College of Engineering
UNIVERSITY OF GEORGIA

Final Project Report

ELEE 4230

Taylor Reese

1 Table of Contents

2 Abstract

New smoker style grills, such as a Recteq, come with controllers for grills installed on them from the factory. These controllers allow the user to set grill temperatures for the grill to maintain on its own from that point forward, as well as monitor food temperatures. The goal of this project is to create a sensor system that replicates this functionality for older smokers that did not come equipped with electronic controllers. Therefore, the goal is to create a controller that senses food and grill temperatures and can use those readings to maintain a set grill temperature without runaway overheating or choked out coals.

A fully working controller was created using an Arduino mega, motor shield, and LCD screen with a keypad for processing, control, and user interface. Two type K thermocouples were used to sense temperatures, and a fan was used to control air flow to the coals burning inside the smoker. The controller has a functioning user interface that allows users to input grill and food temperature setpoints and displays the food and grill temperatures during operation. The grill controller uses the readings to gradually adjust the grill temperature up and down to prevent runaway heating or cooling conditions by changing fan run settings based on the temperature readings.

3 Problem Identification

Unlike some new and extremely expensive grills, many smokers or grills such as a Big Green Egg (See Figure 1) do not come with any type of electronic controls. These types of grills rely on the person smoking on their grill to spend countless hours beside the grill adjusting the draft air intakes (See Draft Door in Figure 1) and exhaust ports (See Top in figure 1 to maintain the perfect temperature for their smoke. This is far from ideal and can allow for the grill to get way too hot or cold, ruining the cook, if it is ignored for even a few minutes. So, the problem this project attempts to solve is the instability built into smokers without any kind of electronic control. Expanding that out into a more detailed problem list, nonelectric smokers cannot:

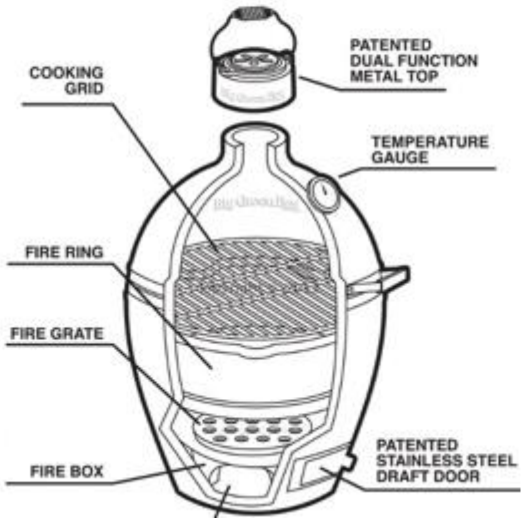

COOKING GRID
PATENTED DUAL FUNCTION METAL TOP
TEMPERATURE GAUGE
FIRE RING
FIRE GRATE
FIRE BOX
PATENTED STAINLESS STEEL DRAFT DOOR

Figure 1: Big Green Egg

1. Sense food temperature
2. Sense grill temperature
3. Use food and grill temperatures to adjust air flow to coals being used to fuel the smoker.

To solve this problem, a portable/removable grill controller needs to be created that uses at least two sensors to read the food and grill temperatures, as well as have a method of determining what temperature the grill should be and controlling air flow to the coals to maintain that temperature.

4 Sensor and Actuator Selection

As established in the problem identification section, the grill controller, at minimum, needs to sense the food temperature, grill temperature, and grill temperature setpoint. This grill controller will also sense a food temperature setpoint. Using those things, the grill controller has to control the airflow to the coals to maintain the temperature setpoint. Now, knowing what has to be sensed, the parameters can be outlined. For the food temperature sensor, it must be composed of a non-corrosive material in order to penetrate into the food without damaging itself of contaminating the food. It also must be able to sense temperatures up to approximately 200 degrees Fahrenheit and survive in environments up to 400 degrees Fahrenheit. For the grill temperature sensor, it must be able to survive in and read temperatures up to 400 degrees Fahrenheit. For sensing the grill and food temperature setpoints, a physical interaction with visual feedback must occur for a person to set each temperature.

4.1 Food and Grill Temperature Sensors

For sensing food temperatures, there were three serious options:

Measurand	Sensor Family	Sensor
Food Temperature	Resistive	RTD
	Resistive	Thermistor
	Self-Generating	Thermocouple

At this point, the thermistor is the least viable of the three sensors due to the ease of manufacturing a RTD or thermocouple into a long probe for penetrating into foods. That combined with the increased durability and range of temperature sensing in RTDs and Thermocouples eliminated thermistors for food and grill temp sensing. So, the options become:

Measurand	Sensor Family	Sensor
Food Temperature	Resistive	RTD
	Self-Generating	Thermocouple

At this point, the interchangeability and cheapness of thermocouples if one was to one day fail whereas a RTD is more expensive and less interchangeable became the deciding factor in choosing thermocouples of RTD’s. Withing thermocouples are several different types. For this project, K type include the measurand range and is the most widely available, so it was chosen.

For sensing the grill temperature, the Adafruit Accessories Thermocouple Type-K (See image 2) was chosen because it can work up to 900 degrees Fahrenheit and survive in the environment of a smoker. For sensing the food temperature, a Prosense Thermocouple with a 6” probe was chosen to because the material is stainless steel and safe up to 400 degrees Fahrenheit, which is above smoking temperatures.

Image 2: Adafruit Accessories Thermocouple Type-K Image 3: Prosense Type K Thermocouple 6” Probe

4.2 Grill and Food Temperature Setpoint Sensing

Keeping in mind that the sensing method must show a visual indicator of the setpoint, the options for sensing become more limited. The viable options are:

Measurand	Sensor Family	Sensor
Temperature Setpoints	Resistive	Potentiometer
	Capacitive	Thermistor
	Digital	Buttons + LCD Screen

The capacitor was cut from the option list because it required the most from a sensor conditioning standpoint to transform the change in capacitance into a change in frequency and then into a digital signal whereas a potentiometer is simple to linearize and condition. The options then become:

Measurand	Sensor Family	Sensor
Temperature Setpoints	Resistive	Potentiometer
	Digital	LCD Shield + Buttons

While the potentiometer would work, it was eliminated because it was determined that an LCD screen would allow for more user interaction and confidence in the right value being set for each temperature. It would also allow for more interactive coding where real time temperatures could be displayed, so an LCD shield with a keypad was chosen. The DFRobot 1602 LCD with Keypad shown below meets that criterion and can easily stack on top of an Arduino Uno or Mega.

Image 4: DFRobot Gravity: 1602 LCD Keypad Shield for Arduino

4.3 Air Flow Control

Here there were really only two options (see table blow) which can be summed up as retaining the original valve style of natural air flow control, but controlling how open and shut the valve is, or making valves always narrow and forcing air through when needed.

Factor Controlled	Method	Actuator
Air Flow	Controlling Natural Air Convection	Electronic Air Intake and Exhaust Valves
	Forced Air	Air Intake Fan

Ultimately, the deciding factor was that controlling natural air convection only works when there is a steady enough fire to have good convection. If the fire begins to die out, control is reduced. The forced air method does not encounter its problem. It relies on fixed, narrow valves and a fan forcing air through as needed for the coals to burn. Ultimately, a NMB Technologies DC Centrifugal Blower was chosen because of its 10.5 CFM air flow capacity and 70,000 Hour life. Temperature was not a consideration as it will be mounted external to the grill and have cool air flowing through it.

4.4 Conclusion and Final Sensor/Actuator List

Parameter	**Importance**	**Final Method**	**Justification**
Food Temperature	To measure doneness	Type K Thermocouple	• Cheap • Interchangeable • Probe Shape
Grill Temperature	To know if the grill needs to increase or decrease temperature	Type K Thermocouple	• Cheap • Interchangeable • Probe Shape
Temperature Setpoints	To know what temperature the grill should maintain	LCD Screen with Keypad	• No signal conditioning needed • Visual Feedback
Air Flow Control	To control the rate at which coals burn which controls temperature	Forced Air	• Provides more control at low temperatures • Narrow valves stop natural convection at high temps

5 Signal Conditioning

5.1 Thermocouple Signal Conditioning

Thermocouples are self-generating sensors that produce a temperature dependent voltage. Each thermocouple requires 4 things in signal conditioning: filtering, linearization, cold junction compensation, and analog to digital conversion. In order to simplify the circuit, and increase accuracy and dependability, especially with human food involved, a thermocouple amplifier was chosen to accomplish this. Therefore, each thermocouple is wired to a MAX31856 Thermocouple Amplifier.

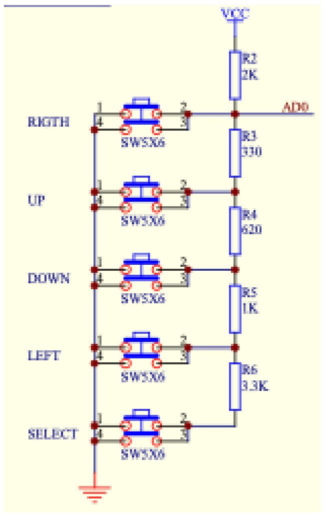
The MAX31856 is powered by 5V and starts out by converting the thermocouple voltage to a digital signal by first amplifying the signal. The MAX31856 accepts ±78.125mV input and amplifies that signal to 0-5V for its analog to digital converter. The amplified signal is then run through notch filtering at 50Hz and 60Hz and their harmonics to eliminate noise from other electronic sources. Last, the ADC converts the analog signal into a 19-bit digital value for a high resolution despite a high range, specifically 0.005 degrees Fahrenheit for a K-type thermocouple.

Most unique to thermocouples, is the need for cold junction compensation. Each junction between metals produces a voltage, so thermocouples produce a combined output voltage of the junction at the measurand end, and the circuit end of a thermocouple. To calculate the measurand end's voltage, cold-junction compensation must be performed to remove the circuit end of the thermocouple's voltage contribution. The MAX31856 has an internal semiconductor (transistor) based temperature sensor that has an accuracy of ±1.26 degrees Fahrenheit which then goes through the same ADC conversion process as the thermocouple. Both the previous overall 19-bit digital value and the cold-junction 19-bit digital value are then used in a look up table (LUT) programmed into the MAX31856 to compensate for the cold junction as well as nonlinearities. This is possible because all type K thermocouples produce the same response to temperature.

Finally, the MAX31856 transmits this final temperature reading over a four-pin serial peripheral interface (SPI): serial data out, serial data in, chip select, and serial clock. These pin functions can be programmed into any digital input/output on the Arduino for communication.

5.2 Keypad Buttons on LCD Shield

The buttons on the LCD shield are powered automatically by the LCD Hat being plugged into the Arduino and are wired as a 5-step voltage divider (see schematic below) to Analog Pin 0



Button	Analog Value
Right	0
Up	204
Down	406
Left	609
Select	824

Image 5: Button Schematic and Analog Values

5.3 DC Centrifugal Blower

The Centrifugal blower required no signal conditioning other than a motor hat driver to accommodate its .29A nominal current. This was accomplished by placing an Adafruit Motor Shield v2.3 between the Arduino and LCD Shield with the blower wired to the M1 output.

6 Logic Implementation

6.1 Hardware Selection

The decision was made to use a microcontroller over logic chips for two main reasons: the need for a user interface, and for programmability. With logic chips and circuit-based logic in general, there is little room for flexibility in testing and tuning the logic to test results. Knowing these things would be valuable, alongside the heavy need for code, proved the need for a programmable microcontroller. Arduino was chosen for the microcontroller over Raspberry Pi for price, ease of setup, and ease of programming. The specific Arduino chosen was the Arduino Mega 2560 for the processing and logic implementation over an Arduino Uno because 14 digital pins were needed, and the Arduino Uno only offers 13 digital pins.

6.2 Code Implementation

As shown in the flowchart below, the code is structured to work through three main stages:

1. Determine the Food and Grill Setpoints
2. Reading and Displaying Actual Food and Grill Temperatures
3. Running the fan to maintain the grill temperature setpoint

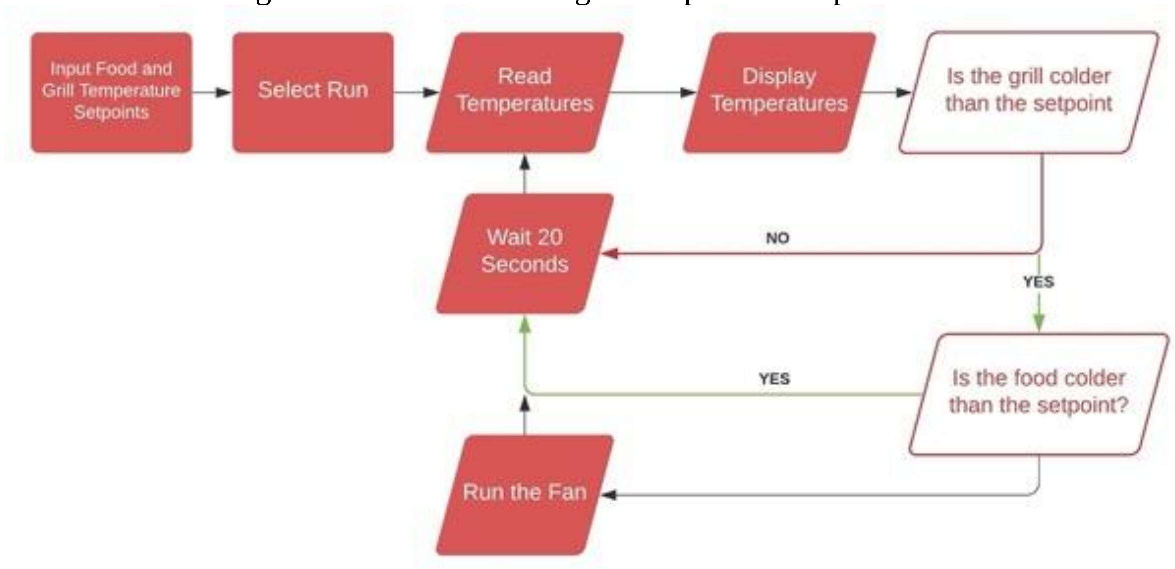
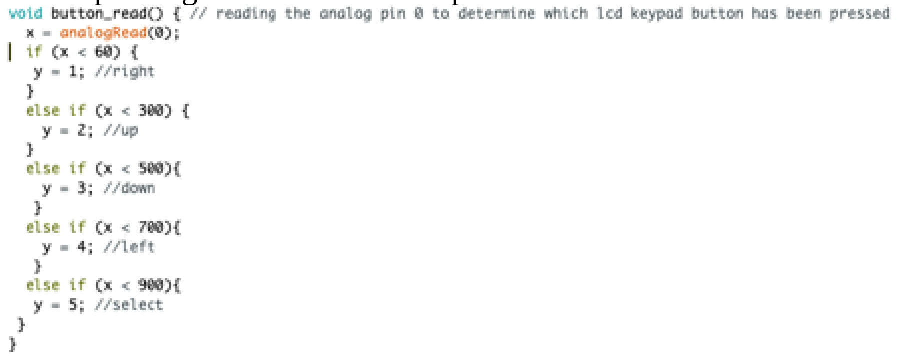

Input Food and Grill Temperature Setpoints
Select Run
Read Temperatures
Display Temperatures
Is the grill colder than the setpoint
NO
Wait 20 Seconds
YES
YES
Is the food colder than the setpoint?
Run the Fan

Image 6: Overall Code Structure Flowchart

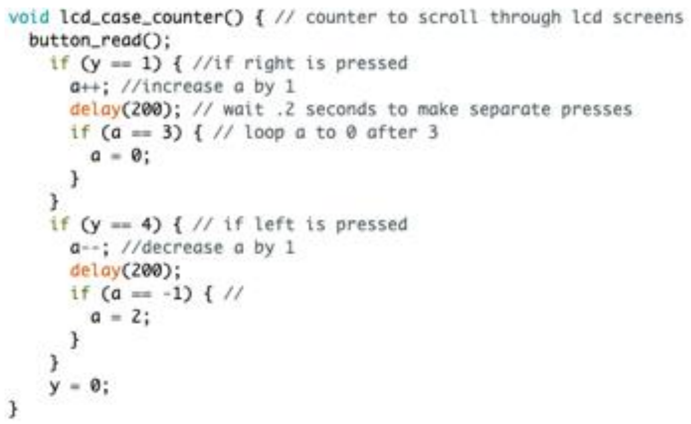
This is all done via functions inside of an LCD Menu. The LCD Menu operates by initially cycling through three main screens by pressing the right button. Buttons are read to set variable "y" to one of 5 values depending on which button is pressed in the void function "button_read() seen below.



```
void button_read() { // reading the analog pin 0 to determine which lcd keypad button has been pressed
  x = analogRead(0);
  if (x < 60) {
   y = 1; //right
  }
  else if (x < 300) {
    y = 2; //up
  }
  else if (x < 500){
    y = 3; //down
   }
  else if (x < 700){
    y = 4; //left
   }
  else if (x < 900){
   y = 5; //select
 }
}
```

Image 7: Button Read function

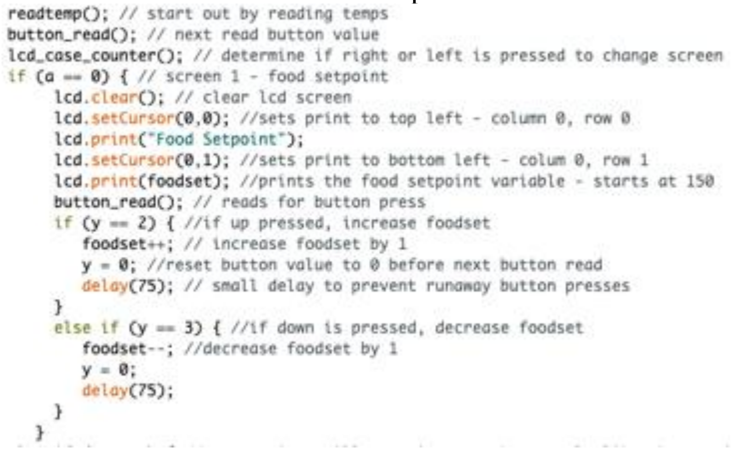
The correct screen is chosen to be displayed by the void function "lcd_case_counter()" (see below).



```
void lcd_case_counter() { // counter to scroll through lcd screens
  button_read();
    if (y == 1) { //if right is pressed
      a++; //increase a by 1
      delay(200); // wait .2 seconds to make separate presses
      if (a == 3) { // loop a to 0 after 3
        a = 0;
      }
    }
    if (y == 4) { // if left is pressed
      a--; //decrease a by 1
      delay(200);
      if (a == -1) { //
        a = 2;
      }
    }
    y = 0;
}
```

Image 8: LCD Screen Case Counter

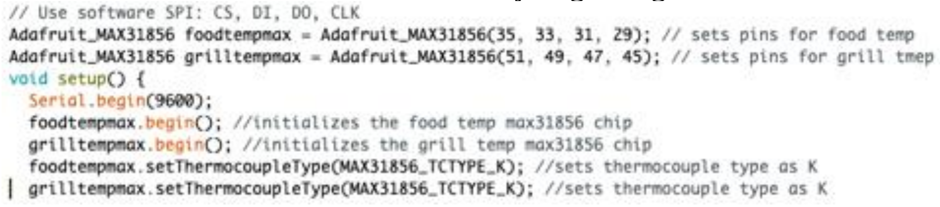
Then, in the void loop function, a series of "if a is equal to…" are used to determine the correct screen to be displayed. Screen 1 Displays "Food Setpoint" on first line and the food setpoint variable on the second line, which can be changed by pressing the up or down buttons on the keypad. Here is the code for Screen 1 which is inside the void loop function:



```
readtemp(); // start out by reading temps
button_read(); // next read button value
lcd_case_counter(); // determine if right or left is pressed to change screen
if (a == 0) { // screen 1 - food setpoint
    lcd.clear(); // clear lcd screen
    lcd.setCursor(0,0); //sets print to top left - column 0, row 0
    lcd.print("Food Setpoint");
    lcd.setCursor(0,1); //sets print to bottom left - colum 0, row 1
    lcd.print(foodset); //prints the food setpoint variable - starts at 150
    button_read(); // reads for button press
    if (y == 2) { //if up pressed, increase foodset
      foodset++; // increase foodset by 1
      y = 0; //reset button value to 0 before next button read
      delay(75); // small delay to prevent runaway button presses
    }
    else if (y == 3) { //if down is pressed, decrease foodset
      foodset--; //decrease foodset by 1
      y = 0;
      delay(75);
    }
  }
```

Image 9: Code for Screen 1

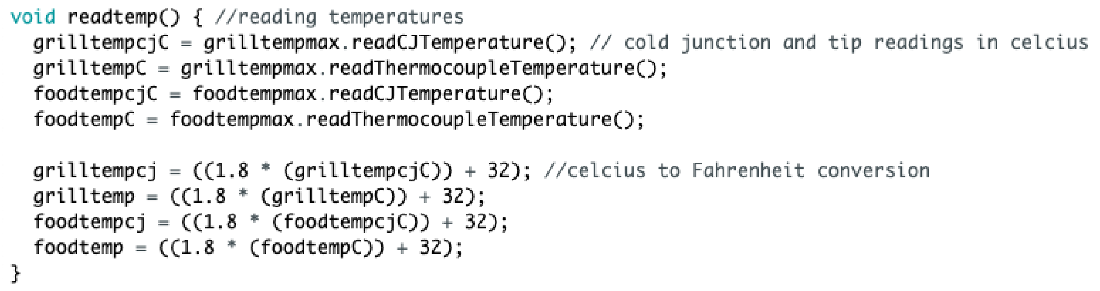
Next comes the code for Screen 2, the "Grill Setpoint" screen. Its code is the exact same as Screen 1's code except it prints "Grill Setpoint" and the "grillset" variable. Screen 3 asks if the user would like to begin running the controller. If select is pressed, the controller will loop indefinitely in a cycle where it reads the food and grill temperature, displays the food and grill temperature, determines the setting for if the fan needs to be ran and for how long, and then runs the correct fan setting. Starting with reading temperatures, the code communicates with the MAX31856 chips using SPI communication which is initialized at the very beginning of the code with the following:



```
// Use software SPI: CS, DI, DO, CLK
Adafruit_MAX31856 foodtempmax = Adafruit_MAX31856(35, 33, 31, 29); // sets pins for food temp
Adafruit_MAX31856 grilltempmax = Adafruit_MAX31856(51, 49, 47, 45); // sets pins for grill tmep
void setup() {
  Serial.begin(9600);
  foodtempmax.begin(); //initializes the food temp max31856 chip
  grilltempmax.begin(); //initializes the grill temp max31856 chip
  foodtempmax.setThermocoupleType(MAX31856_TCTYPE_K); //sets thermocouple type as K
| grilltempmax.setThermocoupleType(MAX31856_TCTYPE_K); //sets thermocouple type as K
```

Image 10: MAX31856 Initialization Code

Then the void function "readtemp()" shown below is used to read the temperatures, and then convert them to Fahrenheit

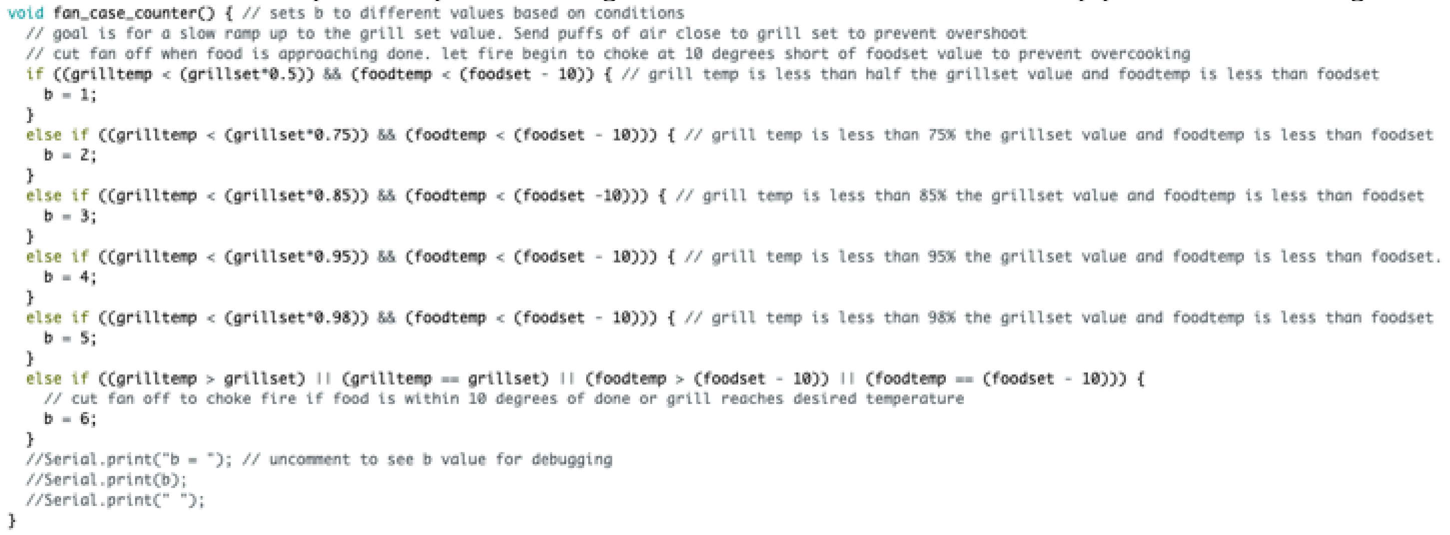


```
void readtemp() { //reading temperatures
  grilltempcjC = grilltempmax.readCJTemperature(); // cold junction and tip readings in celcius
  grilltempC = grilltempmax.readThermocoupleTemperature();
  foodtempcjC = foodtempmax.readCJTemperature();
  foodtempC = foodtempmax.readThermocoupleTemperature();

  grilltempcj = ((1.8 * (grilltempcjC)) + 32); //celcius to Fahrenheit conversion
  grilltemp = ((1.8 * (grilltempC)) + 32);
  foodtempcj = ((1.8 * (foodtempcjC)) + 32);
  foodtemp = ((1.8 * (foodtempC)) + 32);
}
```

Image 11: Temperature Reading Function

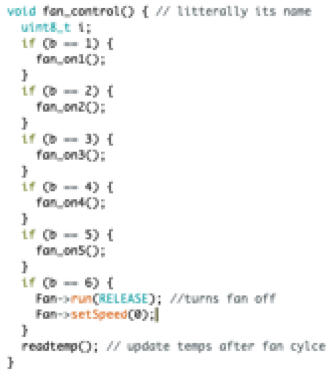
After that, the code displays the temperatures in a similar manner as before. Then, it runs the void function "fan_case_counter()" (see below). The point of choosing a variety of cases is that constantly blowing air onto the coals will create an inferno, so a gradual approach is needed. As temperatures get closer to the setpoint, the fan should stoke the fire less and less. Also, if the food reaches its setpoint temperature, the grill should not be stoked at all to help prevent overcooking.



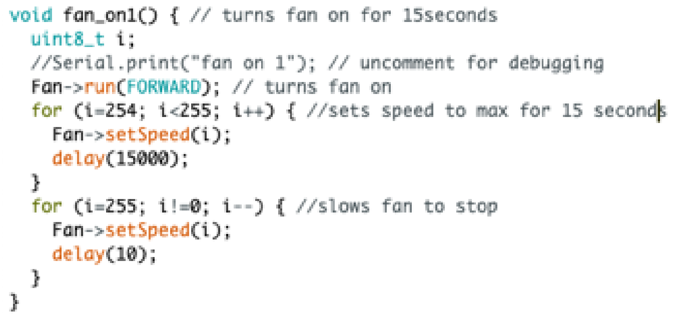
```
void fan_case_counter() { // sets b to different values based on conditions
  // goal is for a slow ramp up to the grill set value. Send puffs of air close to grill set to prevent overshoot
  // cut fan off when food is approaching done. let fire begin to choke at 10 degrees short of foodset value to prevent overcooking
  if ((grilltemp < (grillset*0.5)) && (foodtemp < (foodset - 10)) { // grill temp is less than half the grillset value and foodtemp is less than foodset
    b = 1;
  }
  else if ((grilltemp < (grillset*0.75)) && (foodtemp < (foodset - 10))) { // grill temp is less than 75% the grillset value and foodtemp is less than foodset
    b = 2;
  }
  else if ((grilltemp < (grillset*0.85)) && (foodtemp < (foodset -10))) { // grill temp is less than 85% the grillset value and foodtemp is less than foodset
    b = 3;
  }
  else if ((grilltemp < (grillset*0.95)) && (foodtemp < (foodset - 10))) { // grill temp is less than 95% the grillset value and foodtemp is less than foodset.
    b = 4;
  }
  else if ((grilltemp < (grillset*0.98)) && (foodtemp < (foodset - 10))) { // grill temp is less than 98% the grillset value and foodtemp is less than foodset
    b = 5;
  }
  else if ((grilltemp > grillset) || (grilltemp == grillset) || (foodtemp > (foodset - 10)) || (foodtemp == (foodset - 10))) {
    // cut fan off to choke fire if food is within 10 degrees of done or grill reaches desired temperature
    b = 6;
  }
  //Serial.print("b = "); // uncomment to see b value for debugging
  //Serial.print(b);
  //Serial.print(" ");
}
```

Image 11: Fan Case Counter

Next, the void function "fan_control()" is ran to actually turn on the fan for one of 6 durations: 15 seconds, 7.5 seconds, 5 seconds, 2.5 seconds, 1 second, 0 seconds. The duration is chosen by the b value set in "fan_case_counter()". Here is the code that does that on the left:



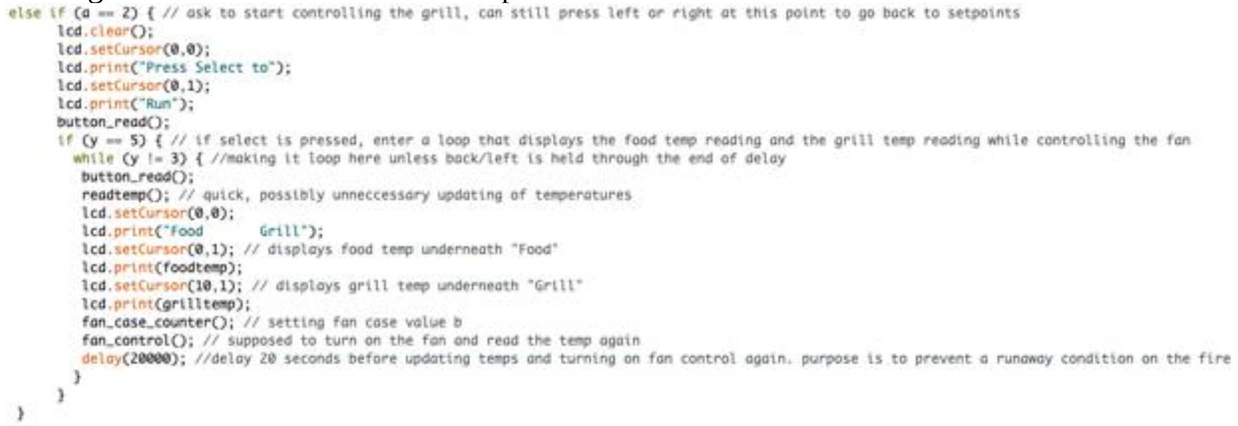
```
void fan_control() { // litterally its name
  uint8_t i;
  if (b == 1) {
    fan_on1();
  }
  if (b == 2) {
    fan_on2();
  }
  if (b == 3) {
    fan_on3();
  }
  if (b == 4) {
    fan_on4();
  }
  if (b == 5) {
    fan_on5();
  }
  if (b == 6) {
    Fan->run(RELEASE); //turns fan off
    Fan->setSpeed(0);
  }
  readtemp(); // update temps after fan cylce
}
```



```
void fan_on1() { // turns fan on for 15seconds
  uint8_t i;
  //Serial.print("fan on 1"); // uncomment for debugging
  Fan->run(FORWARD); // turns fan on
  for (i=254; i<255; i++) { //sets speed to max for 15 seconds
    Fan->setSpeed(i);
    delay(15000);
  }
  for (i=255; i!=0; i--) { //slows fan to stop
    Fan->setSpeed(i);
    delay(10);
  }
}
```

Image 12: Fan Control Function (LEFT) and Example Fan ON Function (RIGHT)

The functions "fan_on1(), fan_on2(), fan_on3(), fan_on4(), and fan_on5()" are written exactly like the "fan_on1()" function (See Image 12), except with different delays to change the run time between the times listed above, in that order. Once all this is added together, you get the full screen 3 and grill control code in the void loop function as shown here:



```
else if (a == 2) { // ask to start controlling the grill, can still press left or right at this point to go back to setpoints
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Press Select to");
      lcd.setCursor(0,1);
      lcd.print("Run");
      button_read();
      if (y == 5) { // if select is pressed, enter a loop that displays the food temp reading and the grill temp reading while controlling the fan
        while (y != 3) { //making it loop here unless back/left is held through the end of delay
         button_read();
         readtemp(); // quick, possibly unneccessary updating of temperatures
         lcd.setCursor(0,0);
         lcd.print("Food      Grill");
         lcd.setCursor(0,1); // displays food temp underneath "Food"
         lcd.print(foodtemp);
         lcd.setCursor(10,1); // displays grill temp underneath "Grill"
         lcd.print(grilltemp);
         fan_case_counter(); // setting fan case value b
         fan_control(); // supposed to turn on the fan and read the temp again
         delay(20000); //delay 20 seconds before updating temps and turning on fan control again. purpose is to prevent a runaway condition on the fire
        }
      }
  }
```

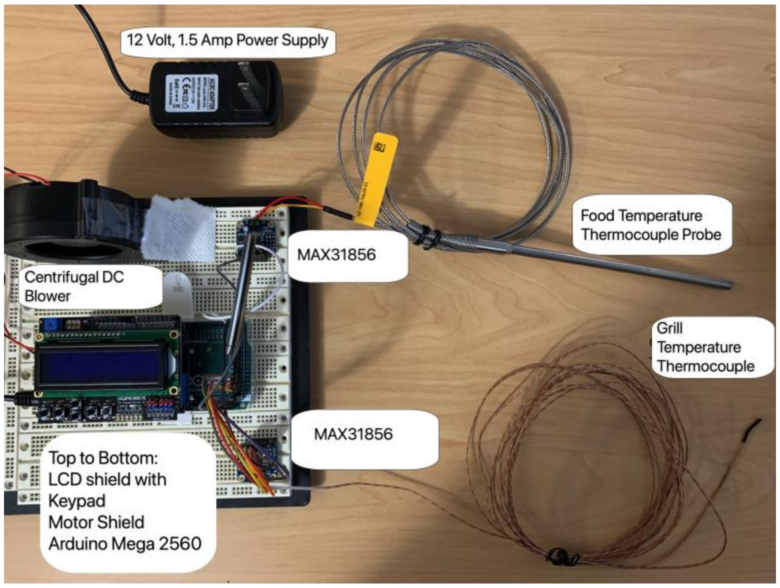
For the full code, please see the GitHub link in the appendix.

7 Physical Considerations

There are two main physical considerations for this project that require looking into. The first is the obvious, temperature. Grills get hot. Fortunately, this controller is used for long term smoking, which happens at temperatures less than 350 degrees Fahrenheit, so both sensors must be able to exceed that upper limit, and they do. The other consideration that had to be kept in mind was the food ruining the food probe sensor by corroding it or the food probe sensor contaminating the food. For this reason, a stainless-steel probe was chosen to prevent these issues. For more information on the physical parameters everything had to meet, see the checklists in the appendix.

8 Results

8.1 Final Product


12 Volt, 1.5 Amp Power Supply
Food Temperature Thermocouple Probe
MAX31856
Centrifugal DC Blower
Grill Temperature Thermocouple
MAX31856
Top to Bottom: LCD shield with Keypad Motor Shield Arduino Mega 2560

8.2 Final Testing Results

For testing the temperature readings, the temperature had to be within 5 degrees Fahrenheit and run the fan for the correct amount of time. An oven was used to heat the thermocouples and a store bought digital thermometer was used to confirm the readings.

Fan Case Tested	Grill Setpoint	Food Setpoint	Grill Thermocouple Reading	Food Thermocouple Reading	Grill Actual	Food Actual	Fan Run as expected?
1	300	200	120	68	121	68	Yes
2	300	200	216	68	214	68	Yes
3	300	200	243	68	241	68	Yes
4	300	200	270	68	273	68	Yes
5	300	200	288	68	292	68	Yes
6	300	200	307	68	305	68	Yes
6	300	200	68	321	69	325	Yes

After testing and getting results, it is shown that the thermocouple readings are accurate, the LCD displays them correctly, the menu selection works, and the correct fan control function is selected and ran in the seven main conditions than can occur. Therefore, the code works exactly as expected.

9 Revision 2

For a final revision, two things need to occur. The first one is replacing the Arduino with a multicore processor. The single core Arduino is only capable of doing one task at a time, and while that is sufficient for this project, it is not ideal either. Having multiple cores would allow for multiple things the run at once. That means the temperatures on the LCD screen could update while the fan control function is running, making the operation and code run more smoothly. The second thing is the overall construction of the controller. It cannot survive long term on a breadboard, so a second revision would see it transition from a breadboard to a solderable perfboard or PCB that a case could be built around.

10 References

The only resource used in completing this project was Adafruit's MAX31856 Wiring and Test guide (see Appendix for link). The Wiring and Test guide goes through and explains how to communicate with the MAX31856 chip to get readings, and it was used to include the correct libraries in the code and to talk to the chip. The entire structure of how the code used the MAX31856 readings to control the fan was completely original, as well as the user interface, and all other relevant coding knowledge on the project came from previous projects such as the XY Plotter in Design Methodology or was figured out on this project using trial and error.

11 Appendix

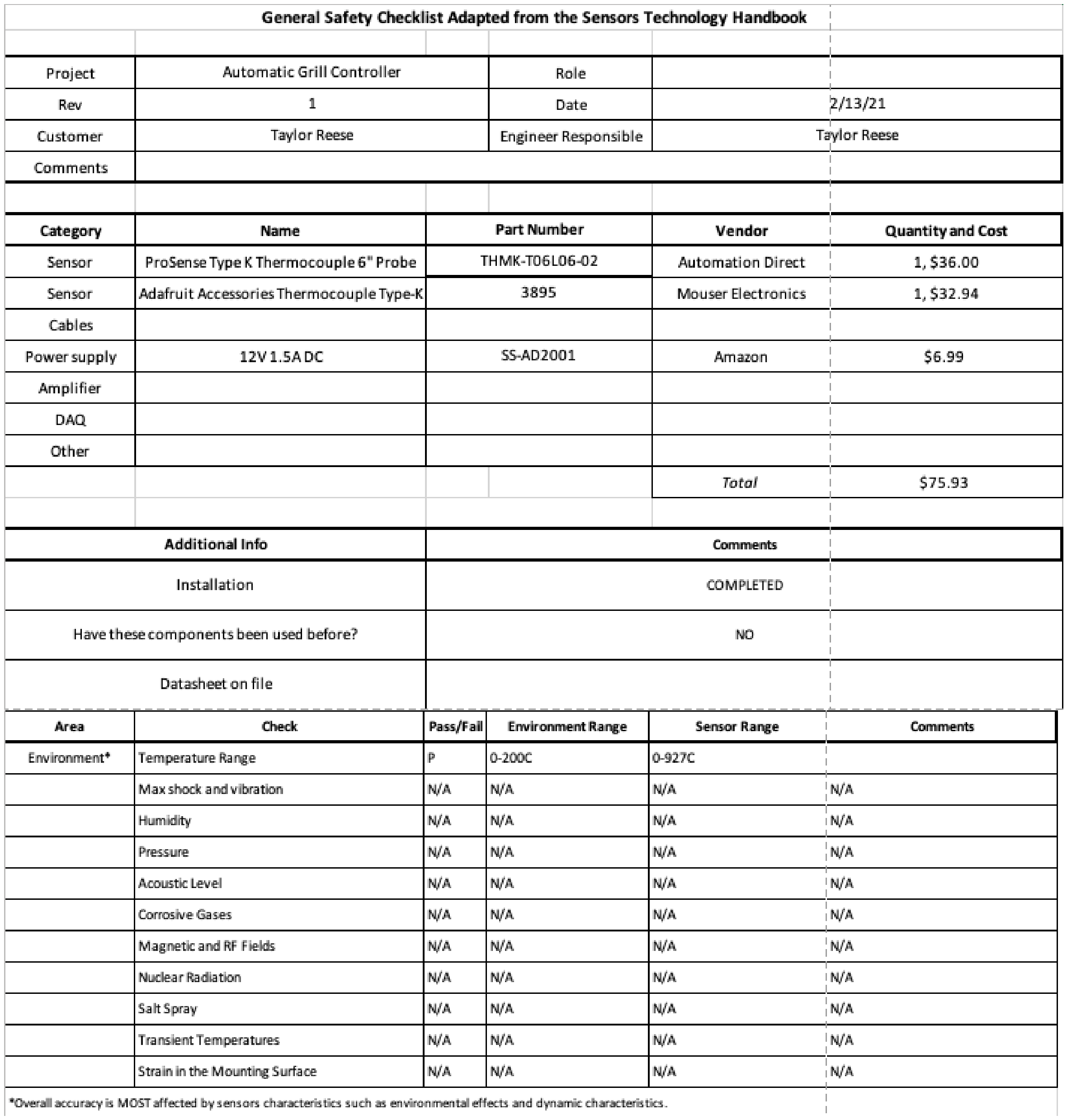
11.1 Project Github Page

https://github.com/tbr60702/Sensors-Final-Project

11.2 Final Code

https://github.com/tbr60702/Sensors-Final-Project/blob/main/Grill_Controller_Final.ino

11.3 Sensors Checklist



General Safety Checklist Adapted from the Sensors Technology Handbook

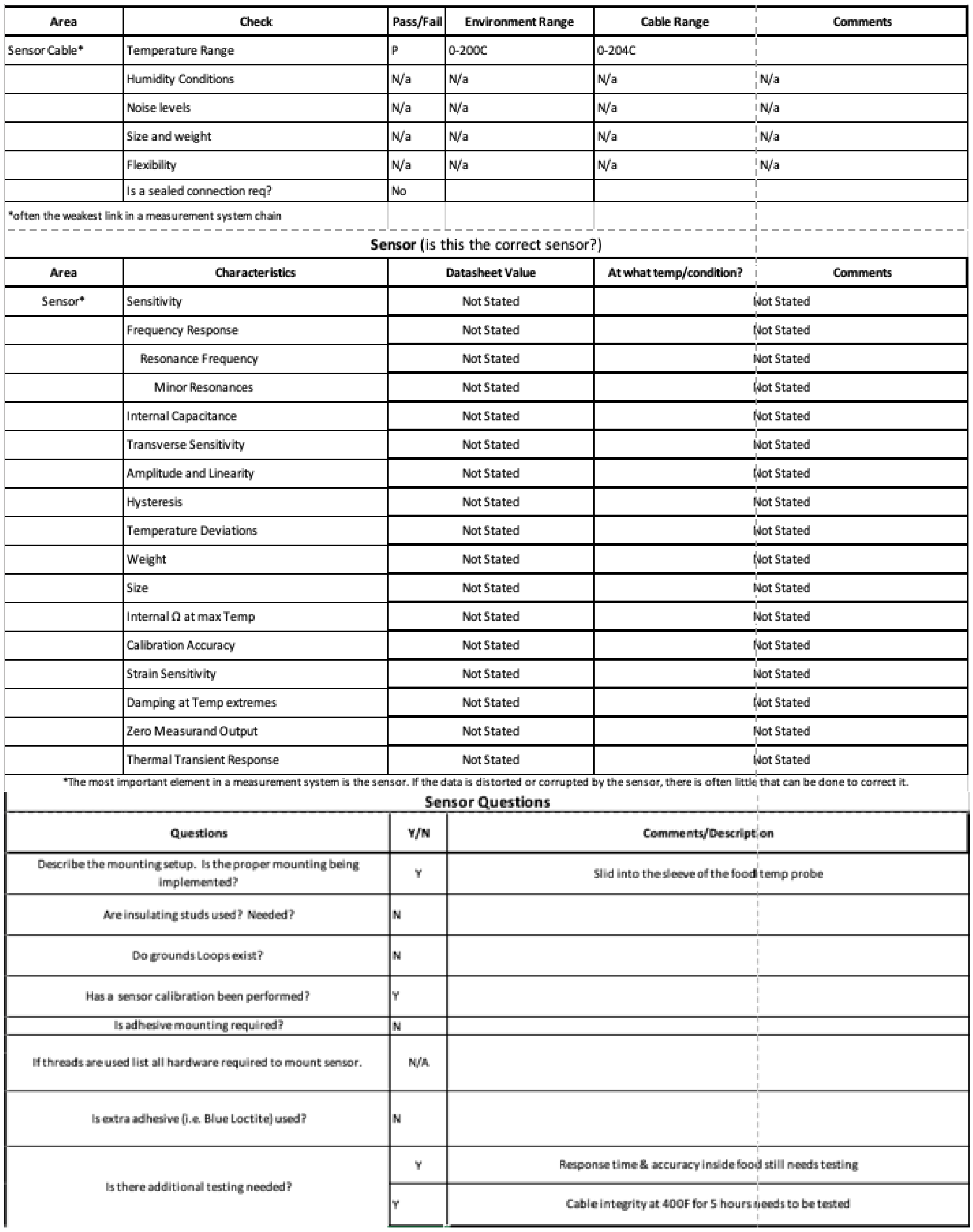
Project	Automatic Grill Controller	Role	
Rev	1	Date	2/13/21
Customer	Taylor Reese	Engineer Responsible	Taylor Reese
Comments			

Category	Name	Part Number	Vendor	Quantity and Cost
Sensor	ProSense Type K Thermocouple 6" Probe	THMK-T06L06-02	Automation Direct	1, $36.00
Sensor	Adafruit Accessories Thermocouple Type-K	3895	Mouser Electronics	1, $32.94
Cables				
Power supply	12V 1.5A DC	SS-AD2001	Amazon	$6.99
Amplifier				
DAQ				
Other				
			Total	$75.93

Additional Info	Comments
Installation	COMPLETED
Have these components been used before?	NO
Datasheet on file	

Area	Check	Pass/Fail	Environment Range	Sensor Range	Comments
Environment*	Temperature Range	P	0-200C	0-927C	
	Max shock and vibration	N/A	N/A	N/A	N/A
	Humidity	N/A	N/A	N/A	N/A
	Pressure	N/A	N/A	N/A	N/A
	Acoustic Level	N/A	N/A	N/A	N/A
	Corrosive Gases	N/A	N/A	N/A	N/A
	Magnetic and RF Fields	N/A	N/A	N/A	N/A
	Nuclear Radiation	N/A	N/A	N/A	N/A
	Salt Spray	N/A	N/A	N/A	N/A
	Transient Temperatures	N/A	N/A	N/A	N/A
	Strain in the Mounting Surface	N/A	N/A	N/A	N/A

*Overall accuracy is MOST affected by sensors characteristics such as environmental effects and dynamic characteristics.



Area	Check	Pass/Fail	Environment Range	Cable Range	Comments
Sensor Cable*	Temperature Range	P	0-200C	0-204C	
	Humidity Conditions	N/a	N/a	N/a	N/a
	Noise levels	N/a	N/a	N/a	N/a
	Size and weight	N/a	N/a	N/a	N/a
	Flexibility	N/a	N/a	N/a	N/a
	Is a sealed connection req?	No			

*often the weakest link in a measurement system chain

Sensor (is this the correct sensor?)

Area	Characteristics	Datasheet Value	At what temp/condition?	Comments
Sensor*	Sensitivity	Not Stated		Not Stated
	Frequency Response	Not Stated		Not Stated
	Resonance Frequency	Not Stated		Not Stated
	Minor Resonances	Not Stated		Not Stated
	Internal Capacitance	Not Stated		Not Stated
	Transverse Sensitivity	Not Stated		Not Stated
	Amplitude and Linearity	Not Stated		Not Stated
	Hysteresis	Not Stated		Not Stated
	Temperature Deviations	Not Stated		Not Stated
	Weight	Not Stated		Not Stated
	Size	Not Stated		Not Stated
	Internal Ω at max Temp	Not Stated		Not Stated
	Calibration Accuracy	Not Stated		Not Stated
	Strain Sensitivity	Not Stated		Not Stated
	Damping at Temp extremes	Not Stated		Not Stated
	Zero Measurand Output	Not Stated		Not Stated
	Thermal Transient Response	Not Stated		Not Stated

*The most important element in a measurement system is the sensor. If the data is distorted or corrupted by the sensor, there is often little that can be done to correct it.

Sensor Questions

Questions	Y/N	Comments/Description
Describe the mounting setup. Is the proper mounting being implemented?	Y	Slid into the sleeve of the food temp probe
Are insulating studs used? Needed?	N	
Do grounds Loops exist?	N	
Has a sensor calibration been performed?	Y	
Is adhesive mounting required?	N	
If threads are used list all hardware required to mount sensor.	N/A	
Is extra adhesive (i.e. Blue Loctite) used?	N	
Is there additional testing needed?	Y	Response time & accuracy inside food still needs testing
	Y	Cable integrity at 400F for 5 hours needs to be tested

Sensor (and the environment) (Adafruit Accessories Type K Thermocouple)

Area	Check	Pass/Fail	Environment Range	Sensor Range	Comments
Environment*	Temperature Range	P	0-200C	0-500C	
	Max shock and vibration	N/A	N/A	N/A	N/A
	Humidity	N/A	N/A	N/A	N/A
	Pressure	N/A	N/A	N/A	N/A
	Acoustic Level	N/A	N/A	N/A	N/A
	Corrosive Gases	N/A	N/A	N/A	N/A
	Magnetic and RF Fields	N/A	N/A	N/A	N/A
	Nuclear Radiation	N/A	N/A	N/A	N/A
	Salt Spray	N/A	N/A	N/A	N/A
	Transient Temperatures	N/A	N/A	N/A	N/A
	Strain in the Mounting Surface	N/A	N/A	N/A	N/A

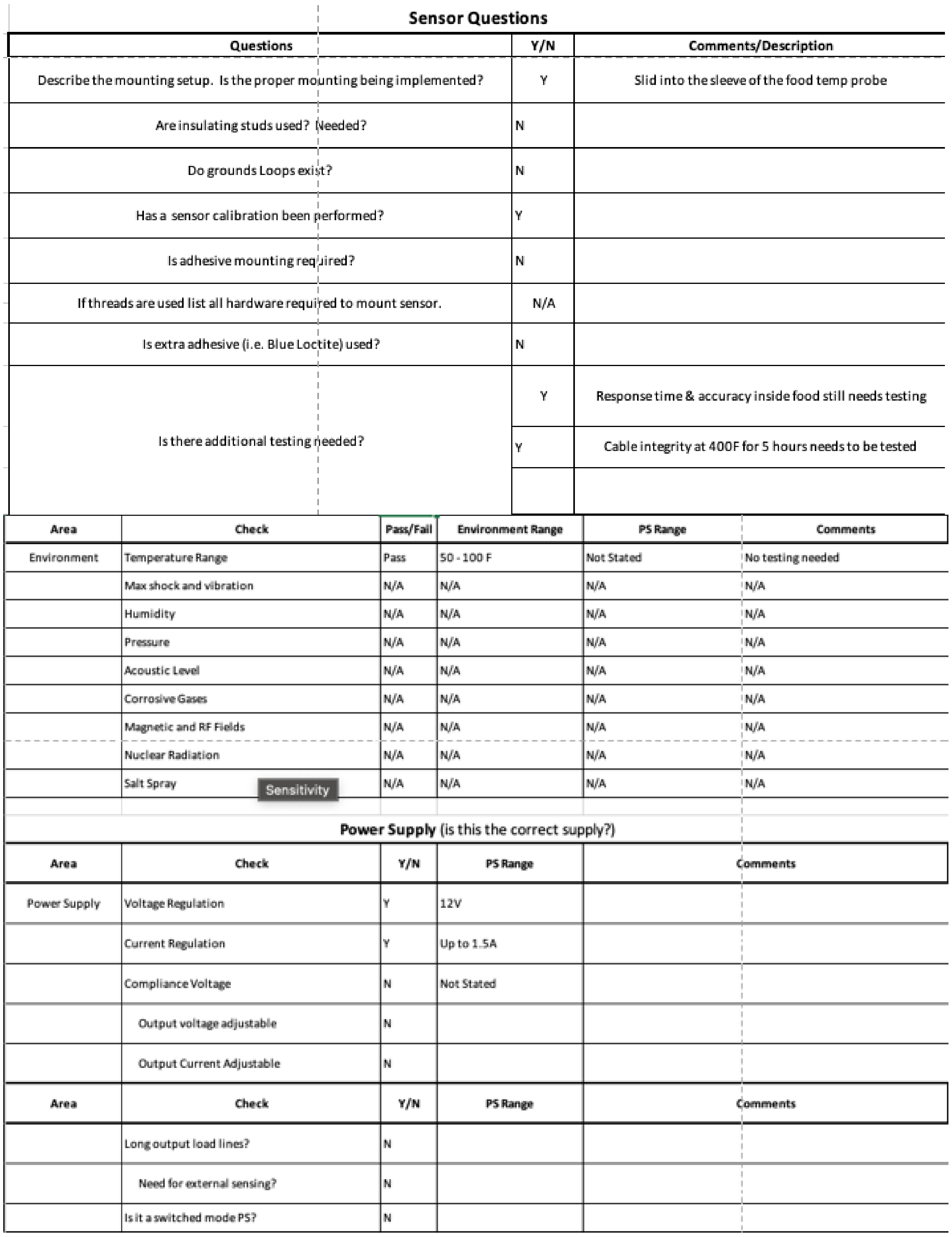
*Overall accuracy is MOST affected by sensors characteristics such as environmental effects and dynamic characteristics.

Area	Check	Pass/Fail	Environment Range	Cable Range	Comments
Sensor Cable*	Temperature Range	P	0-200C	0-250C	
	Humidity Conditions	N/a	N/a	N/a	N/a
	Noise levels	N/a	N/a	N/a	N/a
	Size and weight	N/a	N/a	N/a	N/a
	Flexibility	N/a	N/a	N/a	N/a
	Is a sealed connection req?	No			

*often the weakest link in a measurement system chain

Sensor (is this the correct sensor?)

Area	Characteristics	Datasheet Value	At what temp/condition?	Comments
Sensor*	Sensitivity	Not Stated	Not Stated	
	Frequency Response	Not Stated	Not Stated	
	Resonance Frequency	Not Stated	Not Stated	
	Minor Resonances	Not Stated	Not Stated	
	Internal Capacitance	Not Stated	Not Stated	
	Transverse Sensitivity	Not Stated	Not Stated	
	Amplitude and Linearity	Not Stated	Not Stated	
	Hysteresis	Not Stated	Not Stated	
	Temperature Deviations	Not Stated	Not Stated	
	Weight	Not Stated	Not Stated	
	Size	Not Stated	Not Stated	
	Internal Ω at max Temp	Not Stated	Not Stated	
	Calibration Accuracy	Not Stated	Not Stated	
	Strain Sensitivity	Not Stated	Not Stated	
	Damping at Temp extremes	Not Stated	Not Stated	
	Zero Measurand Output	Not Stated	Not Stated	
	Thermal Transient Response	Not Stated	Not Stated	


Sensitivity

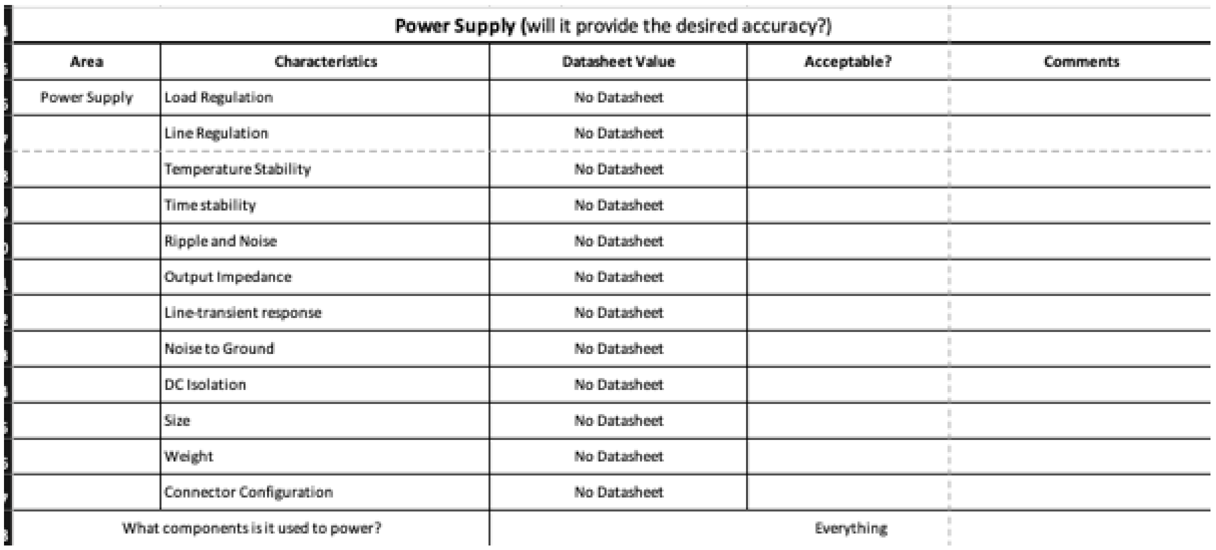
Sensor Questions

Questions	Y/N	Comments/Description
Describe the mounting setup. Is the proper mounting being implemented?	Y	Slid into the sleeve of the food temp probe
Are insulating studs used? Needed?	N	
Do grounds Loops exist?	N	
Has a sensor calibration been performed?	Y	
Is adhesive mounting required?	N	
If threads are used list all hardware required to mount sensor.	N/A	
Is extra adhesive (i.e. Blue Loctite) used?	N	
Is there additional testing needed?	Y	Response time & accuracy inside food still needs testing
	Y	Cable integrity at 400F for 5 hours needs to be tested

Area	Check	Pass/Fail	Environment Range	PS Range	Comments
Environment	Temperature Range	Pass	50 - 100 F	Not Stated	No testing needed
	Max shock and vibration	N/A	N/A	N/A	N/A
	Humidity	N/A	N/A	N/A	N/A
	Pressure	N/A	N/A	N/A	N/A
	Acoustic Level	N/A	N/A	N/A	N/A
	Corrosive Gases	N/A	N/A	N/A	N/A
	Magnetic and RF Fields	N/A	N/A	N/A	N/A
	Nuclear Radiation	N/A	N/A	N/A	N/A
	Salt Spray	N/A	N/A	N/A	N/A

Power Supply (is this the correct supply?)

Area	Check	Y/N	PS Range	Comments
Power Supply	Voltage Regulation	Y	12V	
	Current Regulation	Y	Up to 1.5A	
	Compliance Voltage	N	Not Stated	
	Output voltage adjustable	N		
	Output Current Adjustable	N		
Area	**Check**	**Y/N**	**PS Range**	**Comments**
	Long output load lines?	N		
	Need for external sensing?	N		
	Is it a switched mode PS?	N		



Power Supply (will it provide the desired accuracy?)				
Area	Characteristics	Datasheet Value	Acceptable?	Comments
Power Supply	Load Regulation	No Datasheet		
	Line Regulation	No Datasheet		
	Temperature Stability	No Datasheet		
	Time stability	No Datasheet		
	Ripple and Noise	No Datasheet		
	Output Impedance	No Datasheet		
	Line-transient response	No Datasheet		
	Noise to Ground	No Datasheet		
	DC Isolation	No Datasheet		
	Size	No Datasheet		
	Weight	No Datasheet		
	Connector Configuration	No Datasheet		
What components is it used to power?			Everything	

11.4 MAX31856 Wiring and Test Guide

https://learn.adafruit.com/adafruit-max31856-thermocouple-amplifier/wiring-and-test