GNG 1103 – Engineering Design

Project Deliverable G

Prototype II and Customer Feedback

Group#15

Lucas Siviero	St#300178151
Noah Aynalem	St#300166191
Rakshita Mathur	St#300215340
Riley de Gans	St#300170104
Timi Tella	St#300128051

Date: March 14, 2021
Presented to Justine Lucie Boudreau

Table of contents

1.0 Introduction

Having completed the first prototype of the climate-shake alarm sensor, the group has a much-improved understanding of the capabilities and limitations of the module. With the code having been completed, and most of the materials arrived, the time for more comprehensive prototypes has arrived. These physical prototypes will help the group even more in creating the best possible module for JAMZ by providing the group with more realistic and comprehensive results from testing. This will allow the group to change and tweak the module, and result in better more accurate outcomes for the client. The prototypes documented below serve to test the dimensional and functional capabilities of each sub-system—physically validated dimensions of the electronic components compared to their respective cases, a mock serial communication between two Arduinos, benchmarked values for future temperature sensor testing, temperature sensor testing using a traditional thermometer, and testing the jerk algorithm under various conditions.

2.0 General Prototyping Objectives

With last week's testing on the various components and subsystems being successful, the goal this week was to consolidate the various parts into one whole module. All the components that were previously tested solos, such as the thermostat and the accelerometer, were tested in unison, with the preliminary case designs being laser cut as well. Additionally, all the components were tested while connected to the same circuit. This marked a shift towards more comprehensive and physical testing. The reasoning behind more comprehensive testing is that with all the individual components having been properly tested, the question facing the group was could these individual subsystems be combined into one whole module?

The goal with prototyping this week is to ensure that there is no fatal flaw in the module and that all the subsystems work together as well as they do separately. The goal with the casing is to ensure that the dimensions were properly measured in the design online. The goal with the testing of the code is to ensure that it can run in sequence on an Arduino, and that serial communication can be established.

3.0 Prototyping Objectives, Tests & Results

3.1 Sensor Case Prototyping Objective

The objective of the sensor module prototype is to correct the faulty dimensions for the screw points of attachment, the inner dimensions where the sensors are housed, and to correct for the material choice. The goal is to use physically verifiable dimensions of the sensors and screws benchmarked against

the first laser-cut prototype. The secondary objective with this new prototype is to update the weight analysis and CAD model with the new dimensions obtained from test fitting the physical components in prototype 1. In summary, these objectives utilize the verified physical metrics of the M3 screws as well as the DHT22 and the accelerometer to update the case design with proper fitting cases.

3.2 Sensor Case Test and Results

The testing method used to correct for the dimensional errors was standard measurement using a ruler. In the first laser-cut prototype the screws did not fit within the holes in the sensors properly, they were too small for an M3 screw. The screw holes on the outside were also too small for the radius of an M3 screw. The sensors themselves also had more room inside the case than necessary, so the outer case dimensions can be reduced while maintaining the proper fitting of the sensors. The analysis below shows the measurements of prototype 1 compared to the new measurements for prototype 2 obtained from trying to fit test the sensors and screws in prototype 1. The weighted analysis is also updated with the correct volume measurements for both prototypes 1 and 2 obtained from OnShape. Finally, version 2 of the CAD model has been updated to reflect the new changes to the dimensions. Looking forward, this prototype is still in progress as it needs to be made into a physical form so the same dimensional analysis can be done to verify these dimensions with more certainty. This will be done using a laser-cut version of the part to verify the dimensions of the part before a 3D print is attempted.

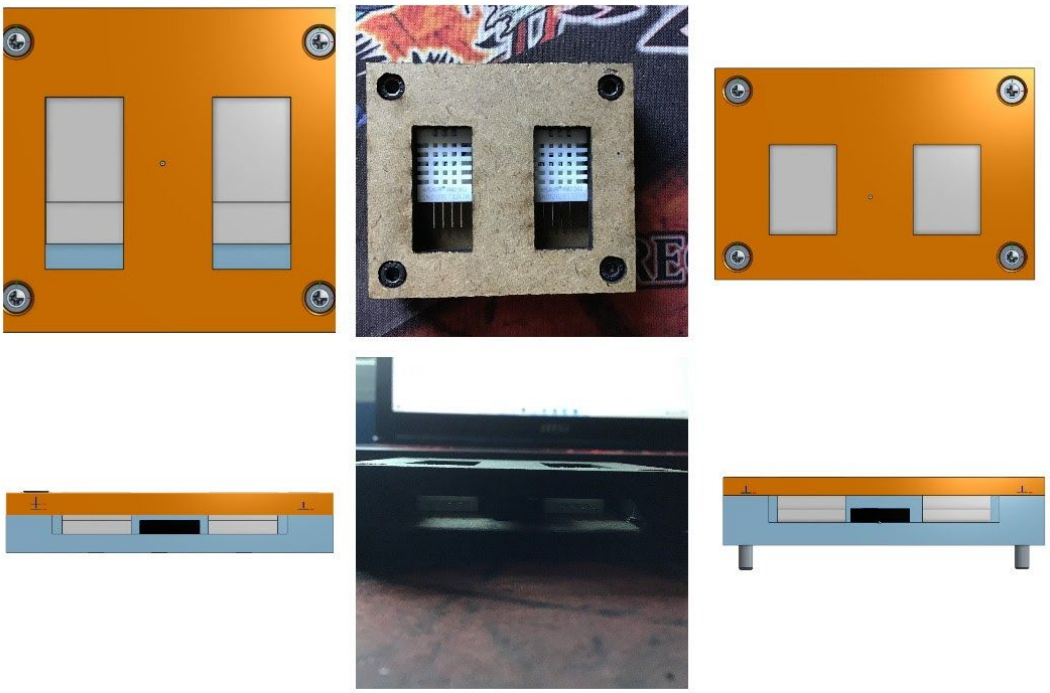

Figure 1: From the left, the first 4 photos show prototype 1 in both a CAD model and a laser cut model. The last two photos show the CAD model for prototype 2.

Dimensional & Weight Analysis:	2021-03-14
Lucas Siviero	

Dimensions	Prototype 1	Prototype 2
Length (mm)	71	71
Width (mm)	62	48
Height (mm)	13	16
Screw holes (mm)	3	3.5

Notes* The length remains the same between prototypes.

The new width reduces the width in order to be more compact, while still fitting the components.

The new height requires more clearance for the wires inside attacched to the accelermoeter

The new screw holes have enough clearance for the 3M screws

Screws must be >16mm (height of the case itself) because they must attach to the drone case design.

Based on the hardware list from JAMZ, the screw heights that will fit are between 20mm or 30mm.

Constants	Value	Units	Quantity
g	9.81	m/s^2	-
Mass DHT22	0.0024	kg	2
Mass LSM6DS3	0.0007	kg	1
PH Countersunk flat head screw M3x0.5 x 20	0.001182	kg	4
PH Countersunk flat head screw M3x0.5 x 30	0.001729	kg	4
Density MDF	0.0008	kg/cm^3	-
Density PLA	0.001024	kg/cm^3	-
Density Acrylic	0.00119	kg/cm^3	-
Volume Protoype 1	40.787667	cm^3	1
Volume Protoype 2	35.599889	cm^3	1

Notes* The volume for prototype 1 ≠ volume from first anaylisis because it was calculated incorrectly.

Screw masses are approximated as they depend on material available.

Using PH Countersunk flat head screw M3x0.5 x 20 (kg)		
Material	Total Mass Prototype 1	Total Mass Prototype 2
MDF	0.042858134	0.038707911
PLA	0.051994571	0.046682286
Acrylic	0.058765324	0.052591868

Note* This assumes 30% infill of PLA

PH Countersunk flat head screw M3x0.5 x 30 (Kg)		
Material	Total Mass Prototype 1	Total Mass Prototype 2
MDF	0.045046134	0.040895911
PLA	0.054182571	0.048870286
Acrylic	0.060953324	0.054779868

The screw lengths depend on how deep the case must be drilled into the drone.

The case must also be durable enough to withstand rain-like conditions.

Since the weight is <<5kg (metric maximum):

The weight of PLA is negligable in comparison to the material durability PLA offers.

The final choice for this prototype is PLA

Figure 2: Dimensional model comparing prototype 1 and 2 of the sensor module based on weight and updated screw masses.

3.3 Arduino Module Prototyping Objectives

The objective for making the case module prototype for the Arduino was to verify that everything was structurally sound and could fit all the components. This was tested by building the casing out of MDF, attempting to put the completed circuit inside of it and exposing the case to impact and adverse conditions such as rain.

3.4 Arduino Module Test Results

The testing proved that none of the Arduino case dimensions were correct. The length and width were the exact sizes of the Arduino, so there was no room to insert it. The height forgot to account for the size of the jumper wire endings and the mini-breadboard. The MDF material was not structurally sound, collapsing very easily, and also ran into issues when exposed to moisture, becoming very soft and flexible.

Dimensional & Weight Analysis:	2021-03-14
Lucas Siviero	

Dimensions	Prototype 1	Prototype 2
Length (mm)	78.65	84.65
Width (mm)	51.4	67
Height (mm)	13.4	46
Screw holes (mm)	5.6	3.5
Notes* The length is bigger to house the arduino correctly		
The new width increases the width to fit the arduino correctly		
The new height requires more clearance for the wires inside attacched to the breadboard		
The new screw holes are reduced to fit the 3M screws properly		
Screws must >46mm with a deep countersink hole because they must attach to the drone case design.		
Based on the hardware list from JAMZ, the screw height that will fit is 40mm		

Constants	Value	Units	Quantity
g	9.81	m/s^2	-
Mass Arduino	0.025	kg	1
Mass Breadboard	0.013	kg	1
PH Countersunk flat head screw M3x0.5 x 40	0.001182	kg	4
Density MDF	0.0008	kg/cm^3	-
Density PLA	0.001024	kg/cm^3	-
Density Acrylic	0.00119	kg/cm^3	-
Volume Protoype 1	24.61398	cm^3	1
Volume Protoype 2	29.21195	cm^3	1
Notes* Screw masses are approximated as they depend on material available.			

Using PH Countersunk flat head screw M3x0.5 x 20 (kg)		
Material	Total Mass Prototype 1	Total Mass Prototype 2
MDF	0.087419184	0.09109756
PLA	0.092932716	0.097641037
Acrylic	0.097018636	0.102490221
Note* This assumes 30% infill of PLA		
The case must also be durable enough to withstand rain-like conditions.		
Since the weight is <<5kg (metric maximum):		
The weight of PLA is negligable in comparison to the material durability PLA offers.		
The final choice for this prototype is PLA		

Figure 3: Dimensional model comparing prototype 1 and 2 of the arduino module based on weight and updated screw masses.

3.5 Accelerometer Circuit Test Objectives

The objective for the accelerometer circuit was to confirm that the circuit, code and libraries functioned together. This was tested using the Arduino IDE, a variety of different libraries and the circuit from the previous prototype. A successful result would be outputting acceleration values to the serial monitor that change when the sensor is moved.

3.6 Accelerometer Circuit Test Results

At first, the accelerometer would not work correctly, no matter which library, wire configuration or code was implemented, as shown in the figure below. Each test was completed 6 times, with the SDA and SCL pins, with the SCX and SDX pins and with 5K Ohm pull-up resistors to SDA and SCL, all with 2 different Arduinos. I2C detect code was also used to find the address of the sensor, returning null, indicating bad wiring, bad code or faulty module. Most likely second or third given that all wiring configs were followed. This was solved by ignoring the datasheet for the sensor, using 5V input instead of 3.3V and resoldering the connections. When this change was made, there were successful results from the sparkFun library, used with I2C wiring configuration and the sample code.

Table 1 - Testing Wiring and Library Configurations for Accelerometer

Library	Analog Pins (4,5)	I2C Built-In Pins	SPI (CS, SDA, SAO, SCL = 10,11,12,13)
SparkFun	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU
BMI 160	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU
Adafruit	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU
SeeedStudio	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU	No SPI Compatibility
AST	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU	No SPI Compatibility
PoloLu	Code compiled, Failed to Initialize IMU	Code compiled, Failed to Initialize IMU	No SPI Compatibility

3.7 Serial Communication Objectives

Communication between the client's computer and the Arduino on the module is important, and so an important part of the code is incorporating serial communication. Testing for this was done using two Arduino boards, and the main goal was to ensure that serial communication could be implemented in the consolidated code in as simple a way as possible. The testing was done by connecting the two boards at their RX TX pins, and writing some code to send a string of data from one to the other. If the communication was successful, then in the serial monitor the data that was sent would be printed. To run the test code, the usage of the Arduino library SoftwareSerial was required.

3.8 Serial Communication Test Results

At first, the code that was written did not compile at all, due to an avrdude error being thrown by the board. After researching, the cause of this was determined to be due to the usage of pins 0 and 1 on the Arduino board, and so initializing different pins was determined to be the best course of action. So after only changing the pins that were used to transmit the data, the tests were successful, and the results showed that using the serial port to communicate between Arduinos required few simple lines of code. These lines of code are very easy to implement, and in no way do they affect any rate-determining steps or other facets of the consolidated code.

3.9 Consolidated Code Objectives

The objective for consolidating the code was to confirm that all the code functioned the same way in the void loop section, shared the same rate-determining steps and could provide sufficient output with every half a second per JAMZ criteria. The testing was done by first testing the individual parts and code. Then, the void loop sections of the code were analyzed for delays and how readings were taken. Next, the complete circuit was created and the codes were combined to only take 1 reading each per void loop, deciding an appropriate rate for the void loop to run and outputting data at the appropriate intervals. Finally, the consolidated code was tested by compiling the code and viewing the results in the serial monitor. If the test was successful there would be data printed on the serial monitor every half a second with no compilation errors and no run-time errors.

3.10 Consolidated Code Test Results

At first, we ran into issues because both codes were taking all their required readings in one void loop, with multiple delays within each loop. This was adjusted so that each sensor took one reading per loop, one delay was used for each loop and data was outputted every fifth void loop. Once the code was compiled there was another error with a null-pointer exception due to a wiring issue with a DHT-22

sensor, this was fixed quickly. After this, the appropriate data was outputted to the serial monitor correctly every half a second as desired.

3.11 Consolidated Circuit Objectives

The objective for consolidating the circuit was to ensure that all the components functioned and to determine the number of wires required in the through cable. The goal was to minimize this number while also maintaining functional components and safety. The test was conducted by first using leads to test all the ground and voltage connections. Then all of the components were connected and tested using the consolidated code. Because of the nature of this test the consolidated code and circuit were co-requisites for success in a way because each will only have the correct output if the other works. A successful test would have the same output as the consolidated code test and determine the optimal number of wires to use without the danger of short-circuiting.

3.12 Consolidated Circuit Tests and Results

Immediately, as expected all of the components worked correctly after the one small issue with the DHT-22 wiring was fixed. It was determined that 7 wires were needed as shown in the following table below. This number was confirmed using LEDs to test for short-circuiting as described in the objectives. There were no issues and this number could not be reduced any more, as all wires were essential. The table could not be shown on tinkercad as none of the parts were available for use. A figure of the wiring is provided below and further explained in the table.

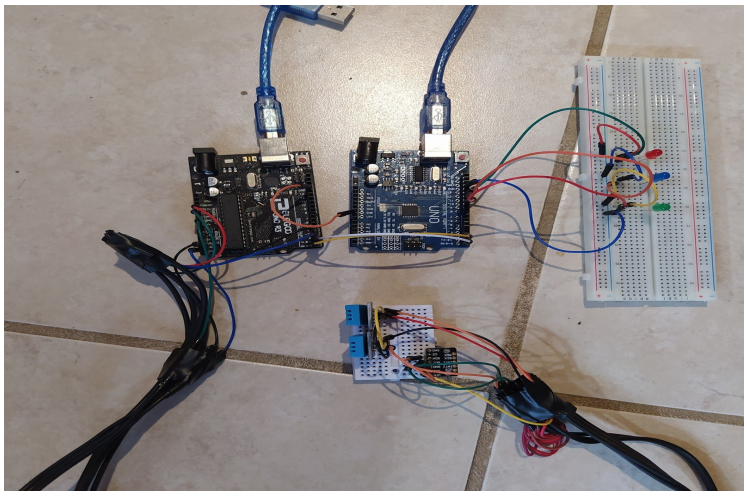

Figure 4: Showing the whole circuit, including the thermostats and the serial communication between Arduino

Table 2 - Results from Testing Number of Wires Required

Wire Function	Wire Destination (s)

5V	LSMDS3 V_{In}
3.3V	DHT-22 #1 VCC, DHT-22 #2 VCC
DHT #1 Signal	DHT-22 #1 Signal
DHT #2 Signal	DHT-22 #2 Signal
LSM6DS3 SDA Signal	LSM6DS3 SDA Signal
LSM6DS3 SCL Signal	LSM6DS3 SDA Signal
Ground	LSM6DS3 GND, DHT-22 #1 GND, DHT-22 #2 GND
Total Number of Wires	7

3.13 Jerk Algorithm Test Objectives

Jerk is a relatively abstract measure compared to displacement, velocity or even acceleration. Because of this, the test objectives were to use practical experimentation, in as controlled a manner as possible. Using the previously tested circuit and code the accelerometer was tested under a variety of movement patterns that it may experience and the binary output for a jerk or no jerk was recorded. The jerk values could have been measured, but because of the low accuracy of the sensor, noise levels and difficulty of determining units, instead, a relative comparison approach was taken. The code was designed so that if a non-zero (with noise-reducing functions applied) jerk was detected multiple times in a period of time a violent shake was detected. A successful test would show that only violent shaking gives a true jerk binary output.

3.14 Jerk Algorithm Test Results

At first, the jerk algorithm was constantly outputting that a jerk was occurring. This was fixed by applying a small threshold for the jerk, thus ignoring any jerk caused by different noise levels from the accelerometer. After this was applied the algorithm worked exactly as expected, as shown in the table below and the linked video. The next step for this test would be to find a way of having quantitative input for acceleration values.

Prototype testing video for jerk algorithm.

Table 3 - Test Results for Jerk Algorithm Using Different Movements

Movement Type	Jerk Result (T/F)
Uniform Straight Line Acceleration in X	F

Uniform Straight Line Acceleration in Y	F
Uniform Straight Line Acceleration in Z	F
Uniform Straight Line Acceleration in X and Reverse Direction	F
Uniform Straight Line Acceleration in Y and Reverse Direction	F
Uniform Straight Line Acceleration in Z and Reverse Direction	F
Circular Motion in XY	F
Circular Motion in YZ	F
Circular Motion in XZ	F
Random Violent Shake in XY	T
Up and Down Violent Shake in Z	T

3.15 Climate Sensor Test Objectives

The objective of the climate sensor testing was to measure the accuracy of the sensors against an industrial temperature and humidity sensor. Because of the time constraints, these tests were only conducted with room temperature conditions and ranges. In future prototypes, the sensors will be tested in a variety of environments. The test results will be the average difference between the results from our sensors and the trusted sensor.

3.16 Climate Sensor Test Results

As shown in the figure below the average accuracy of our climate sensors was around +- 0.5 degrees celsius. This matches the criteria provided by JAMZ. The next step will be more in-depth testing and comparison with accompanying visual representations of data and applying the results in the code to improve accuracy. In addition, by finding the ranges of temperature and humidity for the food delivery we can modify the code and can provide a binary output if the temperature is not in the desired range and so does the humidity. The code will be modified such that, it will take the first average temperature reading to determine whether the food is hot or cold. Then will call the functions to determine if its temperature and humidity are in the benchmarked range for hot and cold food, respectively. Further, the code will be tested with an ice cream in the cardboard box for the cold food and a hot-coffee for the hot food. The serial monitor will show a boolean output true if the food is in the correct temperature and humidity range and false it is not.

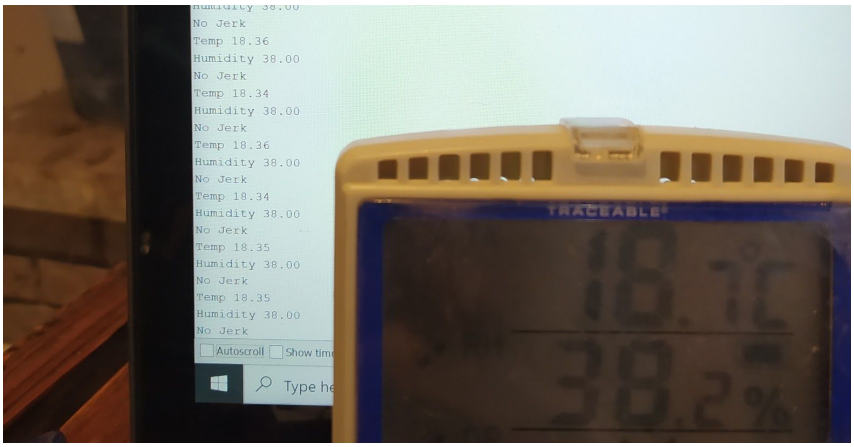

Figure 5: Showing the running code with output. Tested by the thermometer.

Table 4 - Benchmarking the temperature-humidity Ranges

	Skip the dishes	Doordash	Uber Eats
Temperature Range	Hot Food: 57 C or Higher Cold Food: 5 C or lower	Hot Food: 50 C or Higher Cold Food: 5 C or lower	Hot Food: 55 C or Higher Cold Food: 5 C or lower
Humidity Range	50-55%	50%	50-55%

4.0 Conclusion

Having completed the second phase of prototyping for this project, the group feels much more confident and assured in all aspects of the module and project as a whole. Results from the testing demonstrate that serial communication is straightforward and that the code that has been written can give proper results from Arduino's readings. Furthermore, once the circuit is assembled completely it functions properly, and no component receives less voltage/current than it needs. With regards to the casing, incorrect dimensions were identified and fixed promptly. This caught a potential error right in its tracks. The results are positive as a whole and put the group one step closer to finishing the design.

In the next phase, the circuit will be put inside the casing, and the module itself will be tested for functionality. This will mark a shift to testing that is solely physical and comprehensive, in an attempt to tie a bow on this iteration of the prototyping phase.

- Find library, code and wiring configuration for accelerometer
- Consolidate the code into one file that can be run consecutively on the Arduino
- Confirm that all the components function properly once they are put together in one circuit
- Evaluate the first cases that are made, and improve on any deficiencies
- Benchmark the temperature and humidity ranges of the food delivery system in Canada
- Test jerk algorithm under various condition
- Measure humidity and temperature accuracy