arduino from weather damage and from low impact collisions. The sensor protection case is needed to protect the sensor from moisture inside the box. The most critical component of the climate sensor is the communication between the sensor and the Arduino. This is why it was imperative that the code be tested. The results from these tests helped us to understand how the sensor responded to changing environments and also helped us to understand the accuracy of the sensor. Future tests still need to be run to ensure that the design works as intended.