

ECE 198 Design Document - Group 27

Adrien Tarantino, Isimbi Karama, Seth Morrow
Fall 2024

1. Needs Assessment:	3
1.1. Client Definition	3
1.2 Competitive Landscape	3
1.3 Requirements	4
2. Analysis	5
2.1 Design	5
2.2 Scientific/Mathematical Principles	9
3. Costs	11
3.1 Manufacturing Costs	11
3.2 Installation Manual and User guide	13
Overview Photo	13
Components Needed	14
Step 1: Connect the LEDs on Breadboard #1	14
Step 2: Wiring STM32 #1 to Breadboard #1	14
Step 3: Connect STM32 #1 to STM32 #2:	14
Step 4: Connect STM32 #2 to Breadboard #2	14
Step 5: Wiring Breadboard #2	15
3.3 User Guide	16
4. Tests and Risk Analysis	17
4.1 Energy Analysis	17
4.2 Risk Analysis	17
Negative Consequence From using the Design as intended:	17
Negative Consequence From using the Design incorrectly:	17
Negative consequence for using in the design in a manner unintended:	18
Possible way the design could malfunction:	18
4.3 Tests	18
1. Data Serialization:	18
2. Data Deserialization:	18
3. Temperature Range Notification System	18
3. Power Supply - Quick Continuous Power Test	19
4. Verify Resistance with temperature data sheet	19
References	20

1. Needs Assessment:

1.1. Client Definition

As reported in the 2020 Canadian Community Health Survey, 49.5% of households in Nunavut are food insecure [1]. Furthermore, 76% of children in Nunavut live with food insecurity [1]. The primary reason for this food insecurity is the cost of providing food to Nunavut. For example, in 2018, Giant Tiger reported that the cost of stocking shelves at their stores was 11 times greater in Arviat, Nunavut, than Winnipeg, Manitoba [2]. Furthermore, when comparing Nunavut's produce prices to the Canadian average, the price of apples is 80% greater [3], while the price of carrots is 203% greater [3]. Therefore, our clients would be the middle-class people of Nunavut. As stated previously, the people of Nunavut must pay a significantly higher price for groceries than the average Canadian and therefore struggle to provide food.

1.2 Competitive Landscape

Due to the minimal access to roads, as well as the many islands making up Nunavut, food is commonly distributed via ship or plane [2]. However, these modes of transport are incredibly cost-inefficient [2]. As stated in the client section, grocery stores must pay a premium to stock their Nunavut stores, and Nunavut's fresh produce costs far more than the average Canadian. As a response to the high cost of food in Nunavut, as well as other northern provinces and territories, the Government of Canada developed the Nutrition North Canada program (NNC) in 2011 [4]. The goal of the NNC was to provide retail subsidies and allow for cheaper produce to be sold in these provinces and territories [4]. However, in recent years, the NNC has been criticized for not providing enough support, and as of March 2019, the average cost to feed a healthy diet to a family of four in the North was \$422.07 per week [4]. On the other hand, the use of greenhouses has been shown to be a cost-effective and reliable alternative to importing food. In a study testing greenhouses in various communities of Nunavut, it was found that 27% of participants surveyed thought the greenhouses were "good, fresh, and organic", 'local', and that 'they taste better' [5]. However, the same study reported that less-experienced gardeners struggled [5] and would often get poor yields of crops [5].

1.3 Requirements

To make growing food in Nunavut communities more feasible, our product must make it easier to create a greenhouse environment in which it is easier to grow fresh fruits and vegetables. To do this, we have to consider the following:

1. Our product must record the ambient air temperature data within a greenhouse environment to within 2°C of the actual temperature.
2. The microcontroller responsible for measuring the temperature will also need to be able to communicate temp. data to another microcontroller over a distance of at least 1 meter. This will allow one microcontroller to be placed inside of the greenhouse environment, while the other can be placed in a home. This will make the product usable without requiring the user to potentially travel to the greenhouse during adverse weather, risking the environment inside the greenhouse.
3. The second microcontroller must be able to activate and deactivate two LEDs to communicate to the user if the ambient air temperature inside the greenhouse environment is within a preset range. The system must be able to detect and notify with LEDs within 5 seconds when the temperature falls within or outside of a preset threshold range, e.g., between 15°C and 25°C. This range should be configurable by the user in increments of 1°C.
4. The user must also have the ability to activate and deactivate the entire device using an integrated pushbutton on the microcontroller inside the house. The pushbutton should respond within 1 second of activation or deactivation, with a minimum operational lifespan of 50,000 cycles.
5. The product must be able to run continuously when activated by the user, so the device must draw power from a wall outlet. The power drawn should be no more than 30V at any given moment.

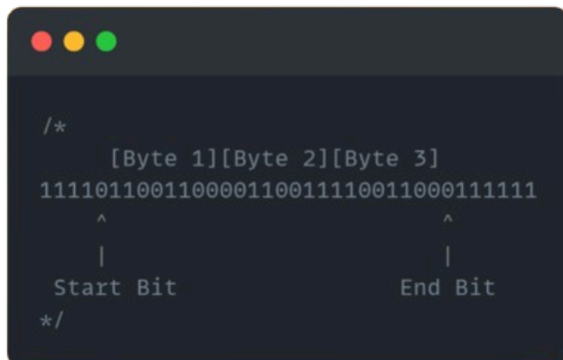
2. Analysis

2.1 Design

Our solution to the described problem will be a custom made environment monitoring device. This device will be used to measure the ambient air temperature (AAT) within a greenhouse environment, as this is how the people of Nunavut will attempt to increase the variety of crops they can grow in their communities. The device will be constructed with two STM32 Nucleo-F401RE microcontroller units (MCU), a 10k Ω negative temperature coefficient (NTC) thermistor, a 10k Ω resistor, a red LED, a green LED, two breadboards, and various insulated copper wires (two of which must be 1+ meters long).

One MCU will be designated the Control MCU (C-MCU), and will display temperature data to the user. The other MCU will be the Measurement MCU (M-MCU), and will be responsible for collecting voltage data and calculating the AAT. The M-MCU will be connected to a breadboard circuit which will provide a thermistor with a constant voltage of 3.3 volts. The thermistor's resistivity will change depending on the temperature of its surroundings [6]; a voltage divider will allow the M-MCU to record the resulting difference in voltage across the circuit caused by the change in resistance of the thermistor. Because it is an NTC thermistor, when the AAT increases, the resistivity of the thermistor decreases, and so will the voltage, and vice versa. The C-MCU will take the difference in voltage value and perform a calculation using a simplified Steinhart-Hart equation outlined in the mathematical principles portion of the document, to calculate the AAT.

In order to communicate data between the two MCU's we will use a custom communication method which works similar to UART. The M-MCU will be responsible for sending data, and the C-MCU will be responsible for receiving the data. Both the M-MCU and C-MCU will have a GPIO pin connected to each other by a copper wire, as well as another copper wire connecting the grounds. Furthermore, both will operate at the same clock cycle speed, and will send and read data at the same speed. The GPIO pin on the M-MCU is outputting 3.3v by default. When the C-MCU's GPIO pin reads 3.3v, it ignores the data read since nothing is intended to be sent. When the sending MCU needs to send data, it will set its GPIO pin to 0v for one clock cycle, and follow by sending 3 bytes of data encoded by a one equalling 3.3v and a 0 equalling 0v.

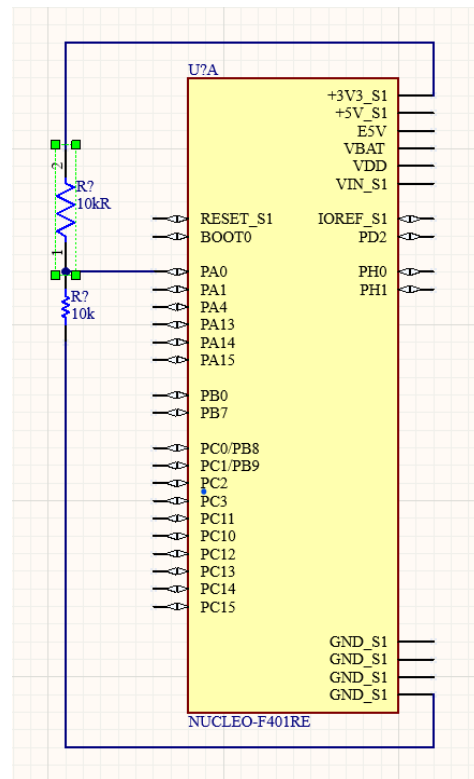
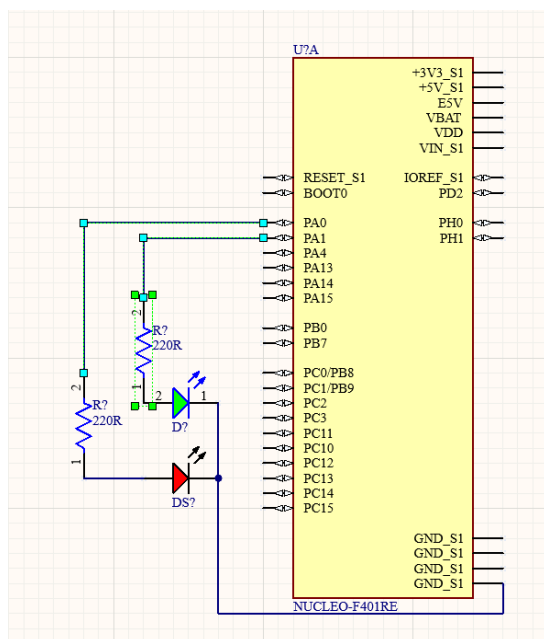


Likewise once the GPIO pin on the C-MCU reads a zero, it will begin to read the data being sent to it, reading one bit of data per clock cycle. It will save this read data to 3 different integers; one for each byte.

The C-MCU then determines if the AAT is within an acceptable, predetermined range of temperatures. This temperature range will be hardcoded into the product before it is installed in the user environment. If the calculated AAT is within the acceptable range, the C-MCU will keep a green LED on. If the AAT drops below the acceptable range, or spikes above the acceptable range, the MCU will turn off the green LED, and turn on a red LED until the temperature is again calculated to be within the accepted limits. Because the device will continuously monitor and check the temperature values, once the temperature returns to within the acceptable range, the green LED will again turn on, and the red will turn off.


To initiate the above process, the user will press the integrated blue push button on the C-MCU. This button will toggle the system from remaining on standby (on, but not monitoring), to monitoring the greenhouse environment (calculating, transmitting, and displaying temperature data).

Circuit Schematics:



Software Design Outline:

LEDS:



```
while (true) {
    bool condition1, condition2;

    // Call the function to get updated conditions
    GetConditions(condition1, condition2);

    if (condition1) {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET); // Turn on LED1
    } else {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_RESET); // Turn off LED1
    }

    if (condition2) {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET); // Turn on LED2
    } else {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET); // Turn off LED2
    }
}
```

For the Led's a simple program featuring a while loop is used to constantly monitor the conditions of the greenhouse. If condition1 is true, that means the green house is in an acceptable temperature range and the green light will turn on, showing the user that the greenhouse conditions are stable. If condition 2 is true, that means the greenhouse conditions outside of acceptable range and the red light will turn on, notifying the user. These conditions will be constantly monitored using a while loop and once conditions change the LEDS will change accordingly.

Thermistor:

```
//Variables needed for calculations:
//Known value of resistor placed in series with thermistor (10k ohm):
const float resistor = 10000.0;
//Known resistance of thermistor at a "nominal" temperature (25 degrees celcius):
const float nominalResistance = 10000.0;
//Temperature of nominal resistance (25 degrees celcius):
const float nominalTemperature = 25.0;
//Beta coefficient of thermistor:
const float beta = 3380;
//Max value of STM32 ADC (range is 0-4095):
const int ADCMax = 4095;
//Voltage being supplied to the circuit from the STM32:
const float inputVoltage = 3.3;
//Final calculated temperature value found using the Steinhart-Hart equation:
const int temperature();
//Measured voltage as a result of the voltage divider circuit
float outputVoltage{};
//Calculated resistance of the thermistor:
float thermistorResistance{};
//ADC value measured at the input pin of the STM32:
int ADCValue{};

//Function takes variables defined above and calculates the ambient air temperature:
float calculateTemperature(){

    //Calculates the output voltage of the thermistor circuit for later use in calculation:
    outputVoltage = ADCValue * (5.0 / ADCMax);

    //Calculates the resistance of the thermistor:
    thermistorResistance = resistor * ((inputVoltage / outputVoltage) - 1);

    //The steinhart-Hart equation is used to calculate the final temperature:
    temperature = (1 / (beta / log(thermistor/nominalResistance) + 1/(nominalTemperature + 273.15))) - 273.15;

    return temperature;
}
```

The function “calculateTemperature” takes in both hard-coded and measured variables to calculate the temperature the thermistor is exposed to. The above code first initializes variables of known value (resistor, nominalResistance, nominalTemperature, beta, ADCMax, inputVoltage). It then initializes variables that will be acquired by reading pin values in the final prototype (outputVoltage, thermistorResistance, ADCValue). The function calculateTemperature will first calculate the output voltage created by the voltage divider, and save that value to outputVoltage. It will then use the output voltage in an interpolated linear equation to calculate the value of the thermistor’s resistance at that moment, which will be stored in thermistorsResistance. Finally, the function will enter all known values into a Steinhart-Hart equation to calculate the temperature and store it in a variable called temperature, and return the value of temperature.

2.2 Scientific/Mathematical Principles

Data Serialization Algorithm

The purpose of the data serialization algorithm is to send data from the M-MCU to the C-MCU. The algorithm uses bitwise AND as well as right bit shifting to serialize the data. The algorithm works as follows:

```
void serializeValue(int values, int clockSpeed) {
    // Runs 8 times because it's sending 8 bits.
    for(int i{0}; i < 8; ++i) {
        // If the rightmost bit is a 1, send a 1.
        // If the rightmost bit is a 0, send a 0.
        if(value & 1 == 1) { setDataPin(1); }
        else { setDataPin(0); }
        // Rightshift the value by 1 bit,
        // Wait until the next clock cycle.
        value >>= 1;
        HAL_Delay(clockSpeed);
    }
}
```

The code will have the M-MCU first check the rightmost bit of a measured value. If the rightmost bit is equal to a 1, the M-MCU sets its designated data pin to 3.3v for one clock cycle. If the right most bit is equal to a 0, the M-MCU will set its data pin to 0v. Finally the number is right shifted by one bit. This process repeats until all bits are sent.

Data Deserialization Algorithm

The purpose of the data deserialization algorithm is to allow the C-MCU to read incoming sequences of 3.3v and 0v from the data pin, and interpret it as some value. The algorithm relies on both the bitwise AND operation, as well as the left bit shift operator.

```
int deserializeValue(int clockSpeed) {
    int result{0};
    // Runs 8 times because it's receiving 8 bits.
    for(int i{0}; i < 8; ++i) {

        // Reads the incoming bit value,
        // adds it to the total sum.
        int resultBit = readDataPinValue(GPIOA, dataPin);
        result += resultBit << i;

        // Waits until the next clock cycle
        // before reading the next bit
        HAL_Delay(clockSpeed);
    }
    // Returns the results of the data recieved.
    return result;
}
```

The code will have the C-MCU first initialize a variable called result. The code will then read the voltage of the data pin. If the voltage is 3.3v, it will add 1 left shifted by the amount of times data has been read ($1 \ll i$) to the total result. If the voltage is 0v, it will add 0 left shifted by the amount of times data has been read ($0 \ll i$) to the total result. The program will then wait one whole clock cycle, and then repeat until it has read 8 bits of data.

Steinhart-Hart Equation:

The Steinhart-Hart equation is an equation used to accurately represent the resistance of NTC thermistors at varying temperatures, and reads as follows:

$$T = \frac{1}{A + B \ln(R) + C(\ln(R))^3}$$

“T” is in degrees Kelvin and “A”, “B”, and “C” are coefficients. We can simplify this equation as follows:

Firstly, we realize that this equation is meant to be accurate over a temperature scale much larger than we will need in our use case. We will only need our thermistor to be accurate within a range of -50°C, to 15°C, as these are the average yearly high and low temperatures for the nunavut region [7]. The value of $C(\ln(R))^3$ becomes important when finding temperature values outside of a temperature range of -50°C to 150°C (this range is specific to our NTC thermistor). Because we will be staying comfortably within these ranges, we can neglect this “C coefficient.” Taking the reciprocal of each side, we now have:

$$T = \frac{1}{A + B \ln(R)}$$

Next, we can eliminate “A” and “B” values by utilizing information about the resistor provided to us from the manufacturer. Provided in the data sheet of the NTC from the manufacturer is the known value of “B” known as the beta value, which in our case is . Helpfully, the manufacturer also provides us with a known value of A at a known temperature. In our case, at a known reference temperature value of 25°C (approximately room temperature), “A” is equal to the reciprocal of the thermistor’s resistance at 25°C, which is 10k ohm’s. We are now able to simplify the equation into a form that is easily translatable to code:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right)$$

Where “T” is final temperature, “To” is the known reference temperature, “B” is the Beta-value, “R” is the resistor of a known resistor placed in series to the thermistor in the circuit, and “Ro” is the thermistors reference resistance.

By writing code to store these values as variables, and by creating a function that plugs them into the finalized equation above, our STM32 will be able to accurately find the ambient air temperature inside the greenhouse environment [8].

Kirchhoff's Voltage Law - Voltage Divider Rule

The voltage divider rule is a function used to determine the output voltage, V_{out} , in terms of resistance R_x . In this context, R_x refers to the value of the resistor connected to ground. For our use case, the resistance R_x is determined by the thermistor and the V_{out} is read by the M-MCU.

The formula for this function is as follows: $V_{out} = V_{Supply} \left(\frac{R_x}{R_x + R_{Supply}} \right)$

Note that R_{Supply} is the resistor connected to the 5v supply voltage, and V_{Supply} is the 5v supply voltage.

3. Costs

3.1 Manufacturing Costs

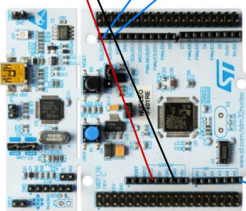
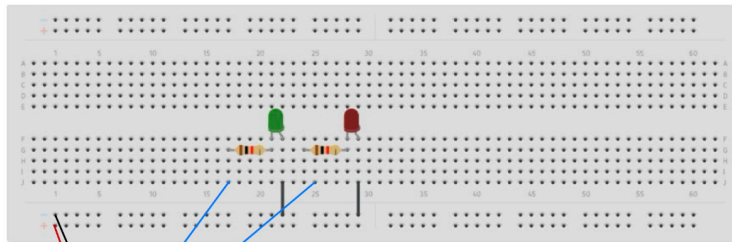
Part	Manufacturer	Distributor	Cost
STM32F401RE Nucleo-Board w Male [9]	STMicroelectronics 350 Burnhamthorpe Rd W, Mississauga, ON L5B 3J1	Amazon Seattle, Washington	= (\$43.14 + tax)*2 = \$48.75 each
Solderless Flexible Breadboard Jumper [10]	RGBZONE Shenzhen, Guangdong 518000,CN	Amazon Seattle, Washington	= (\$12.99 + tax)/120pc =\$ 0.11 each
Thermistor [11]	SOLUSTRE China	Amazon Seattle, Washington	= (\$10.79 + tax)/ 100pc = \$0.12 each
100 ohm Resistor [12]	Uxcell 223 Hing Fong Road Kwai Fong New Territories 00000 Hong Kong, China	Amazon Seattle, Washington	=(\$10.99 + tax)/40pc = \$0.30 each
100k ohm Resistor [13]	E-PROJECTS 4004 97 St NW Unit 44 Edmonton, AB T6E 6N1	Amazon Seattle, Washington	= (\$9.12+tax)/ 10pc = \$1.03 each
1pcs Solderless PCB Breadboard Prototype	MMOBIEL	Amazon	= (\$9.99 + tax) = \$11.28 each

Circuit [14]	Kleibultweg 101, 7575 BW Oldenzaal, Netherlands	Seattle, Washington	
LED's [15]	GikFun China	Amazon Seattle, Washington	= (\$8.88 + tax)/100pc = \$0.10 each

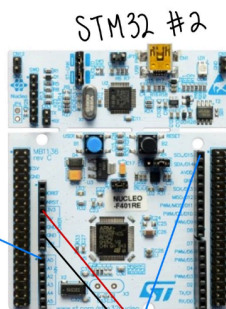
3.2 Installation Manual and User guide

Overview Photo

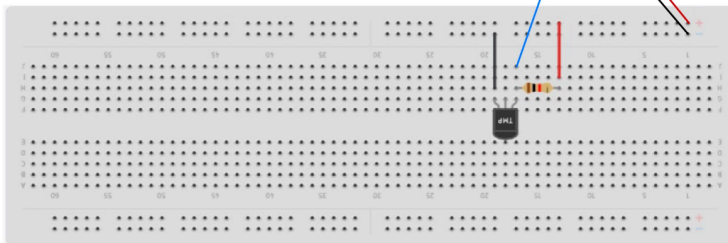
BREAD BOARD #1



STM32 #1



STM32 #2



BREAD BOARD #2

Components Needed

- 2 STM32 boards (labeled STM32 #1 and STM32 #2)
- 2 Breadboards (labeled Bread Board #1 and Bread Board #2)
- 1 Green LED
- 1 Red LED
- 2 100 ohm Resistors (connected to the LEDs)
- Jumper wires

Step 1: Connect the LEDs on Breadboard #1

- Insert the green LED into the breadboard. Ensure the longer leg (anode) is on the left side.
- Insert a resistor between the left leg of the green LED and the ground (horizontal row at the bottom of the breadboard).
- Repeat the above steps to place the red LED to the right of the green LED with its own resistor connected to ground.

Step 2: Wiring STM32 #1 to Breadboard #1

- Place STM32 #1 near Breadboard #1.
- Connect the ground (GND) pin of STM32 #1 to the ground rail on BreadBoard #1 using a black jumper wire.
- Connect two GPIO pins from STM32 #1 to the anodes (positive legs) of the green and red LEDs on Breadboard #1 using blue jumper wires.

Step 3: Connect STM32 #1 to STM32 #2:

(DISCLAIMER*** Scale: the stm32 are closer together then expected. Scale accordingly)

- Use a red jumper wire to connect the VCC (3.3V) output from STM32 #1 to the VCC pin on STM32 #2.
- Connect a black jumper wire from the ground (GND) pin of STM32 #1 to the ground pin on STM32 #2.

Step 4: Connect STM32 #2 to Breadboard #2

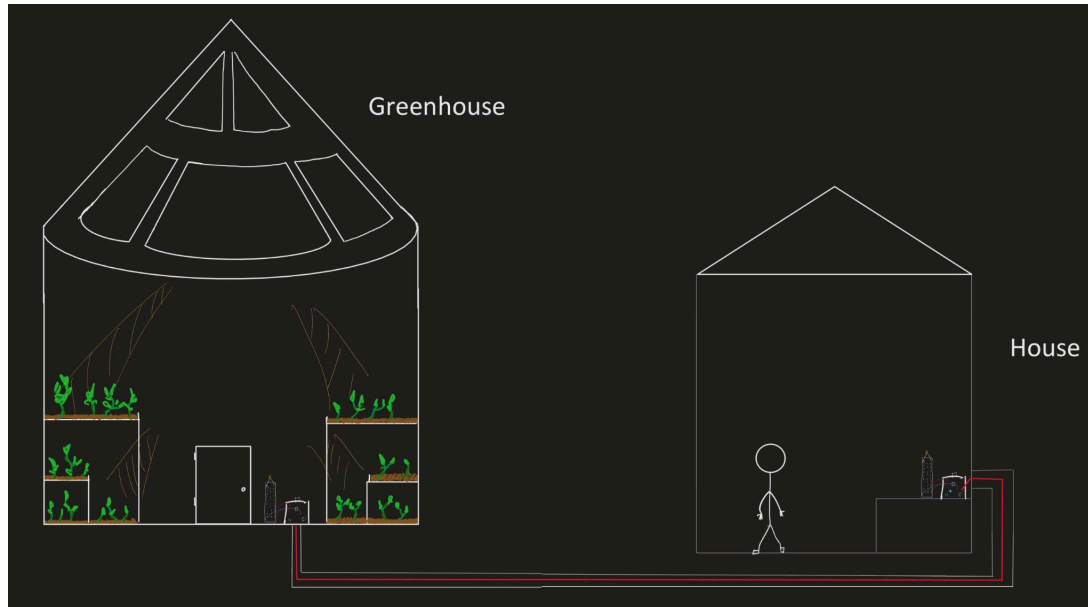
- Insert the voltage regulator (4-pin IC component) into Breadboard #2, as shown in the image.
- Use a red jumper wire to connect VCC from STM32 #2 to the input pin of the voltage regulator on Breadboard #2.
- Use a black jumper wire to connect ground (GND) from STM32 #2 to the ground rail on BreadBoard #2.

Step 5: Wiring Breadboard #2

- Place a resistor between the output pin of the voltage regulator and a horizontal row on the breadboard.
- Following these steps should help you replicate the setup in the image. Make sure to double-check each connection for accuracy, especially the power and ground connections, to avoid short circuits or potential damage to components.

3.3 User Guide

1. Place the STM microcontrollers #2 in the room with the plant you want to monitor
2. Place the STM microcontroller #1 in the room you want to monitor the plant from.
3. Turn on the STM microcontroller #1 by pressing the embedded blue button to turn on the microcontroller.



4. Tests and Risk Analysis

4.1 Energy Analysis

To calculate the total power consumed we will state the following assumptions:

1. All components are connected in parallel to the same power supply
 - a. This will assume the maximum wattage present
2. The STM32 Nucleo-F401RE has the following power requirements
 - a. 5V power supply [9]
 - b. 20mA current draw from power supply [9]
3. The LEDs used have the following power requirements
 - a. 5v LEDs [10]
 - b. 50mA current draw from power supply [10]

$$P_{Total} = V_{Supply} (\sum I_{Each\ Component})$$

$$P_{Total} = 5(0.020 + 0.020 + 0.050 + 0.050)$$

$$P_{Total} = 5(0.14)$$

$$P_{Total} = 0.7Watts$$

$$0.7\ Watts < 30\ Watts$$

Therefore our project will be well under the current limit.

4.2 Risk Analysis

Negative Consequence From using the Design as intended:

Environmental Impact on readings: It is highly probable that condensation, dust, or soil, will build up on the temperature sensor, impairing its accuracy. This can lead to incorrect temperature sensing, and the agriculture could be exposed to unfavorable conditions, even when the reading deems that the temperature is at an “acceptable range”. If left unfixed, the plant’s health could be left in a detrimental state.

Negative Consequence From using the Design incorrectly:

Using the greenhouse for a plant that survives outside of the hard coded temperature range: For the intent of our design, we have decided to set the acceptable temperature range to match the optimal temperature conditions for a carrot. If a plant that has a different temperature range is placed in the greenhouse without changing the settings via code, then the design will work incorrectly for that plant and it will lead to plant loss.

Negative consequence for using in the design in a manner unintended:

Soil temperature Monitoring rather than Air temperature: For our design, Its temperature monitoring is to be done in its surrounding rather than the soil temperature. If the temperature sensor used in this project is placed in the soil instead, first inaccurate readings will impair the system, and second the soil's moisture may cause further internal damage to the system, increasing the risk of short circuits and further component damage.

Possible way the design could malfunction:

Overheating of Components: Continuous operation of the STM32 Microcontroller and sensors in an enclosed greenhouse environment could lead to overheating, especially when growing warmer climate agriculture. This could lead to the increased risk of fire if the main microcontroller unit reaches critical temperatures.

4.3 Tests

1. Data Serialization:

Test Method: To test if the M-MCU is serializing the data correctly, we can use the USB Logic Analyzer with sigrok PulseView made by SparkFun [11]. This tool allows us to view digital logic on an easy-to-read graph [11].

Pass/Fail Criteria: In order to test if the data is being serialized correctly we can read the graph and inspect whether or not the output is as expected.

2. Data Deserialization:

Test Method: After the data serialization has been tested, we can test if the C-MCU is receiving data correctly by verifying using test values and printing data received to the Serial Monitor.

Pass/Fail Criteria: For example if we send a value of 42 to the C-MCU, the C-MCU should output a value of 42 to the serial monitor. If any other value is shown, the test fails.

3. Temperature Range Notification System

Test Method: Simulate different temperatures within the device's range and record the LED response times when crossing preset thresholds (e.g., 22°C and 24°C). For example, increase the temperature slowly from below 22°C to above 24°C and observe LED state changes.

Pass/Fail Criteria: The LEDs should change state within 5 seconds of crossing the threshold. Repeat the test at multiple points to ensure consistent responsiveness.

4. Power Supply - Quick Continuous Power Test

Test Method: Plug the device into a standard wall outlet and monitor for an hour to ensure it powers continuously without interruptions or overheating.

Pass/Fail Criteria: The device should remain powered on without drawing excessive power or shutting down, verifying initial continuous operation stability.

5. Verify Resistance with temperature data sheet

Test Method: Measure the resistance values across the thermistor in several areas with varied, known ambient air temperatures. Compare the collected data to the resistance vs temperature data sheet provided by the manufacturer.

Pass/Fail criteria: If the values of both sets of data match, then the thermistor circuit is accurately measuring the ambient air temperature to within 2°C. If the data does not match, then there is a flaw in the design of the circuitry.

References

- [1] "Rates | Nunavut Food Security Coalition," Nunavutfoodsecurity.ca, 2014.
<https://www.nunavutfoodsecurity.ca/en/rates> (accessed Oct. 26, 2024).
- [2] K. Stucyk and Athabasca University, "Good Governance of Food Security in Nunavut ," Good governance of food security in Nunavut, pp. 9-10, May 04, 2018.
https://epe.lac-bac.gc.ca/100/201/300/jrn_food_research/2018/JFR-V7N4-All.pdf#page=14 (accessed Sep. 17, 2024).
- [3] Nunavut Bureau of Statistics. "2018 Nunavut Food Price Survey." Government of Nunavut.
https://www.gov.nu.ca/sites/default/files/documents/2022-11/food_price_survey_statsupdate_2018.pdf (accessed Sept. 17, 2024).
- [4] O. Leblanc-Laurendeau, "Food Insecurity in Northern Canada: An Overview," lop.parl.ca, Apr. 01, 2020.
https://lop.parl.ca/sites/PublicWebsite/default/en_CA/ResearchPublications/202047E (accessed Oct. 26, 2024).
- [5] A. Lamalice et al., "Building food security in the Canadian Arctic through the development of sustainable community greenhouses and gardening," Écoscience, vol. 25, no. 4, pp. 325–341, Jul. 2018, doi: <https://doi.org/10.1080/11956860.2018.1493260>. (accessed Sep. 25, 2024).
- [6] P. Kane, "Thermistors/Temperature Measurement with NTC Thermistors," www.jameco.com.
<https://www.jameco.com/Jameco/workshop/TechTip/temperature-measurement-ntc-thermistors.html> (accessed Oct. 30, 2024).
- [7] "Nunavut Climate, Weather By Month, Average Temperature (Canada) - Weather Spark," weatherspark.com. <https://weatherspark.com/countries/CA/14> (accessed Oct. 30, 2024)
- [8] Ametherm, "NTC Thermistor - Steinhart and Hart Equation | Ametherm," Ametherm, 2016. <https://www.ametherm.com/thermistor/ntc-thermistors-steinhart-and-hart-equation> (accessed Oct. 30, 2024).
- [9] "NUCLEO-F401RE - STMicroelectronics," STMicroelectronics, 2022.
<https://www.st.com/en/evaluation-tools/nucleo-f401re.html#documentation>
- [10] "Gikfun 5mm LED light assorted kit DIY leds for Arduino (pack of 100pcs) EK8437," Amazon.ca: Electronics. <https://shorturl.at/RAgXB> (accessed Oct. 30, 2024).

[11] element14 presents, "Instrument Basics: Logic Analyzer - Workbench Wednesdays," YouTube, Nov. 06, 2019. <https://www.youtube.com/watch?v=u1DYs2l-IU> (accessed Oct 30, 2024)

[12] "MMOBIEL 1pcs solderless PCB breadboard prototype circuit board – 1x830 point - compatible with DIY Arduino, Raspberry Pi 2/3 / 4/5 projects Proto Shield Distribution - connecting blocks," Amazon.ca: Electronics, <https://shorturl.at/i7hUI> (accessed Oct. 30, 2024).

[13] "Uxcell 40Pcs 100 ohm resistor, 3w 5% tolerance metal oxide film resistors, lead, Flame Proof for DIY electronic projects and experiments," Amazon.ca: Industrial & Scientific, <https://shorturl.at/hrdVm> (accessed Oct. 30, 2024).

[14] "Solustre 100pcs thermistor probe transducer temperature sensors negative temperature coefficient thermistors," Amazon.ca: Industrial & Scientific, <https://tinyurl.com/5n8934dd> (accessed Oct. 30, 2024).

[15] "RGBZONE 130pcs solderless flexible breadboard jumper wires male to male for Arduino breadboard and robotics and raspberry," Amazon.ca: Electronics, <https://tinyurl.com/bdekudsu> (accessed Oct. 30, 2024).

[16] "Nucleo-F401RE STM32 nucleo-64 development board with STM32F401RE MCU, supports St Morpho connectivity," Amazon.ca: Electronics, <https://tinyurl.com/yv82hf6a> (accessed Oct. 30, 2024).

[17] "Pearl-Shaped Precision NTC Thermistor for Temperature Measurement." Available: https://www.cantherm.com/wp-content/uploads/2017/05/cantherm_mf52_1.pdf (accessed Oct. 30, 2024).