



uOttawa

GNG 1103: Deliverable F

Group D12

## Introduction

The first prototype of the JAMZ Climate Sensor project is outlined in this deliverable. Initial prototyping includes the prototyping and dimensioning of the initial set of subsystems. The prototype test plan, which is the list of testing objectives for the given subsystems are shown, along with the description of the prototype and the description of the results.

## Prototype Test Plan

Test ID	Test Objective	Description of Prototype Used and Basic Test Method	Description of Results to be Recorded and how these Results will be Used	Estimated Test Duration and Planned Start Date
1	Check the functionality for the code for serial communication between Arduino and raspberry pi	-Focused -Physical Connect the Arduino and raspberry pi through USB and execute the code	Observe if output from Arduino is received, and correct on raspberry pi to confirm that serial communication code is functional.	1 day Mar 4-5
2	Create a simple, functional algorithm for data processing and check for functionality.	-Focused -Analytical Feed the Arduino code the unfiltered data and have it filter it and record the output.	The filtered and non-filtered output data from the Arduino is plotted on a graph and compared against the analytically created plot. The filtered data and analytically produced data match to see if the algorithm is working as intended	1 day Mar 4-5
3	Test new dimensions and ratios of the Arduino case after the following changes were made to the final design drawing in Deliverable E: i. Alternative design that lowers production costs (Focus on decreasing the number of screws needed to	Prototype I: CAD model of Arduino Protective Case - Analytical Prototype - Focused - Fidelity: Low  CAD model acts as a proof-of-concept.  Program produces a 3D visual of the system that tests:	Results will appear as CAD is rendered.  Model is easy to modify after program's immediate feedback. Changes are made "as you go", no material resources are used up.	Start Date: March 3 Duration: 2 days

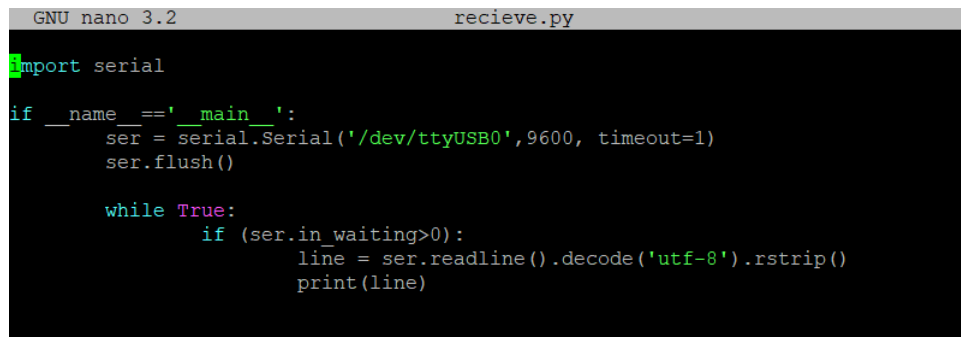
	<p>mount case, create a mounting system for the Arduino that can be 3D printed)</p> <p>ii. Alternative design so wire seals are directly printed onto the Arduino Case. Design must be water resistant, and functional (wires can be moved when needed).</p> <p>iii. The aesthetics of the case (sleek, reduce distinction between top and bottom parts).</p>	<ul style="list-style-type: none"> <li>- Contradictions in measurements/restrictions</li> <li>- Alignment of parts during assembly</li> </ul>	<p>The end of Prototype I testing should produce a system where the appropriate design changes were met, no contradictions in measurements are highlighted, and parts appear aligned during assembly.</p> <p>The completed CAD model will be 3D printed for future testing.</p>	
4	Test the general dimensions and alignment of parts for the sensor case.	<p>Prototype 1: CAD model of sensor case</p> <p>-Analytical prototype</p> <p>CAD model acts as a proof-of-concept</p> <p>Program produces a 3D visual of the system that tests:</p> <ul style="list-style-type: none"> <li>- Contradictions in measurements/restrictions</li> <li>- Alignment of parts during assembly</li> </ul>	<p>Results are rendered in Onshape as a CAD model.</p> <p>The CAD model allows for easy adjustments to the dimensions of the sensor case without requiring material recourses.</p> <p>The results of the CAD model will be 3D printed for future prototypes and testing.</p>	<p>Start Date: March 5</p> <p>Duration: 2 days</p>

## Prototype 1

### Serial communication Prototype

```
void setup() {  
  //Begins the serial  
  Serial.begin(9600);  
  while(!Serial){ //Waits until serial begins  
    delay(20);  
  }  
  //initializes the sensors  
  sensor1State = sensor1.begin(0x44);  
  delay(10);  
  sensor2State = sensor2.begin(0x45);  
  //Checks if sensors are detected by the arduino  
  if(sensor1State && sensor2State)  
  {  
    Serial.println("Sensors detected: Setup successful");  
  }  
  if(!sensor1State){  
    Serial.println("ERROR: Sensor 1 not detected");  
  }  
  if(!sensor2State)  
  {  
    Serial.println("ERROR: Sensor 2 not detected");  
  }  
}
```

Fig 1.0. Serial prototype code



The image shows a terminal window with a dark background. The title bar at the top reads "GNU nano 3.2" on the left and "recieve.py" on the right. The code is written in a light green font. It starts with "import serial". Then, an "if \_\_name\_\_ == '\_\_main\_\_':" block contains the main logic. Inside, "ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)" is followed by "ser.flush()". A "while True:" loop contains an "if (ser.in\_waiting > 0):" condition. Inside this condition, "line = ser.readline().decode('utf-8').rstrip()" is followed by "print(line)".

```
GNU nano 3.2                                recieve.py  
import serial  
  
if __name__ == '__main__':  
    ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)  
    ser.flush()  
  
    while True:  
        if (ser.in_waiting > 0):  
            line = ser.readline().decode('utf-8').rstrip()  
            print(line)
```

Fig 1.1. Raspberry pi code<sup>1</sup>

```

pi@webber:~/Documents/serial $ python recieve.py
ERROR: Sensor 1 not detected
ERROR: Sensor 2 not detected

```

Fig 1.2. Serial reading from raspberry pi

The serial output was as expected since neither sensors were connected to the Arduino and the message was received correctly on the raspberry pi . This test will need to be conducted with the sensors connected and the serial communication through GPIO instead of USB in the future when the parts are available. This will ensure that it will actually work once it is hooked up to the drone. For now, this confirms that the serial communication code is functional.

## Data processing Prototype

```

/*Taking the averaged temperature of the two sensors, humidity, and previous weighted values it
applies the exponential filter to return the filtered inputs */
float filterInput(float t,float h,float *wt,float *wh)

{
    *wt = w*t + (1-w)*(*wt);
    *wh = w*h1 + (1-w)*(*wh);
}

```

Figure 2.0. Arduino function for data processing

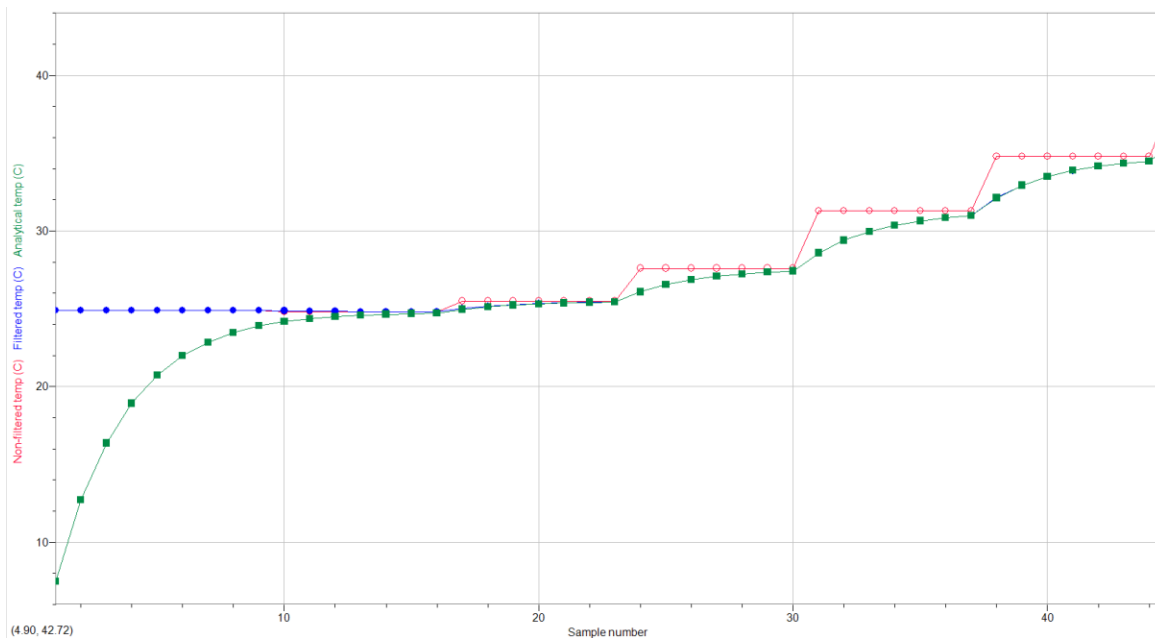


Figure 2.1. Plot of temperature (filtered, non-filtered, and analytical)

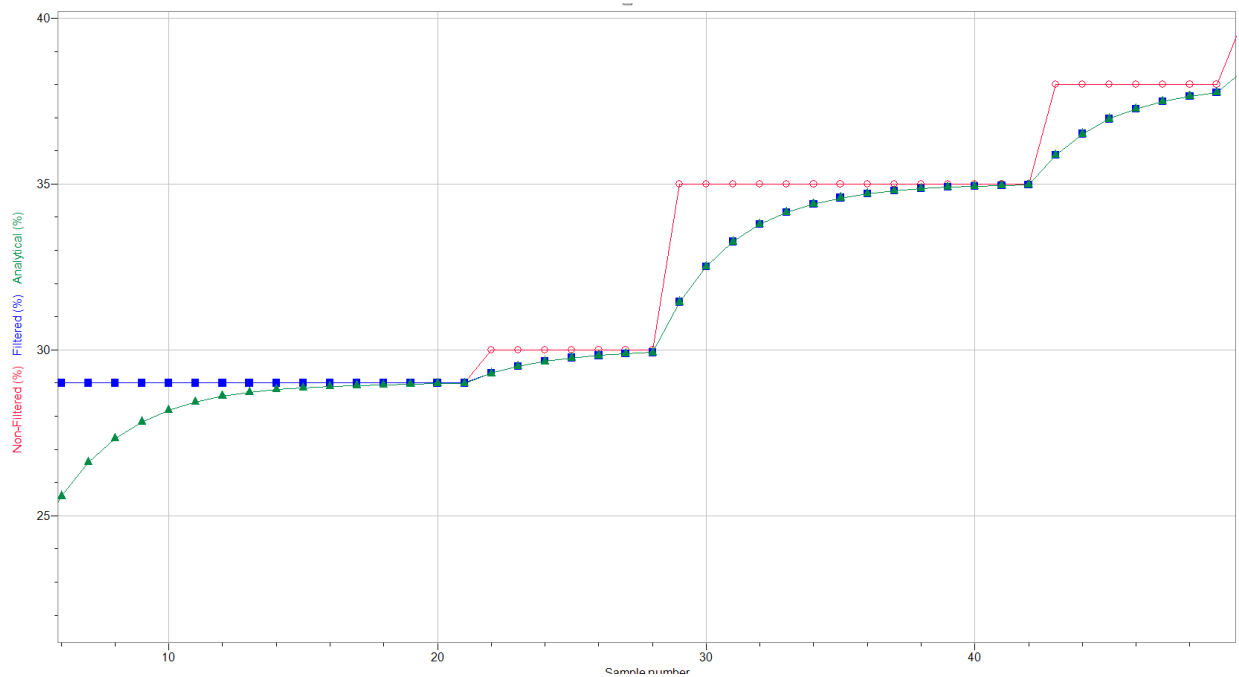


Figure 2.2. Plot of humidity (filtered, non-filtered, and analytical)

The Arduino code matches the analytically produced data, confirming that the code works as intended. The analytical has a dip at the front that is not seen by the Arduino's filtered data, but that is due to the equation itself for the exponential filter and the pre-sampling done by the Arduino code. As seen in the equation  $y(x) = w \cdot t + (1-w) \cdot y(x-1)$ , it takes couple of samples to weight out and properly weigh the reading. The Arduino pre-samples the data to prevent the first initial weighted value from starting off low which would send an alert message to the operator. This pre-sampling is done by having the code loop the incoming data  $n$  number of times (in this case 40) initially, to stabilize the value. Further optimizations and consideration needs to be made for data processing such as, when the operator should be warned, sampling rate, and how often the Arduino should send warnings, etc.

### Arduino Protective Case Prototype I

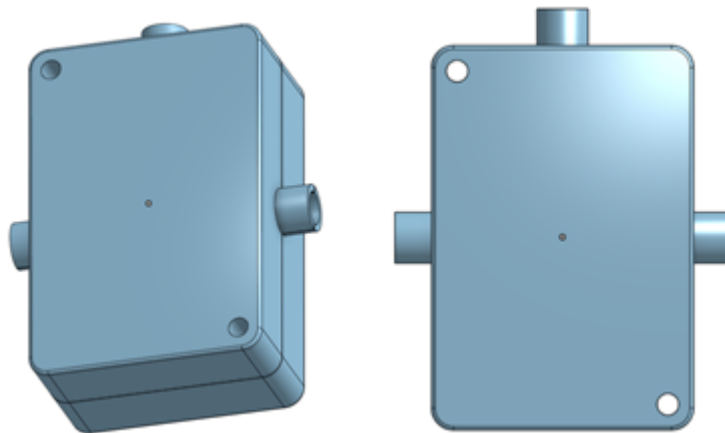


Figure 3.1 CAD assembly of Arduino protective case

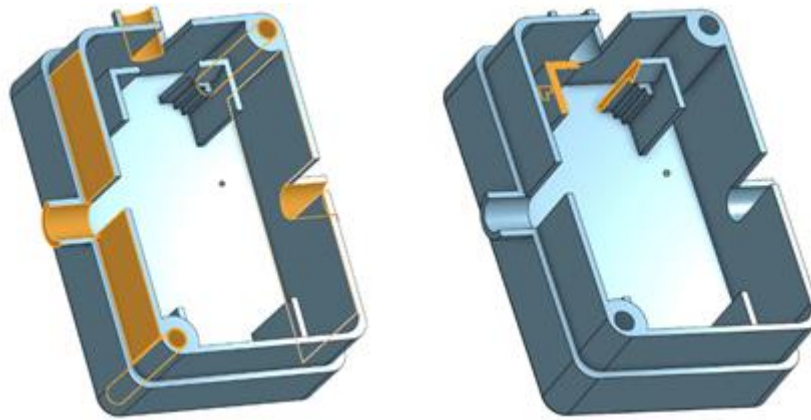


Figure 3.2 Bottom of Arduino protective case

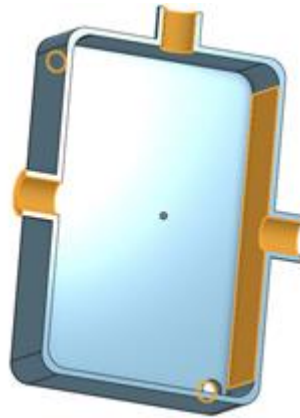


Figure 3.3 Top of Arduino protective case

Test 3 was conducted and all required changes to the final design in Deliverable E were completed successfully. The CAD model contains both aligned parts and no contradictions in measurements. The prototype was a proof-of-concept to reduce production costs of the Arduino protective case. As a result, the team was able to lower the BOM down to the project budget.

First, the number of mounting screws was decreased. Two M3 x30mm screws will go through the case and mount onto the drone instead of the initial four. They sit diagonally of each other to minimize movement, as well as not to interfere with wiring. When the top is installed the walls of the screw holes are 2mm thick. The Arduino mount can be seen in Figure 3.2. The mount is to be 3D printed as part of the case to reduce costs. The large mounting walls highlighted in the second perspective of Figure 3.2 run 15mm high, 1mm thick. They share the same dimensions as the Arduino Nano to help slide its corners in place. Small supports against the large walls keep the board elevated 10mm high. The feature prevents the Arduino pins from making contact with the floor underneath. Bevels extending 1mm out of the tall mounting walls are also highlighted in the second perspective of Figure 3.2. They are located on the same diagonal as the screws. The bevels are meant to stop movement perpendicular to the floor. They allow a 1.6mm space for the board to sit.

NinjaFlex was the initial material intended for the wire seals. Since Deliverable E, it was communicated to the team that the material tends to be unreliable during 3D printing. Changes in the design were made. The wire seals will now be composed of the same ABS material as the case and extend 5mm outward from the exterior walls. For functionality, each wire seal contains an open slot for adjusting wires. The bottom half of the wire seal sits just below the slot. The top of the case contains the other half of the seal that secures the wires in place when the protective case is screwed. Both top and bottom parts of the Arduino protective case were aligned, as shown in Figure 3.1; Therefore, a sleek design was achieved. A small 1mm indent was made around half the case, to allow the 1mm walls of the top part to slide into.

The assembled Arduino case is shown in Figure 3.1. The total dimensions of the case are 52.63x 35.78x 22mm. The exterior shell of the case is 2mm thick. Using the results from Test 3, the team will be able to move forward with prototyping. The CAD will be used to acquire a physical prototype. The 3D printed model will undergo tests on the reactivity of the material to certain forces. Testing will take an estimated duration of 2-3 days. The results will be used to alter dimensions of the material and reinforce any stress points. Physical testing will also be done on the water resistance of the wire seals. The results produced will finalize their diameters.

### Sensor Case Prototype 1

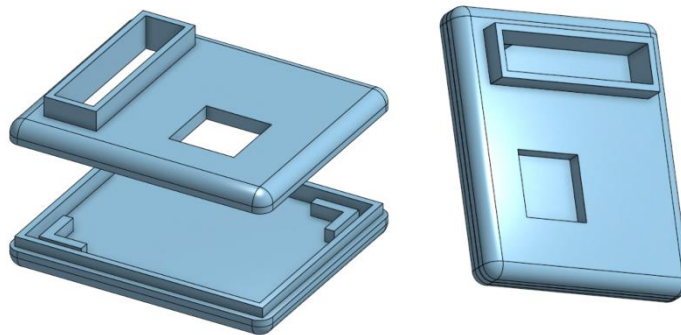


Figure 4.1 CAD assembly of sensor case

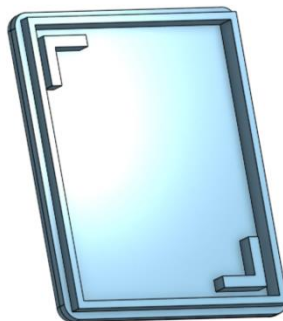


Figure 4.2 Bottom of sensor case



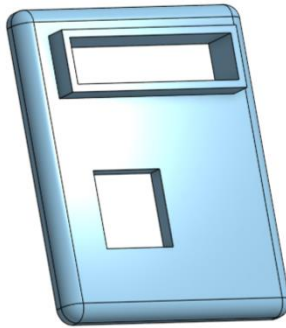


Figure 4.3 Top of sensor case

The sensor case serves the main purpose of supporting the sensor with a rigid outer shell for protection. Further testing is still needed though as the physical sensor has not yet been obtained. Although the sensors dimensions are known from online resources, the dimensions for the two holes on the top of the case are unknown and will have to be measured upon acquiring the sensor. The top hole is for the pins to connect to the sensor and the bottom will allow the sensor to be open to the atmosphere and gather data. The current prototype was made using online photos and resources and scaling them to the size of the climate sensor. Although the two holes in the top may not be accurate as of prototype I, they should be close enough to minimize error.

The other main test for the sensor case is determining where and how to mount it. With the absence of a lid on the drone delivery box, the sensor would have to be mounted to the underside of the drone. Testing will have to be done to visualize how effective the mounting is and how the distance from the food affects the sensors data readings.

The next phases of testing require the sensor to be purchased. Once the sensor is purchased then the sensor case dimensions can all be verified within a day and the case will be ready for physical prototyping. Once there is a physical prototype, testing can be completed to see how rigid and protective the case is along with determining where to mount the case. Testing how protective the case is should only take 1 day but determining where to mount the case will take more thinking and will require about 3-4 days.

## Conclusion

The first prototype built consists completely of focused analytical models of each subsystem. The initial designs were tested and size of the subsystems that will eventually have physical models were determined. A design that the Arduino nano and the two climate sensors would fit inside is to be built. An analytical model of the code that will be used for communication between the sensors and the drone was also designed. In the coming weeks more designs along with a physical prototype will be made to test to orientation of all the subcomponents relative to the drone. Code will also be tested to ensure that it works as intended, and adequate communication between the Arduino, the drone and the drone operator is established.

## Reference

1. <https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/>