GNG1103

Design Project User and Product Manual

Climate Monitoring System for JAMZ Delivery

Submitted by:

Hot Solutions, D12

Joon Lee, 300184520

Mihai Stefanescu, 8690231

Christina Schraa, 300172940

Christian Damignani, 300126797

Ilda Sinoimeri, 300134523

April 11, 2021

University of Ottawa

Table of Contents

1.	Introduction	4
2.	Overview	5
2.1	1 Conventions	7
2.2	2 Cautions & Warnings	7
3 (Getting Started	7
	3.1 Set-up Considerations	7
	3.2 User Access Considerations	8
	3.3 Accessing the System	8
	3.4 System Organization & Navigation	8
	3.5 Exiting the System	9
4 T	Using the System	9
	4.1 User input function (serialEvent)	9
	4.1.1 Temperature/Humidity control	10
	4.1.2 Warning message frequency control	11
	4.1.3 Start/Stop control	11
5]	Troubleshooting & Support	11
:	5.1 Error Messages or Behaviors	11
:	5.2 Special Considerations	12
:	5.3 Maintenance	13
:	5.4 Support	13
6.0	0 Product Documentation	13
(6.1 <physical components=""></physical>	14
	6.1.1 BOM	14
	6.1.2 Equipment list	14
	6.1.3 Instructions	14
(6.2.1 BOM	16
	6.2.2 Equipment List	17
	6.2.3 Instructions	17
	6.3 <electrical component=""></electrical>	20
	6.3.1 BOM	20
	6.3.2 Equipment List	20
	6.3.3 Instructions	21
(6.4 Testing & Validation	21
7 (Conclusions and Recommendations for Future Work	25

8 Bibliography	25
9 Appendix I: Design Files	26
List of Figures	
Figure 1: The Final Prototype	6
Figure 2: Close Up of the Arduino Nano and Multiplexer	6
Figure 3: Flowchart of the Code	10
Figure 4: User Input Function	11
Figure 5: CAD Model of the Arduino Nano Case	15
Figure 6: CAD Model of the Sensor Case	15
Figure 7: Printed Arduino Nano Case	17
Figure 8: Printed Sensor Case	17
Figure 9: Serial Communication Code	18
Figure 10: Plot of Temperature	19
Figure 11: Check Connect Function	19
Figure 12: Spam Prevention Function	20
Figure 13: Condition Check Logic Function	20
Figure 14: Circuit Diagram	22
Figure 15: Serial Communication Test	23
Figure 16: I2C Scanner Test	23
Figure 17: Sensor Disconnected Test	24
Figure 18: Humidity Not Ideal	24
Figure 19: Running the Code	
Figure 20: Complete BOM	
List of Tables	
Table 1: List of Acronyms	4
Table 2: User Selection	7
Table 3: Error Messages	12
Table 4: BOM Physical Components	15
Table 5: BOM Software Components	
Table 6: BOM Additional Electrical Components	21

List of Acronyms

Table 1: A list of acronyms used in this manual and their definitions.

Acronym	Definition			
BOM	Bill of Materials			
CAD	Computer-aided design			
HL	Humidity lower-Represents the lower bound for humidity in the			
	code			
HU	Humidity Upper- Represents the upper bound for the humidity			
	values within the code			
RH	Relative humidity			
TL	Temperature Lower- Represents the lower bound for the			
	temperature within the code			
TU	Temperature Upper- Represents the Upper bound for the			
	temperature within the code			

1. Introduction

The rise in food delivery due to the COVID 19 pandemic and limited coverage within rural communities means a drone operated delivery service could be feasible. Less human resources will be needed to deliver to regions with low profit margins, and delivery labourers can be redirected to reduce bottlenecking. The climate sensor is a necessary component which will relay the temperature and humidity of the compartment which holds food and drinks. A series of subsystems have been defined to construct a fully functioning and balanced climate sensor. To correctly operate the climate sensor, a series of operating standards and user instructions should be followed to ensure proper operation and maintenance.

The first subsystem is the data processor, which is to smoothen the digital output from the Arduino. This system will take the data that is fed by the sensor and will deconstruct it to form a signal which can be read as temperature and humidity. The data processor is supplemented by a visual feedback system. This data is taken from the Arduino and is displayed visually to be easily interpreted by a drone operator. The data processing subsystem attempts to remove random noise from the sensor output to provide a stable and accurate data. This is done in software as it is a digital signal.

The whole system must be protected from the elements in the protection subsystem, which covers the Arduino and all necessary electrical components. The positioning of the climate sensor on the drone is also determined in this step. The two sensors are positioned at the top of the box, while the Arduino case is positioned directly above it. To protect the box from debris and foreign items from entering inside, a sealing subsystem is also defined to ensure that the box does not lose insulation capability despite their being wires inside that extend to the exterior of the box.

The climate/delivery content protection is required to protect the sensor while it is inside the box and exposed to the environment. When the box is retracted and attached to the drone, the sensor will have access to the internal space of the box and may need protection from water and other miscellaneous objects due to shaking or condensation formation with humidity. When the box is lowered, the sensor will be exposed to the outside environment which faces similar issues as the previous case where water from rain, or other debris flying in the air may damage the sensor.

The Arduino protection and casings are required to protect the main control board from damage. It will be placed on the drone but needs protection from the environment as debris and water could damage the board and stop proper data transfer/processing from the sensor to the flight computer.

2. Overview

Amidst the COVID-19 pandemic, JAMZ created a drone delivery business to solve the food delivery problem while following government social distancing regulations. This solution would not only protect the health of the public, but also allow businesses to continue operating while lockdowns are in effect. JAMZ required a lightweight system for accurately monitoring the humidity and temperature of their delivery box. The user must be able to assemble the system and identify the data given by the sensor and code.

This climate system meets JAMZ's requirements and has a few key aspects that makes this system superior. First, the design is extremely lightweight only weighing approximately 60

grams in total. This allows the drone to carry more content as the climate system adds very little weight to the drone and creates very little drag while the drone is in flight. Next the system is very easy to assemble with casings that snap together and only requiring screws to mount the system to the drone. The system also has two sensors which increase accuracy and reliability of data by measuring the climate across the box rather than in one portion. Lastly it is water resistant as there are casings protecting the electronic components of the climate system.

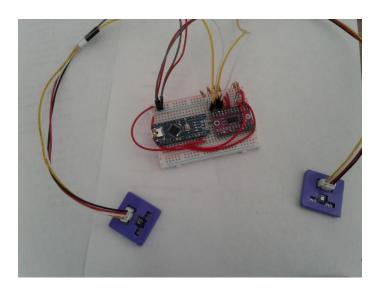


Figure 1: Picture of final prototype

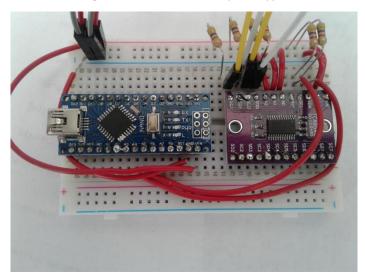


Figure 2: Close up of Arduino Nano with multiplexer

The major function of the system is to relay information back to the drone operator when there are significant changes to the state inside the contents of the package. The operator receives an

error message whenever the sensors read an unideal temperature or humidity value inside the delivery box.

The system is comprised of the following equipment: two sensors (with protective cases), one Arduino Nano (with protective case), one multiplexer, one voltage converter, and wires. These pieces create the climate system and when assembled, can be used by the user.

2.1 Conventions

2.2 Cautions & Warnings

The system is water resistant but is not waterproof as the casings were 3D printed and may still allow water to enter the system.

3 Getting Started

This section provides the user with a general understanding of how the system works, and any steps needed to use the system correctly.

While the user is first setting up the climate system, it is important they become familiar with the circuitry required, as well as the functions provided by the system. This includes the primitive function that turn the system on and off, as well as the user selection function, where parameters to measure ideal temperature and humidity conditions are determined.

3.1 Set-up Considerations

Once received, the user will find five main electrical components that make up the climate system. Two AHT20 sensors gather the temperature and humidity of the delivery box. The multiplexer allows the Arduino Nano to gather data from two input signals. The Arduino Nano gathers data from the sensors and outputs a signal to the user. Lastly, the logic level converter steps down voltage allowing safe communication between the Arduino Nano and the drone's main computer, the Raspberry Pi.

It is the user's responsibility to correctly set up the climate system. Please follow the circuit diagram in Figure 14, *Section 6.3.3 < Electric Component > Instructions*, to set up wiring between the Raspberry Pi and logic level converter pins.

Once the system is properly set up and the drone is in use, the user must select the type of food being transported. As seen in Table 3.1, the system uses pre-set ideal ranges for each type of food. Values measured outside the selected range will send a message to the user when the system is in use.

Table 2: User Selection

Setting	User	Lower Bound	Upper Bound	Lower Bound	Upper Bound
	Entry	(°C)	(°C)	(%)	(%)
Hot	Н	20.00	30.00	20.00	80.00
Soup/Hot	S	25.00	35.00	20.00	80.00
Liquids					
Cold	С	0.00	15.00	20.00	80.00

3.2 User Access Considerations

The climate monitoring system is designed to allow wired communication between its Arduino Nano and any Raspberry Pi device. The current feature is designed in order to pass serial data to the main computer onboard the JAMZ delivery drone.

In the case changes to the onboard computer are made, or alternative uses for the system are intended, it would be advised that the user considers the alternative circuitry or devices required to communicate with the system's Arduino Nano.

3.3 Accessing the System

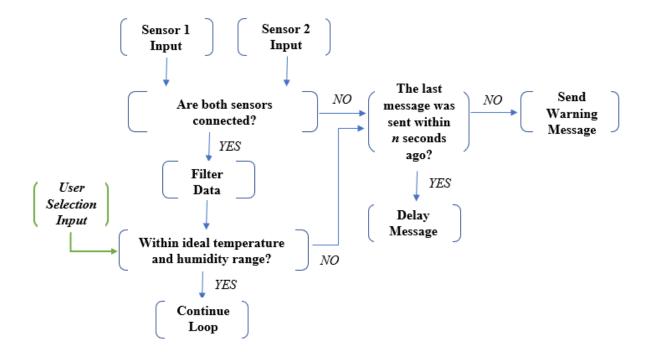
Please follow instructions under Section 3.1 Set-up Considerations.

Once circuitry is completed, the user will be able to turn on the system by entering "c". This indicates to the system the delivery box is closed and climate readings may begin. When the system is on, the user selection feature can be accessed. The user will also be able to receive any output messages from the climate system.

3.4 System Organization & Navigation

The climate system is designed to run with minimal need for the user. The sequence of code run by the system is briefly explained in Figure 3. The only action required by the user while the system is on will be their input from user selection to set the appropriate parameters.

Figure 3 Flowchart of the code for the climate system



3.5 Exiting the System

To exit the system, the user will be required to enter "o", indicating the delivery box is open and climate readings will no longer be sampled.

4 Using the System

There are three features that the operator must know. First is the temperature/humidity control. Using a predetermined character, a message can be sent to the Arduino to change the ideal temperature and humidity values. Second is the warning frequency control. Using a predetermined character, a message can be sent to the Arduino to change the frequency at which the warning messages are transmitted to the raspberry pi. Lastly is the code stop and start control. Using a predetermined character, a message can be sent to the Arduino to start or stop the code. This is all managed by the serialEvent function.

4.1 User input function (SerialEvent)

This function allows the Arduino to receive information from the operator and determine if temperature/humidity, warning message frequency, or start/stop is control desired.

```
void serialEvent(){
 while (Serial.available()) {
   message = Serial.read();
   switch (message) {
   case 'c':
     countC=millis();
     boxOpen=false;
     break:
    case 'o':
     boxOpen=true;
     break:
      /* Additional characters may be added for changing settings in the code dynamically
       * such as temperature range, boxDelay period, messageDelay period, and etc.
   case 'H':
     idealTU = 80;
     idealTL = 20;
     idealHU = 70;
     idealHL = 20;
    case 'T'://Adjust message time delay (in ms)
     mesDelay = 30000;
     break;
     */
   default:
     if((unsigned long)(millis()-countI>mesDelay)){
       Serial.println(F("Invalid input"));
       countI = millis();
```

Figure 4: User input Function

Please refer to line 119 in the "FINAL_CODE" document which can be found by following the link in Table 6 in the appendix.

The parameters values set in the user input function shown in Figure 4- cannot be altered dynamically during use. The code will have to be changed and updated on the Arduino to do so.

4.1.1 Temperature/Humidity control

Parameters controlling ideal temperature/humidity ranges are, idealTU, idealTL, idealHU, and idealHL. The temperature and humidity value is in Celsius and % respectively.

TU represents the upper range of temperature while TL represents the lower range of the temperature. HU represents the upper range of humidty while HL represents the lower range of humidty

The predetermined character and the parameter values associated with the characters may be modified by the user.

4.1.2 Warning message frequency control

Parameter controlling the frequency is mesDelay and the value is in milli-seconds.

The predetermined character and the parameter values associated with the characters may be modified by the user.

4.1.3 Start/Stop control

The character 'o' can be sent to the Arduino to stop the code and character 'c' can be sent to resume the code. The character and parameter associated with the characters should not be adjusted.

5 Troubleshooting & Support

5.1 Error Messages or Behaviors

Table 3: Error messages

Message	Description	Potential Issue	Course of action
Sensor # not	Sensor # is not detected	1.Faulty connection	1.Recheck connection
detected/ Sensor #	by the program	2.Sensor damage	2.Replace
disconnected		3.Multiplexer damage	sensor/sensors
			3.Replace multiplexer
Invalid input	Input received from the	1.Invalid character	1.Send character pre-
	raspberry do not	sent from raspberry pi	defined in the
	correspond to pre-	2.Faulty connection	Arduino
	defined inputs	between serial line	2.Check connection
			between Arduino and
			raspberry pi for
			issues. (Water

			damage, shorting,
			loose connection, etc)
Sensor reading	Temperature/humidity	1.Sensor malfunction	1.Check connection
abnormal	reading is significantly	2.Debris interfering	1.Replace sensor
	different between the	with sensor (Water	2.Remove debris or
	two sensors	droplet near sensor,	wait for debris to
		object blocking	clear
		sensor opening)	3.Repair/replace
		3.Delivery box	delivery box
		damaged (Hole near	
		one of the sensor)	
Humidity not ideal	Humidity exceeds or is	1.Food content	1.N/A
	below the ideal range	increasing humidity	2.N/A
	within the box (This	in box	
	value varies with	2.Outside	
	setting)	environment	
		condition (Rain, fog,	
		etc)	
Temperature not	Temperature exceeds or	1.Food content	1.N/A
ideal	is below the ideal range	increasing 2.N/A	
	within the box (This	temperature	
	value varies with	2.Outside	
	setting)	environment	
		condition	

5.2 Special Considerations

If the resistors need to be replaced, refer to section 9.2.3 in the product manual for TCA9548a for an acceptable range of resistance values for the pull resistors. This manual can be found in Table 6 under the document name "TCA9548a Datasheet" in the appendix.

If any additional information regarding the AHT20 sensor is required, refer to Table 6 under the name "AHT20 User Manual" in the appendix.

5.3 Maintenance

Check the sensor case weekly for any debris covering the sensor opening as that may affect the reliability and accuracy of the climate sensor reading. Remove the debris if there are any.

Regularly check the Arduino case and sensor case for any damage or leaks. If there are any suspicions of water leakage through the case, replace the case to prevent critical system failure.

Occasionally check the wiring for damage, loose connections, or rust. For loose connections, reattach with solder. For minor damage such as a small area of exposed wire through a cut on the rubber coating, cover with electrical tape or any watertight sealant. For rust or severe damage on the wire, relace the affected wire/wires.

5.4 Support

For additional questions or support, please contact slee305@uottawa.ca or cschr045@uottawa.ca.

To report an issue, please contact email list above with the title "Hot Solutions Climate System Issue". Please include your contact information, description of the issue with images if possible, cause of issue if known, and any other relevant information regarding the issue.

For security incidents, once identified and immediate risks have been mitigated, contact the Hot Solutions team with the support contacts listed above and describe the nature of the incident for potential course of action relevant for the climate monitoring system. For the climate monitoring, physical damage to the drone is not possible, but the quality and safety of the food may be impacted if the climate data is tampered with.

6.0 Product Documentation

This section details the building of the final prototype. The final prototype is split into several different subsystems. Physical cases were created for both the Arduino nano and the AHT20 sensors. The software component included finalizing the code so that accurate data was collected

when required. The electrical component involved connecting all the sub-components together and testing that everything worked when connected.

6.1 < Physical Components>

6.1.1 BOM

Table 4: Materials related to the manufactured components of the climate sensor design.

Quantity	Components	Unit Price	Shipping	Total Cost	Source
		(\$)	Cost (\$)	(\$)	
2	3D printed	0.25	0.00	0.34	<u>MakerRepo</u>
	Sensor Case		(pickup)		
1	3D Printed	0.25	0.00	3.85	<u>MakerRepo</u>
	Arduino Case		(pickup)		
2	M3-30mm	0.615	0.00	1.23	Home depot
	Screws		(pickup)		
N/A	Onshape	0.00	0.00	0.00	<u>Onshape</u>
2	AHT20	6.66	0.00	13.32	<u>Digikey</u>
1	Arduino Nano	6.33	0.00	6.33	<u>Amazon</u>

6.1.2 Equipment list

A CAD software was needed to develop designs for the Arduino nano and sensor cases. Once these designs were created a 3D printer was used to create physical prototypes of the Arduino and sensor cases.

6.1.3 Instructions

A CAD model was first created that detailed the design that to be used. Onshape was used for this project. This design was then exported as an STL file and prepared using Ultimaker Cura. Ultimaker Cura is a slicing software. Once the file has been sliced, it is saved as a G file and can be used with a 3D printer.

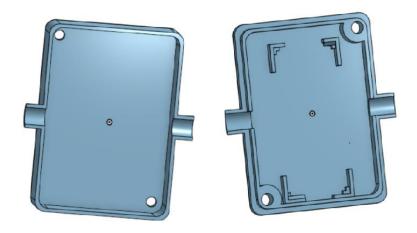


Figure 5: The CAD model of the Arduino Nano Case.

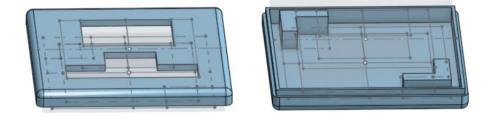


Figure 6: The CAD model for the sensor case.



Figure 7: The printed case containing the Arduino nano.

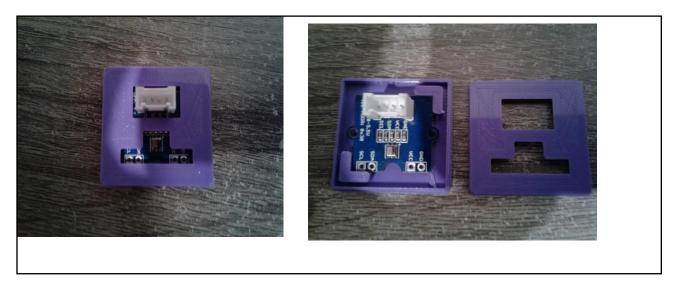


Figure 8: The printed Sensor case with the sensor positioned inside.

6.2 < Software Component>

6.2.1 BOM

Table 5: Materials used for the software component of the climate sensor system.

Quantity	Components	Unit Price (\$)	Total Cost (\$)	Source
N/A	Arduino IDE	0.00	0.00	<u>Arduino</u>
N/A	Adafruit AHT20	0.00	0.00	<u>Arduino</u>
	Library			
N/A	Wire H Library	0.00	0.00	<u>Arduino</u>

6.2.2 Equipment List

The only equipment used during software prototyping was the Arduino coding software.

Additional libraries were downloaded from the Arduino website. The Arduino nano and the AHT20 sensors were also used to test how the code worked.

6.2.3 Instructions

The final code can be broken down into five major parts: serial communication, data processing, error detection, spam prevention, and condition check logic.

```
void setup() {
 //Begins the serial
 Serial.begin(9600);
 while (!Serial) { //Waits until serial begins
   delay(20);
 //initalizes 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");
```

Figure 9: The Serial Communication Code

The serial communication is used to ensure communication between the sensors and the Arduino and between the Arduino Nano and the Raspberry Pi used by the drone.

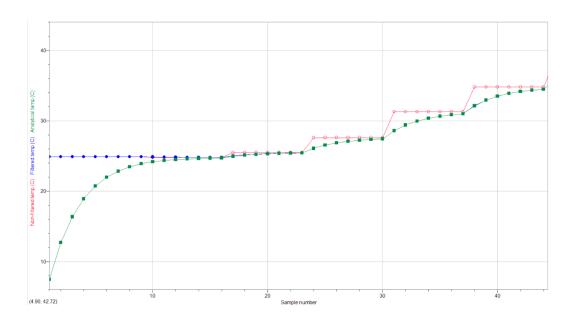


Figure 10: A plot of the temperature (filtered, non-filtered, analytical)

The data processing takes the values given by both sensors and averages them. This averaged value is the value that is communicated to the drone. Taking the average value allows for a more accurate reading on conditions inside the box. An exponential filter was also used in the code to remove noise.

```
* CheckConnect: Checks for connection status and populates the humd and temp with data from the sensor if connected.

* The getEvent provided by the library returns true if connected and collects the data from sensor if it is, removing

* the need to call getEvent again in the void loop.

*/
bool checkConnect(){

if(!(sensor.getEvent(&humd.&temp))){

if(!(sensor.getEvent(&humd.&temp))){

if((unsigned long)(millis()-countD)>mesDelay){

Serial.println(!E("Sensor disconnected"));

countD = millis();

}

return false;
}
else{

return true;
```

Figure 11: The Check Connect Function

The error detection function checks first if the sensors are connected. If the sensors are connected error messages are due to the temperature and humidity values being outside the acceptable range of 20-80 degrees Celsius and 20-80% relative humidity.

```
wt = 5;
wh = 80;
if (Time>10000) {
    if (wt>65||wt<10)
    {
        if ((unsigned long) (Time-countT)>5000) {
            Serial.println((String) "Temperature not ideal: Current temperature = "+wt+"C");
            countT = Time;
        }
        if ((unsigned long) (Time-countH)>5000) {
            Serial.println((String) "Humidity not ideal: Current humidity = "+wh+"%");
            countH = Time;
        }
    }
    if ((unsigned long) (Time-test)>1000)
    {
        Serial.println((String) "Time: "+(Time/1000)+"s");
        test = Time;
    }
}
```

Figure 12: The Spam Prevention Function

The spam prevention function prevents the Arduino from spamming the raspberry pi with unnecessary warnings. It was deemed excessive for the drone operator to be alerted to a problem every tenth of a second. With the spam prevention function, the operator is alerted about an error every ten seconds. The time between error messages can also be easily changed.

```
//Stops the program from running while the box is open
while(boxOpen==true){
    serialEvent():
    delay(300);
}
//Delays the program from running when the box has just recently been closed to wait
//until the conditions inside the box is stabilized
while((unsigned long)(millis()-countC<boxDelay)){
    delay(300);
}
```

Figure 13: The Condition Check Logic Function

The condition check logic function is used to communicate whether the box is open or closed. If the value is set to false, the box is closed, and the sensors will start collecting data. If the value is true, the box is open, and the sensor will stop collecting data so that the operator is not spammed with warnings that are unrelated to food conditions. After the box has been closed the sensor waits before starting back up to allow conditions in the box to stabilize. The time that it waits in the code is ten seconds, this allows for conditions to stabilize without altering the data that will be collected.

6.3 < Electrical Component>

6.3.1 BOM

Table 6: Additional materials related to the circuit.

Quantity	Component	Unit Cost	Shipping	Total Cost	Source
		(\$)	Cost (\$)	(\$)	
6	4.7kΩ Resistors	0.29	0.00	1.74	Amazon
1	Wires (Spool)	1.60	0.00	1.60	Makerstore
1	DC Voltage	5.09	0.00	5.09	<u>Amazon</u>
	Converter				
1	Logic Level	1.09	0.00	1.09	<u>Amazon</u>
	Voltage Shifter				
1	Multiplexer	3.80	0.00	3.80	Amazon

6.3.2 Equipment List

All components in the BOM, except the Arduino and sensor cases, were used to test the electrical system. A breadboard was used to connect all the different components. For a more reliable connection between components, a soldering iron could be used to connect the different components. Wire cutters and strippers were also used to cut the spool of wire that was obtained and make the circuit flush to the breadboard.

6.3.3 Instructions

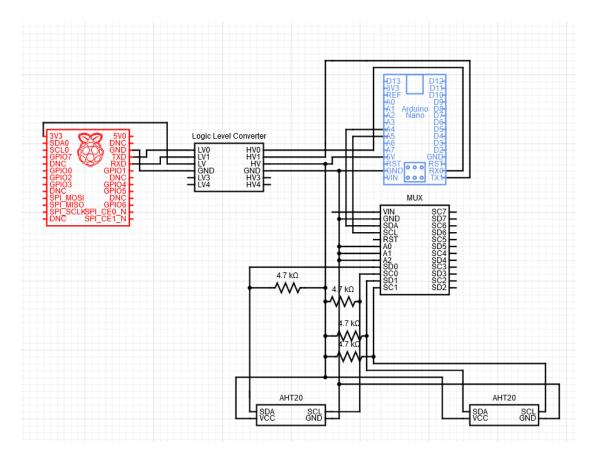


Figure 14: Circuit Diagram of the Climate Sensor System

The circuit diagram shows how the different components in the subsystems were connected. The two sensors contained the same address on the Arduino Nano, so they were connected to a multiplexer. The multiplexer took both inputs and sent the data on to the Arduino nano.

6.4 Testing & Validation

Analytical tests were run on the CAD design to ensure that the sizing for the case was correct. The total volume of the case is 18231.9mm. Additional tests were run on the printed cases. Tests that were run on the Arduino case were waterproof tests and impact resistance tests. To test how waterproof the case was, a small amount of water was poured onto the top of the case and left there. After several hours no water had leaked into the case. To test impact resistance the case was thrown a distance of three metres against a wall. After multiple impacts the case showed no sign of damage. The Arduino nano was still secure inside of the case. This also showed that the case remained together during low impact crashes.

Tests that were run on the code included serial communication tests, I2C sensor tests and tests to ensure that accurate data was being collected. Tests were also run to ensure that the program picked up when the sensor was disconnected, and tests were run to ensure that an error message was given when the temperature and humidity were outside the optimal range.

The serial communication test tested the communication between the Arduino and the Raspberry Pi. This test was run when neither sensor was connected to the Arduino and the results reflect this. The results shown below in Figure 15, show the message that was received by the raspberry pi. As neither sensor was connected this was the expected result.

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

Figure 15: The Results from the Serial Communication Test.

The I2C sensor test was run to ensure that the sensors were being detected by the Arduino when connected to the multiplexer. As shown below in Figure 16 the sensors were detected at the address they were plugged in to. This not only validates that the multiplexer is in full working order, it also shows that the circuit is connected correctly.

```
--- Scan started ---
I2C device found at address 0x38 !
I2C device found at address 0x70 !
--- Scan finished ---

--- Scan started ---
I2C device found at address 0x38 !
I2C device found at address 0x70 !
--- Scan finished ---
```

Figure 16: Results from the I2C Scanner

Error messages were received when a sensor was disconnected and when the temperature and humidity were outside the ideal range.

```
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
```

Figure 17: Results from the Sensor Disconnected Test

After data collection had started, when the sensor was unplugged from the breadboard an error message would be given. This error message conveys to the operator that a sensor is disconnected.

When the temperature values were lower than 20°C and higher than 80°C an error message is also received. When the relative humidity falls below 20% or goes above 80% an error message is also received. These values are outside of the ideal values and this error will be communicated to the drone operator.

```
Temperature 1: 22.31 degrees C
Humidity 1: 91.02% rH
Temperature 2: 25.48 degrees C
Humidity 2: 95.35% rH
Weighted T: 23.86
Weighted H: 92.82
*******
Humidity not ideal
Temperature 1: 22.31 degrees C
Humidity 1: 92.12% rH
Temperature 2: 25.68 degrees C
Humidity 2: 96.27% rH
Weighted T: 23.96
Weighted H: 93.87
*******
Temperature 1: 22.34 degrees C
Humidity 1: 93.14% rH
Temperature 2: 25.90 degrees C
Humidity 2: 97.37% rH
Weighted T: 24.08
Weighted H: 94.91
 *******
Temperature 1: 22.37 degrees C
Humidity 1: 94.04% rH
Temperature 2: 26.12 degrees C
Humidity 2: 98.27% rH
Weighted T: 24.20
Weighted H: 95.86
```

Figure 18: Humidity Not Ideal

When the final code was run the data that was collected was shown to be accurate. The thermometer that the sensors were being tested against read nearly the same value as the sensor. The uncertainty on the thermometer is ± 2 degrees. The uncertainty on the sensor is ± 1 degree. When the uncertainty values are taken into account the temperature given by the sensor and the thermometer agree with each other. As seen in Figure _____ below the temperature and humidity given by the sensor remains consistent when the external environment remains unchanged. When the code is run the sensors give accurate data.

```
remperature 1: 21.59 degrees C
Humidity 1: 27.88% rH
Temperature 2: 22.36 degrees C
Humidity 2: 25.28% rH
Weighted T: 21.97
Weighted H: 26.57
*********
Temperature 1: 21.56 degrees C
Humidity 1: 27.87% rH
Temperature 2: 22.35 degrees C
Humidity 2: 25.27% rH
Weighted T: 21.96
Weighted H: 26.57
********
Temperature 1: 21.56 degrees C
Humidity 1: 27.86% rH
Temperature 2: 22.35 degrees C
Humidity 2: 25.24% rH
Weighted T: 21.95
Weighted H: 26.56
********
Temperature 1: 21.59 degrees C
Humidity 1: 27.87% rH
Temperature 2: 22.35 degrees C
Humidity 2: 25.29% rH
Weighted T: 21.96
Weighted H: 26.55
********
Temperature 1: 21.59 degrees C
Humidity 1: 27.84% rH
Temperature 2: 22.35 degrees C
Humidity 2: 25.23% rH
Weighted T: 21.96
Weighted H: 26.55
```

Figure 19: The Data Received When the Code was Run.

The results of all these tests show that all components of the code function as intended. When the code is run accurate data is gathered.

7. Conclusions and Recommendations for Future Work

Over the course of this project a functional design for a climate sensor module was created. This device was created to be used for drone delivery service and met the clients' requirements. The major components of this design are the two AHT20 sensors, the Arduino nano and the multiplexer. Many aspects of this system were tested and refined over the course of the project until a working prototype was created.

If given more time, possible solutions for removing the multiplexer would be explored to reduce the size and cost of the climate sensor module. Also, adding a waterproof resin and rubber seals to the casings would aid in waterproofing the system and allow for the module to be better equipped for outdoor service. Plans were made to solder the wires of the module but were abandoned due to lack of time. These would be future steps taken if more time was given to work on the climate sensor.

8 Bibliography

- Arduino. (2018, February 5). *SerialEvent*. https://www.arduino.cc/en/Tutorial/BuiltInExamples/SerialEvent.
- BlueDot. (n.d.). *Connecting Multiple Sensors using an I2C Multiplexer*. https://www.bluedot.space/tutorials/connect-multiple-sensors-using-i2c-multiplexer/.
- Lewis, J. (2012, July 16). *Arduino: How do you reset millis()?*. Baldengineer. https://www.baldengineer.com/arduino-how-do-you-reset-millis.html.
- The Robotics Back-End. (n.d.). *Raspberry Pi Arduino Serial Communication*. https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/.
- Wu, C. (2019, October 29). *Tutorial Using the TCA9548A 1-to-8 I2C Multiplexer Breakout with Arduino*. Tronixstuff. https://tronixstuff.com/2019/10/29/tutorial-using-the-tca9548a-1-to-8-i2c-multiplexer-breakout-with-arduino/.

APPENDICES

9 Appendix I: Design Files

The following table is the relevant documents used to arrive/referenced in the document.

Table 7: Reference Documents

Document	Document Location and/or URL	Relation	Issuanc
Name			e Date
AHT20	https://files.seeedstudio.com/wiki/Grove-	Technical	April
Product	AHT20_I2C_Industrial_Grade_Temperature_and_	Document	2020
manual	Humidity_Sensor/AHT20-datasheet-2020-4-16.pdf		
TCA9548a	https://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=	Technical	Novem
datasheet	1618032557963&ref_url=https%253A%252F%252	Document	ber
	Fwww.google.com%252F		2019
FINAL_CO	https://makerepo.com/Cschr045/809.hot-solutions-	Related	April
DE	d12	Document	2020
Deliverable	https://makerepo.com/Cschr045/809.hot-solutions-	Prerequisite	March
С	d12	Document	2020
Deliverable	https://makerepo.com/Cschr045/809.hot-solutions-	Prerequisite	March
D	d12	Document	2020
Deliverable	https://makerepo.com/Cschr045/809.hot-solutions-	Prerequisite	March
Е	d12	Document	2020
Deliverable	https://makerepo.com/Cschr045/809.hot-solutions-	Prerequisite	March
F	d12	Document	2020
Deliverable	https://makerepo.com/Cschr045/809.hot-solutions-	Prerequisite	March
G	d12	Document	2020
Deliverable	https://makerepo.com/Cschr045/809.hot-solutions-	Prerequisite	March
Н	d12	Document	2020

APPENDIX II: Other Appendices

#	Quantity	Components	Unit price (\$)	Shipping cost (\$)	Cost (\$)	Source
1	2	AHT20	6.66	0	13.32	<u>Digikey</u>
2	1	Arduino nano	18.99	0	6.33	Amazon
			(6.33/pc)			
3	NA	Arduino IDE	0	0	0	Arduino
4	NA	Adafruit SHT31 library	0	0	0	Arduino
5	NA	Wire h library	0	0	0	Arduino
6	NA	Onshape	0	0	0	Onshape
7	1	DC voltage converter	5.09	0	5.09	Amazon
8	1	Logic level voltage shifter	10.88	0	1.09	Amazon
			(1.09/pc)			
9	1	Multiplexer	18.99	0	3.80	Amazon
			(3.80/pc)			
10	1	Wires (Spool)	1.60	0	1.60	Makerstore
11	2	3D printed sensor case*	0.25	0 (Pickup)	0.34	Makerpo
12	1	3D printed arduino case**	0.25	0 (Pickup)	3.85	Makerpo
13	1	3D printed voltage shifter	0.25	0 (Pickup)	0.41	Makerpo
		case				
14	2	M3 x30mm screws	1.23	0 (Pickup)	1.23	Home depot
			(0.615/Pc			
)			
14	6	4.7 kΩ Resistors	0.29 /	0	1.74	<u>Amazon</u>
			Resistor			
				Total	38.80	

Figure 20: The full bill of materials for the climate sensor system.