The Thermo**meat**er

Michael Sandrin (217144692)

***Abstract*—The Thermomeater is an Arduino Due based embedded system that helps the user cook their food. The Thermomeater contains two main features to assist the user, these being a Timer to help the user when cooking food based on time (frozen or precooked foods) and Temperature tracker that tells the user when the required temperature has been reached. The device uses accurate methods of data receiving and transmission for all of the devices features, these being *digital*, *analog*, *serial*. Digital being used for the audio signal (speaker) and buttons, analog for the temperature sensor/probe, and serial for the bluetooth receiver.**

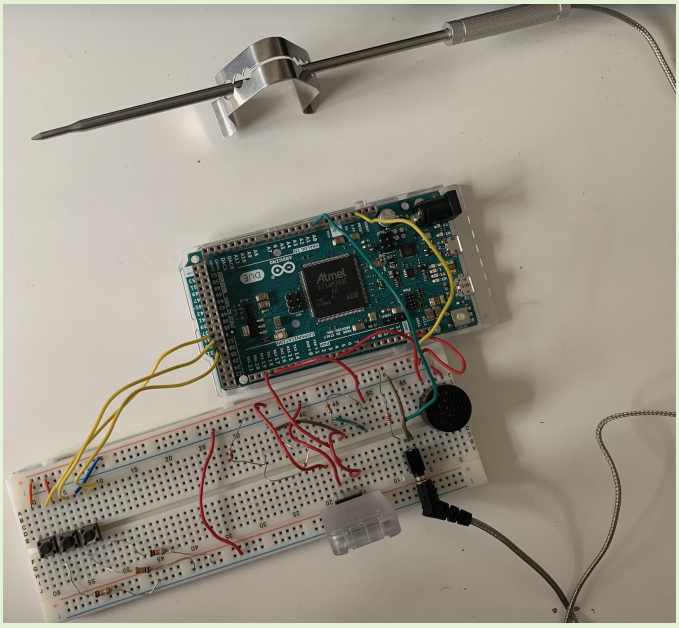

*Figure 1, Photo of the Thermo**meat**er Circuit*

The Thermomeater measures temperature in the standard that the Canadian Government uses for food safety, which is in Fahrenheit and lets the user set their own required temperature, so it can be used for any type of food the user wants to cook. The device has a 3 button array that is used for two specific purposes in the User Interface, either to choose between 3 options or increment a specific value. The bluetooth sensor allows any smart device to connect to the Thermomeater to display the users options and progress of the food or timer. The device also contains a speaker to notify the user from a distance using sounds from a select method of 13 specific sounds.

The final product ended up becoming a working device that is a very useful device in the everyday household and kitchen. The device accurately measures temperature while having an bug free user interface and software, allowing the user to have the best experience possible while getting the most effective use of the device as well. The Thermomeater has become the device it was intended to be and made all of my expectations in the end.

***Index Terms*— Keywords (Alphabetical Order):**

Page 2-3: Button Maps
Page 4: Flowchart
Page 2: Functions
Page 5: Hardware
Page 2: Menus
Page 4: Software
Page 4: Sounds
Page 3: Temperature Conversion
Page 3: Temperature Reading
Page 3: Timer

[1] Introduction

THE thermo**meat**er is a device that is your next assistant in the kitchen, keeping track of all the food you cook, all while ensuring that food safety is a priority for the user. With the use of bluetooth connection, the user can connect their smart devices (IOS or Android) via the DSD Tech App available on both platforms. With the app, the status of the device, food and users options are displayed on the visual interface given by the app.

Being based on the Arduino DUE microcontroller, the device can be assembled easily and made in a cost effective manner while retaining quality. With the standard Mono/TS jack used for most thermometer probes, if the user provides their own probe from a pre-existing device or as a replacement, the user can easily unplug and interchange the probes and have the system work the same.

The Thermo**meat**er can run on two sources of power, either connected using a micro-USB cable or using a 9V Battery, allowing for portability with a battery and a backup power option in case the user does not have a battery. The Thermo**meat**er was made with the user in mind, allowing for easy interchangeability of components, simplicity in the interface and simple integration with any of the users smart devices.

This makes the Thermo**meat**er a device that is accessible to most people due to its simplicity and uses in everyday life while preparing food, resulting in the device being aimed for people who cook often, need assistance while cooking, or someone new to cooking. The only challenge that can be faced with this device is when the thermometer probe enters food, it takes a long time to cool down (approximately 3-5 mins, depending on the temperature it reaches).

I. Proposed Project

A. Program

Functions:

The software found in the Thermo**meat**er system uses functions to complete many important but small tasks, create efficiency in the program and help sort the software to avoid confusion. The following functions are found in the software:

- mainMenu()
- foodMenu()
- incremButtonMap()
- timerButtonMap()
- turningOff()
- minsToMillisec()
- millisecToMins()
- foodTimer()
- tempConversion()
- liveTempReading()
- sounds()

mainMenu():

Prints out the main menu for the user to see on their smart device. With the menu, the user can see the three options at the start of the program or after the program restarts. The main menu is shown in *Figure 2*:

```
[13:33:12]->    Main Menu
[13:33:12]-> -----------------
[13:33:12]-> | 1. Timer
[13:33:12]-> | 2. Food & Meat
[13:33:13]-> | 3. Turn Off
[13:33:13]-> -----------------
```

Figure 2, Main Menu on Device

foodMenu():

Prints out the food menu for the user to see on their smart device. With the menu, the user can see the three options after the user chooses the "Food & Meat" option from the main menu. The food menu is shown in *Figure 3*:

```
[13:36:04]->   Preset Food
[13:36:04]-> -----------------
[13:36:05]-> | 1. Chicken
[13:36:05]-> | 2. Beef
[13:36:06]-> | 3. Other Foods
[13:36:06]-> -----------------
```

Figure 3, Food Menu on Device

incremButtonMap():

Prints out a temperature increment button map for the user to see on their smart device. With the menu, the user can see how to increase and decrease the required temperature by 5 degrees fahrenheit using the **three button array**. This is shown after choosing the "Other Foods" option from the food menu. The button map is shown in Figure 4:

```
[13:37:13]->  Temp Increment
[13:37:14]-> [LEFT|CENTER|RIGHT]
[13:37:14]-> [ -5 |DONE| +5 ]
[13:37:14]->-----------------
```

Figure 4, Temperature Increment Button Map on Device

[1]

timerButtonMap():

Prints out a timer increment button map for the user to see on their smart device. With the menu, the user can see how to increase and decrease the duration of the timer by 1 minute using the **three button array**. This is shown after choosing the "Timer" option from the main menu. The button map is shown in Figure 5:

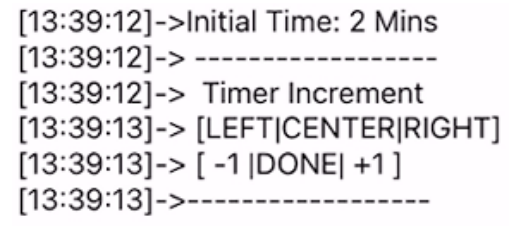

Figure 5, Timer Increment Button Map on Device

turningOff():

Prints out text that tells the user the system is turning off. With the use of a for loop, the text before it is spaced out enough to where it appears off the screen and then plays a shutdown sound from the *sound function* and displays my name and the year the software was complete, "Michael Sandrin 2021". This is shown later in the report in *Figure 12*.

minsToMillisec():

Takes an integer value for minutes and converts the minutes into milliseconds and then returns the value of milliseconds to where the function was called.

millisecToMins():

Takes the integer value for milliseconds and converts the milliseconds into minutes but as a double and then returns the value of minutes to where the function was called. It is a double data type because if the initial minute value is odd, the value halfway through it would have to be 0.50 at the end of the value, meaning it cannot be an integer to print such value.

foodTimer():

Takes the integer value of milliseconds and acts as the whole timer selected in the main menu by choosing the "Timer" option. The function calculates when the timer reaches halfway and when there is one minute left, and saves those times and adds them to the delays in respect to when they should be placed. In addition, the sounds from the *sounds function* were also played with text being presented to the user when the timer started, reached halfway, has one minute remaining, and lastly when finished. The Timer function can be seen on a device in Figure 6:

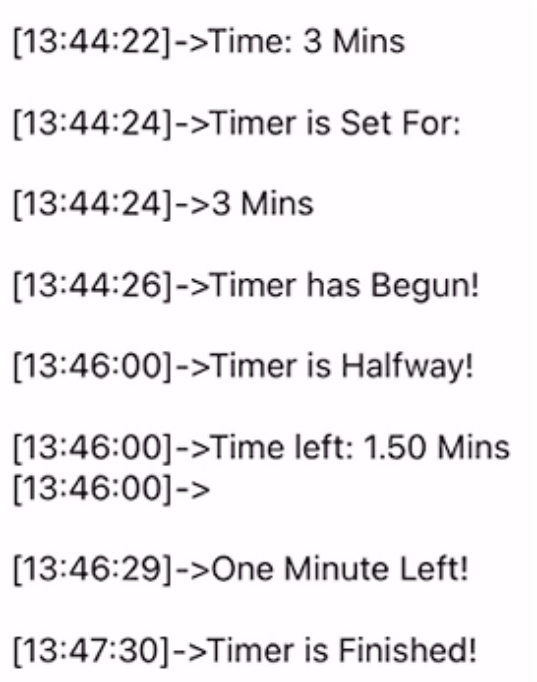

Figure 6, Food Timer Set to 3 Minutes on Device

tempConversion():

The function reads the analog voltage from the thermometer probe and uses that value to calculate a temperature in fahrenheit with a great accuracy. When tested with other food thermometers that use similar technology, the technology is always between ± 2° F. It returns the value of the converted value to where it was called.

liveTempReading():

Uses the tempConversion function and prints out the value for the user to see and also returns the value of the converted value to where it was called (loop function). Figure 7 shows the live reading of temperature through the temperature probe:

Figure 7, Live Temp Reading Example on Device

sounds():

A function that takes an integer that represents a value of which sounds you want to use. The sounds function changes the pitch of the speaker/buzzer in specific sequences to make a jingle/sound. The following sounds are listed below with the corresponding integer value for the sound:

0. Option Select Sound
1. Error Sound
2. Start Up Sound
3. Shut Down Song
4. Timer Start
5. Timer Halfway
6. Timer Last Minute
7. Timer Finish
8. Temperature Detected
9. Food is Chosen
10. Food is Almost Done
11. Food is Cooked

Flowchart (Figure 8):

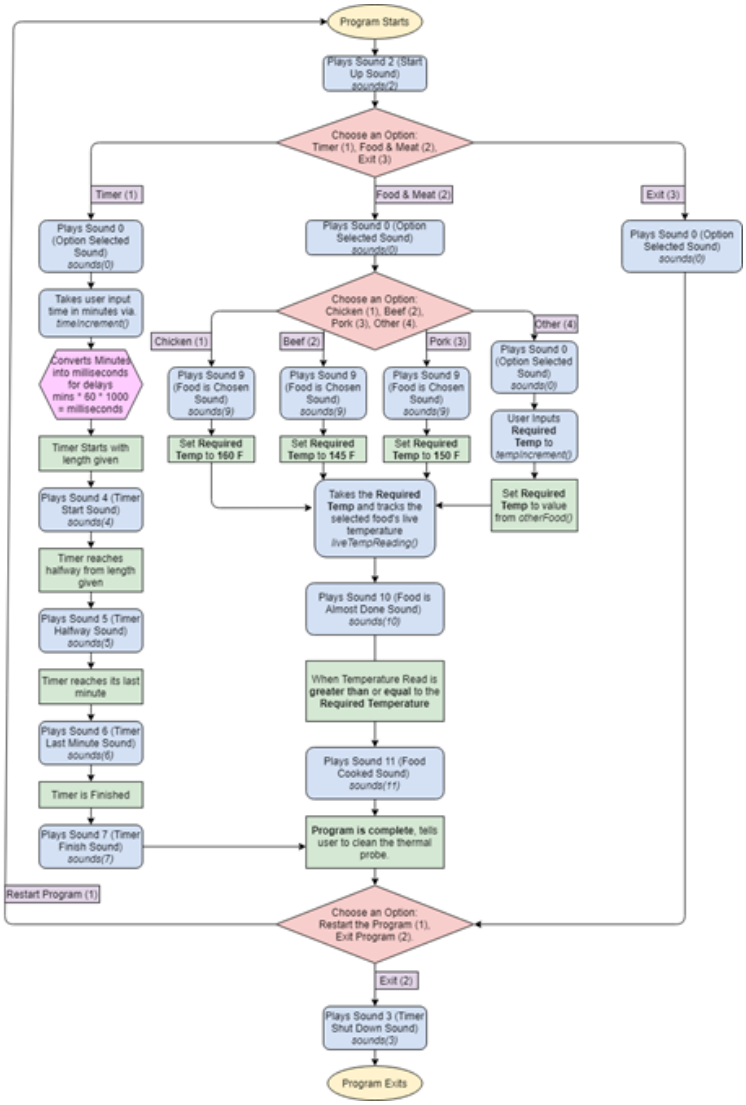

Figure 8, Flowchart of the Software

Flowchart Legend (Figure 9):

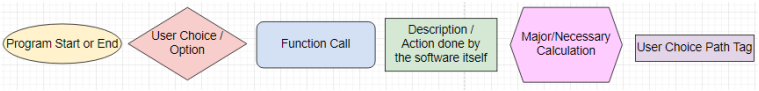

Figure 9, Legend for the Flowchart

Main Changes in Software:

A Major Change in the software is that when the option from the main menu is to exit it goes to the end of the code and asks the user to either restart the program or exit it once more. This was changed as the program would seem redundant if the user wanted to exit the program and turn off the Thermo**meat**er, so if they needed to restart the program they could press the reset button instead.

The first minor change is that the flowchart does not officially show the menus and button maps when the user chooses an option, but mentions the sound functions and when they are used.

Another minor change also is at the beginning of the program starting, it does not mention the sequence of how the startup occurs where it shows a device is connected or that a temperature sensor is detected as well.

Screenshots of Program on Smart Device:

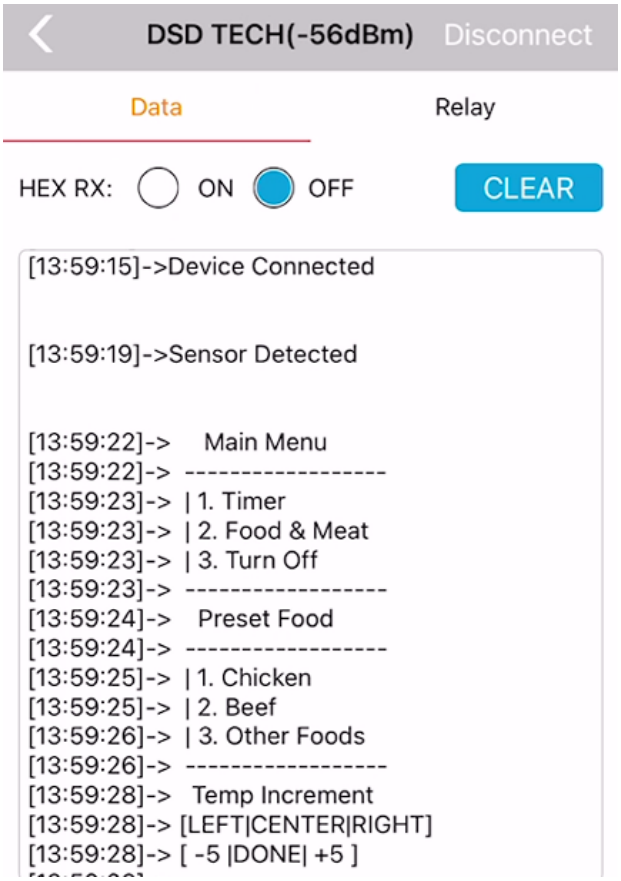

Figure 10, Start of the Program on Device

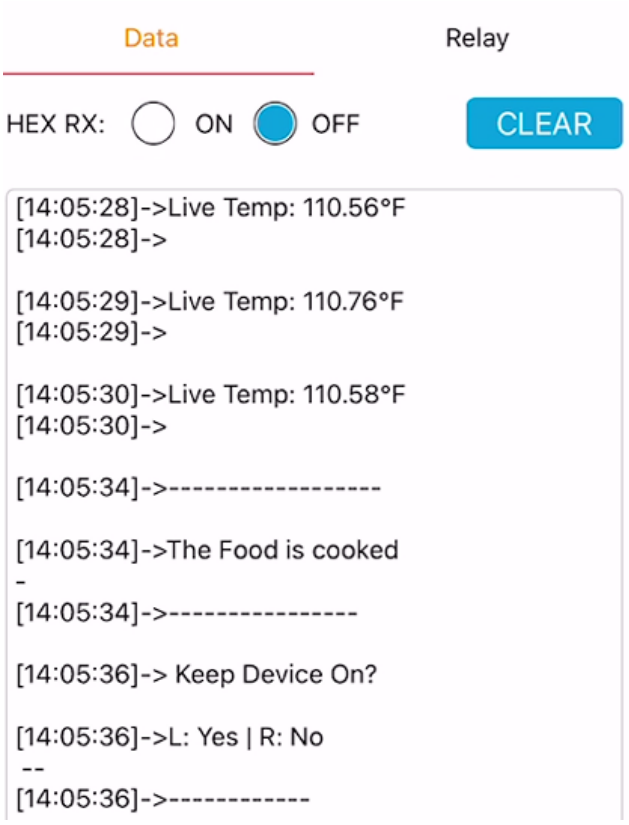

Figure 11, After Food is Cooked on Device

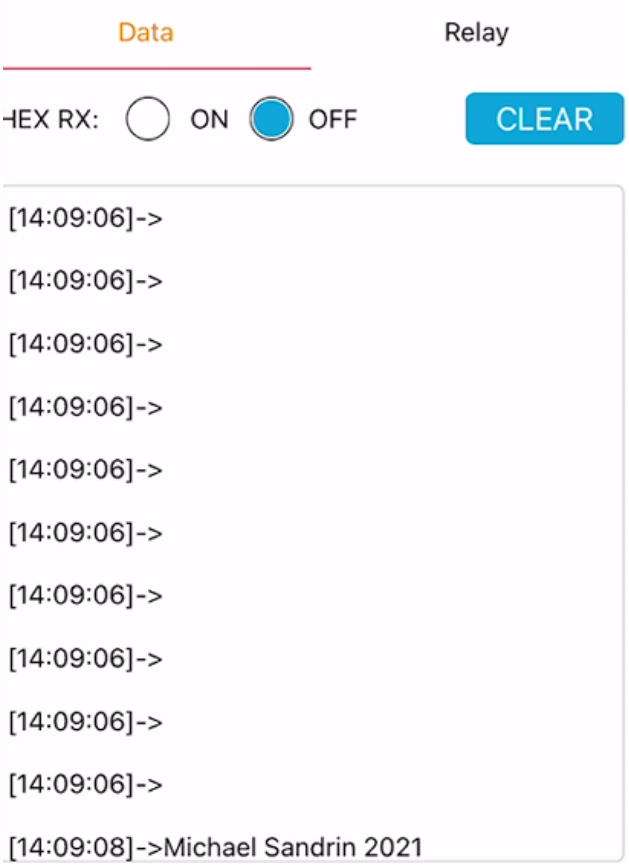

Figure 12, System Turning Off shown on Device

Designs Techniques and Tricks:

One technique is that the buttons are essentially a *do-while loop* with a nested *else-if statement* for each statement for one of the three buttons on the array.

B. Hardware

Figure 13 is a diagram of the final Thermo**meat**er design consisting of all of its components. The components are listed below:

- Arduino DUE
- Breadboard
- HM-10 Bluetooth Module
- TS/Mono Jack
- Speaker/Buzzer
- Three Button Array

Note that the Thermometer Probe is missing from the diagram as it does not directly connect to the Arduino Board but through the TS/Mono Jack instead.

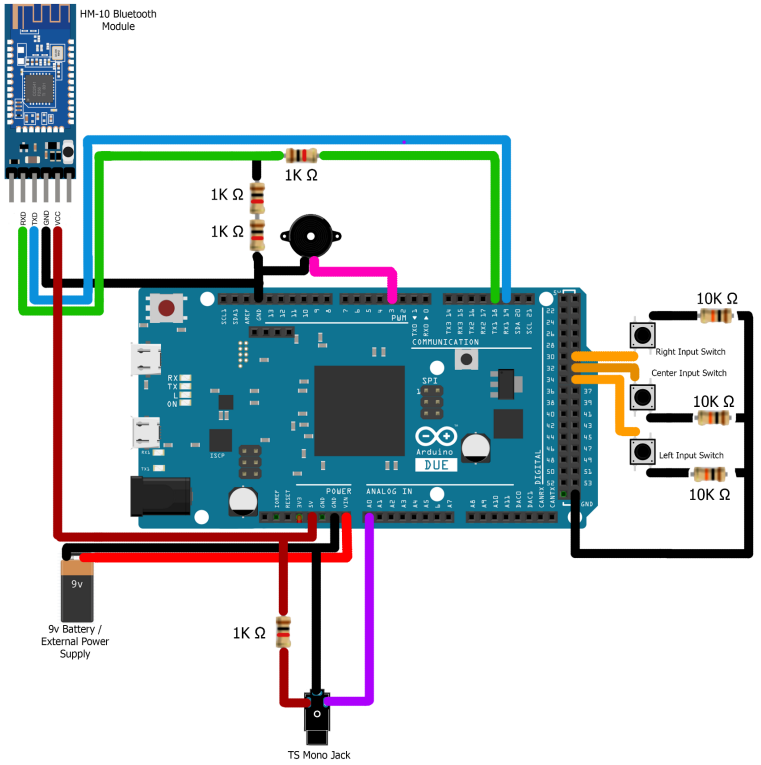

*Figure 13, Circuit Diagram of the Thermo**meat**er*

Arduino Due:

A Microcontroller that connects to a computer to then program the board in C to work with the components attached to it, allowing an embedded system to be made. In this project, all the components are attached to the Arduino board in a specific board so the analog, digital, and serial communications can be done accordingly and integrated

properly to create the final device. With this the devices attached to the pins can be controlled and accessed by the software and allows the device to function as intended. Figure 14 shows Arduino DUE board from Figure 13:

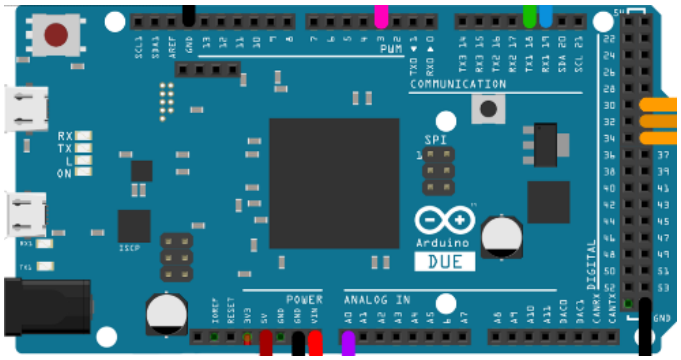

Figure 14, Arduino Board from Diagram

Breadboard:

The Breadboard acts as a device with a path of conductible metals, allowing components to be placed on it and to be connected directly to the Arduino board. Although the Breadboard is not shown in the diagram, it is assumed that the other components listed would be attached to the Breadboard in combination with the appropriate wires and resistors, all while having wires also connecting those components to the Arduino board itself.

HM-10 Bluetooth Module:

The HM-10 Bluetooth Module is used to connect the Thermo**meat**er to the users smart devices (IOS or Android devices) and display the data and user interface of the device. The bluetooth module uses both a Receive (RX1) and Transmission (TX1) pin to send data to the device from the software created using serial signals. The Thermo**meat**er's Bluetooth module is produced by a company named DSD-Tech, which means that if you want data to display to the user's smart device, the user must use the DSD-Tech application on the App Store and Google Play Store. The DSD-Tech HM-10 Module is not fully compatible with other HM-10 Applications, meaning it may not work as intented for the user to have a seamless user experience. Figure 15 shows HM-10 Bluetooth Module from Figure 13:

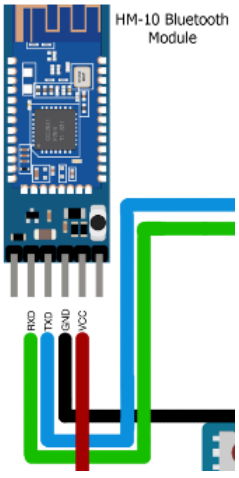

Figure 15, Bluetooth Module from Diagram.

TS/Mono Jack:

The TS/Mono Jack is solely responsible for one purpose and the most important purpose of all, which is reading temperature. The Mono Jack receives an analog signal in millivolts for the Arduino board to receive from an analog input pin (Pin A0). Although it sends the analog signal, it is determined by the Thermometer Probe that must be plugged into the Mono Jack itself, which changes the signal based on the temperature applied to the probe. Note that the analog signal will be skewed if no Thermometer Probe is plugged into the Mono Jack itself, and only performs between 0° F and 400° F applied directly to the tip of the probe. Figure 16 shows TS/Mono Jack from Figure 13:

Figure 16, TS/Mono Jack from Diagram

Thermometer Probe:

The Thermometer Probe is a metal device that contains a thermistor at the tip of it, so when the probe is inserted into a piece of food, the probe can accurately read the internal temperature of the food the probe is inserted into. With this, the probe enters the circuit with the TS/Mono Jack and acts as a resistor in the circuit, allowing the analog signal and voltage read by the Arduino board to change accordingly to the temperature applied to the probe. This results in the temperature to be read with the Arduino board and creates the premise of the device for the user.

Speaker/Buzzer:

The Speaker/Buzzer is used as a secondary method of output for the user to understand the device itself. The speaker uses a PWM pin (Pin 3) to provide an audio signal for the speaker to output. The speaker uses various different signals to change the pitch in sequence to produce a specific sound for each event occurring with the device, all saved into one function. The Sounds function provides sounds for these specific events:

- Startup / Shutdown Sound
- Option Select
- Error Sounds
- Timer Start, Halfway, Almost Complete and Finish
- Thermometer Start, Close to Cooked, Finished Cooking

Figure 17 shows the Speaker/Buzzer from Figure 13:

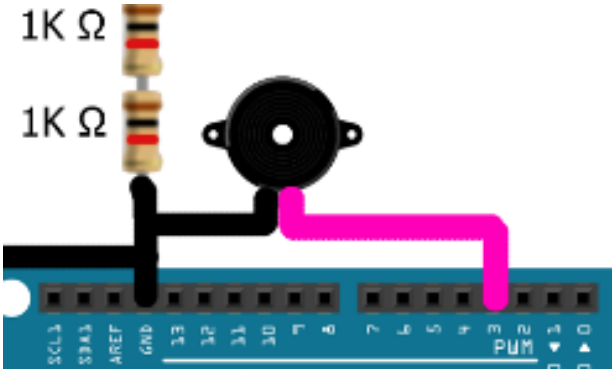

Figure 17, Speaker/Buzzer from Diagram

Three Button Array:

The Three Button Array is used for the user to interact with the device with the Three possible options at once. The two possibilities the Three Buttons that can be used for is to choose between three options or less, and for the user to change a value by incrementing or decreasing a value (used to change required temperature and timer duration). For option choosing, The furthest left button is used for *option 1*, the center button for *option 2,* and lastly the right button for *option 3*.

When incrementing a value, the left option is used to decrease a value, the center button to save the current value chosen by the user, and the right button is used to increase a value. For the timer duration you can increase or decrease by a value of *one minute*, and for the temperature you can increase or decrease the value by 5° fahrenheit. The button array has each button attached individually to a digital pin (Left Button: Pin 35, Center Button: Pin 33, Right Button: Pin 31) while they are also connected to the 10k Ω resistor each attached to ground. Figure 18 shows the Button Array from Figure 13:

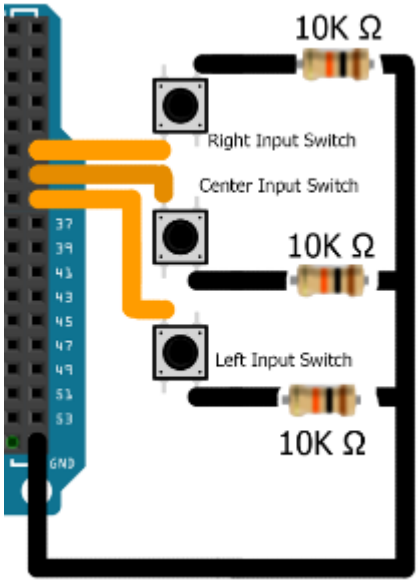

Figure 18, Three Button Array from Diagram

II. Results

The creation of the project has gone through a couple of revisions through the process. The first being the removal of the LED, as I found that it would be useless as the system already plays sounds and the user should focus more or less on what is displayed on the phone screen, resulting in it being obsolete. The second change is the addition of a Three Button Array as it would be easier for a user to have an interface made of buttons instead of having to type multiple different commands on their phone. If the user could type, I would also need to program many errors if the user inputs the wrong values.

The Thermo**meat**er functions off of every component in a specific way. The Bluetooth Module allows the user interface to display to any smart device connected to it, while the Three Button Array controls the interface and system itself. The Speaker plays sounds to notify the user of an action taking place on the system and the TS/Mono Jack in combination with the Thermometer Probe helps measure the temperature for the system to read. With this, all the components will attach to the Arduino Due board and make the overall system, which will allow for the timer function, thermometer function, and the overall system to work.

III. Discussion

The following technical issues can be found in the Thermo**meat**er system:

- **Cooling the Probe:** The thermometer probe takes some time to cooldown after performing in a high temperature (over 200° F) and if reaches such, the user will have to let the temperature probe cool down for approximately (3-5 mins).

- **App Use:** App stores have many different softwares made for HM-10 Bluetooth Modules such as the one used in the system, but the model used in this system is locked to only work on the DSD-Tech Bluetooth App. Therefore, other systems may work with any bluetooth application, but the Thermo**meat**er does not.

IV. Conclusion

The main objective of the assignment was to make a device to only measure temperature to a small LCD Display, but I found that this would be too little for my standards, so I wanted to improve as best as I can. Not only can food be cooked based on its temperature, there are many foods people buy that are precooked and just need to be warmed up just heated up on a stovetop, in an oven, or even on a grill, and the user can be cooking food without a timer nearby, such as cooking prepared food while camping. So that is why I added a timer option to the device. Second, I wanted to make the thermo**meat**er have something different in comparison to other

food thermometers on the market, and integrating the most user friendly devices would help make that true, this being the integration of smart devices.

The product of this device is certainly one that I am proud of by far as the whole system does exactly what it was intended to do from the initial planning stages. Due to the schedule of every requirement per week for the embedded system being posted and the submissions needing to be complete to track the assignments progress, I found that being ahead of those submissions and the labs, it kept me on track and always made me work on my assignment at a very consistent pace. Although education during COVID-19 made the process more difficult due to the overwhelming number of assignments between every class, the development of my embedded system was enjoyable for me and acted like a break from other work, even though it was an assignment itself, making me want to work on the assignment more. The biggest challenge was the delay I had from the parts I ordered, some of the parts coming in weeks after some that I ordered for the system, delaying the overall start to my development process.

Not only has this assignment made me learn how embedded systems perform and work with hands on experience, I have become more intrigued in learning other types of systems, such as Raspberry Pi 4.0 in which I want to work on other side projects to expand my knowledge in the category, With this, I can show my ability to make a device that is useful in a person's everyday life as food preparation is a main factor of one's ability of living. With this, I can show off this in my future as experience with the knowledge needed to create it and progress on how I have improved in comparison to embedded systems I make in the future.

Acknowledgment

I would like to acknowledge the help I have received from my professor **Ebrahim Ghafar-Zadeh** for teaching me the basics and essentials to the Arduino Due board, its software, and course content with the help of my teaching assistant for my labs **Fayas Ahamed Mohamed Hasan** for giving me recommendations, feedback, and helping me stay on track with the creation of the system. Without these two amazing individuals, I would not have the knowledge and understanding to have completed this device.

Michael A. Sandrin is a 2nd Year Computer Engineering Student at the Lassonde School of Engineering at York University in Toronto, Ontario, Canada. Michael has experience working with AC generators and their systems and a passion in working with computational systems, including embedded systems such as his device, the Thermo**meat**er. Ever since Michael was young, his father Franco has always surrounded him in a STEM based environment, with him playing with building-based toys, working on home projects that require assembly and hands on work, car maintenance and much more. In addition, he also wants to follow in the footsteps of his father and grandfather, to continue the engineering passion in the family. This was the biggest factor that sparked the engineering passion as it encompassed his life from its initial stages to the present day. Michael wants to achieve one goal out of his engineering career and that is to make a positive change in the environment with its methods of electrical/computer engineering and to help the world pave a better sustainable future for society.

Program

The following program is written in C and then run in the Arduino Genuino software on an Arduino DUE board.

```
/**The Thermomeater
 *
 * A smart Food Thermometer than connects to your  smart device
 *
 * Created and Developed by: Michael Sandrin 2021
 * Course: EECS 2032
 * Instructor: Ebarhim Ghafar-Zadeh
 */

//Variables
int temp_Value = 0;
int reqTemp = 0;
boolean menuExit = false;
boolean optionExit = false;
boolean foodChoiceExit = false;
boolean readingExit = false;
int milliseconds = 0;
int mainMenuChoice = 0;
String foodName = "No Name";

//Setup ****************************************

void setup() {

// Speaker Audio Signal
#define audio_Pin 3

//Analog Temperature Signal
#define temp_Sensor A0

//User-Interface Buttons
#define BUTTON_LEFT 35
#define BUTTON_CENTER 33
#define BUTTON_RIGHT 31

//Start of the Serial for the Bluetooth Module
Serial1.begin(9600);
delay(500);
menuExit = false;

//Setting Pin Modes
pinMode(audio_Pin, OUTPUT);
pinMode(temp_Sensor, INPUT);
pinMode(BUTTON_LEFT, INPUT);
pinMode(BUTTON_CENTER, INPUT);
pinMode(BUTTON_RIGHT, INPUT);


// Plays Start Up Sound and Tells User "Connection Successful!"
sounds(2);
Serial1.println("Device Connected");
Serial1.println("");
delay(3000);
sounds(8);
Serial1.println("Sensor Detected");
Serial1.println("");
delay(3000);

}

//Start of the Loop ****************************

void loop() {

//Loop Variables
int liveTemp = 0;
int cookedCounter = 0;
boolean cookedExit = false;
boolean cooked = false;

//First Loop for the Main Menu
 while(menuExit != true){

   //Runs the Main Menu Method
   mainMenu();
   delay(100);


 //Button Choice for Main Menu
 do{

 //Left Button Choice
 if(digitalRead(BUTTON_LEFT) == HIGH ){

  //Option Select Sound
  sounds(0);
  //Selects the First Choice
  mainMenuChoice = 1;
  //Exits the Option Selection Loop
  optionExit = true;
 }

 //Center Button Choice
 else if(digitalRead(BUTTON_CENTER) == HIGH ){

  //Option Select Sound
  sounds(0);
  //Selects the Second Choice
  mainMenuChoice = 2;
  //Exits the Option Selection Loop
  optionExit = true;
 }

 //Right Button Choice
  else if(digitalRead(BUTTON_RIGHT) == HIGH ){

   //Option Select Sound
   sounds(0);
   //Selects the Third Choice
   mainMenuChoice = 3;
   //Exits the Option Selection Loop
   optionExit = true;
```

```
  //Exits the Main Menu
  menuExit = true;
 }

 delay(150);

 //While Statement for the Button Choice
 }while(optionExit != true);

 //Resets the Option Selection
 optionExit = false;

 //The Main Menu Choice
 if((mainMenuChoice > 0)&&(mainMenuChoice < 4)){


  // Food Timer Option   ----------------------------

 //If the Food Timer is chosen
 if(mainMenuChoice == 1){

 //Food Timer Variables
 boolean timerExit = false;
 int minuteIncrement = 2;

      //Prints the Intial Time Set by Program
      Serial1.print("Initial Time: ");
      Serial1.print(minuteIncrement);
      Serial1.print(" Mins");
      Serial1.print(" ------------------");

  //Calls on Timer Button Map for User Interface
  timerButtonMap();

//Do-While Loop for the Button Choice
do{

 //Left Button Choice
 if(digitalRead(BUTTON_LEFT) == HIGH ){

  //Option Select Sound
  sounds(0);

  if(minuteIncrement > 2){

  //Lowers the Time by 1 Minute
  minuteIncrement -= 1;

      //Prints the current time for the user
      Serial1.print("Time: ");
      Serial1.print(minuteIncrement);
      Serial1.println(" Mins");


  }else{
   //Plays Error Sound
   sounds(1);

   //Displays Error
   Serial1.println("Too Little Time!");
   Serial1.println("Must Add Time!");

  }
 }

//Center Button Choice
else if(digitalRead(BUTTON_CENTER) == HIGH ){

 //Option Select Sound
 sounds(0);

     //Displays Chosen Time by User
     Serial1.println("Timer is Set For: ");
     Serial1.print(minuteIncrement);
     Serial1.println(" Mins");

     delay(2000);

   //Calls on Millisecond Conversion Function

    milliseconds =  minsToMillisec(minuteIncrement);

 //Exits the Option Selection Loop
 timerExit = true;
}

//Right Button Choice
 else if(digitalRead(BUTTON_RIGHT) == HIGH ){

  //Option Select Sound
  sounds(0);

   if(minuteIncrement < 720){

 //Lowers the Time by 1 Minute
 minuteIncrement += 1;

     //Prints the current time for the user
     Serial1.print("Time: ");
     Serial1.print(minuteIncrement);
     Serial1.println(" Mins");


 }else{

  //Plays Error Sound
  sounds(1);

  //Displays Error

  Serial1.print("Not Possible!");
  Serial1.print("Must Add Time!");
 }
}

//Set the value to true so the Menu can be exited
menuExit = true;

delay(150);
```

```
//Finishes the loop as the time has been confirmed
// by the user
}while(timerExit != true);

 //Calls on the Food Timer Function for the user
 foodTimer(milliseconds);

 }

 //Food Thermometer Option -------------------------
 else if(mainMenuChoice == 2){

    //Calls the Food Menu for the User Interface
    foodMenu();
  delay(100);

 //Food Thermometer Menu Variables
 optionExit = false;
 foodChoiceExit = false;

 //Food Thermometer Do-While Loop
 do{

   //Button Choice Do-While Loop
   do{

  //Left Button Choice
 if(digitalRead(BUTTON_LEFT) == HIGH ){

   //Option Select Sound
   sounds(0);

   //Sets Values for Chicken
   foodName = "Chicken";
   reqTemp = 165;

   //Displays Chosen Option to User
   Serial1.println("Chicken (165°F)");

   //Tells Program that an Option has been Chosen
  optionExit = true;
  foodChoiceExit = true;

 }

  //Center Button Choice
 else if(digitalRead(BUTTON_CENTER) == HIGH ){

   //Option Select Sound
   sounds(0);

   //Sets Values for Beef
   foodName = "Beef";
   reqTemp = 145;

   //Displays Chosen Option to User
   Serial1.println("Beef (145°F)");

   //Tells Program that an Option has been Chosen
  optionExit = true;
  foodChoiceExit = true;

 }

  //Right Button Choice
  else if(digitalRead(BUTTON_RIGHT) == HIGH ){

   //Option Select Sound
   sounds(0);
   menuExit = false;
   reqTemp = 100;

     //Calls on Increment Button Map for the User Interface
     incremButtonMap();

       //Loop for the User to set Required Temperature
       do{


 //Left Button Choice
 if(digitalRead(BUTTON_LEFT) == HIGH ){

  //Option Select Sound
  sounds(0);

 //If the temperature is greater than the min temperature (0°F)
  if(reqTemp > 0){

  //Lowers the Temperature by 5 Degrees
  reqTemp -= 5;

      //Prints the current temperature for the user
      Serial1.print("Current Temp: ");
      Serial1.print(reqTemp);
      Serial1.println("°F");

  }else{

   //Plays Error Sound
   sounds(1);

   //Displays Error
   Serial1.print("Temp Too Low ");
   Serial1.print("Cannot Go Lower \n");

  }
 }

 //Center Button Choice
 else if(digitalRead(BUTTON_CENTER) == HIGH ){

  //Option Select Sound
  sounds(0);

  //Exits the Option Selection Loop
   menuExit = true;
 }

 //Right Button Choice
  else if(digitalRead(BUTTON_RIGHT) == HIGH ){
```

```
 //Option Select Sound
 sounds(0);

  //If the temperature is lower than the max
  //temperature (390°F)
 if(reqTemp < 390){

 //Raises the Temperature by 5 Degrees
 reqTemp += 5;

 //Prints the current temperature for the user
     Serial1.print("Current Temp: ");
     Serial1.print(reqTemp);
     Serial1.println("°F");

 }else{

  //Plays Error Sound
  sounds(1);

  //Displays Error
  Serial1.println("Temp Too High ");
  Serial1.println("Cannot Go Higher");

 }
}

delay(150);

  //Finishes the loop as the temperature has been
  // confirmed by the user
  }while(menuExit != true);

     //Tells Program that an Option has been Chosen
     Serial1.print("Required Temp:");
     Serial1.print(reqTemp);
     Serial1.println("°F");

  //Sets Values for the Food
  foodName = "Food";

  //Sets new values for the Variables
  menuExit = false;
  optionExit = true;
  foodChoiceExit = true;
 }

//Finishes the loop as the food has been confirmed
// by the user
}while(foodChoiceExit != true);
 foodChoiceExit = false;

 // Live Temp Reading

  delay(2000);

  //Tells the User that the Temperature is now being read
Serial1.print("Now Reading Temp");
  delay(2000);

 //Do-While Loop for Temperature Reading
 do{

 //Sets the value of the Live Temperature Reading with
 // the function call
 liveTemp = liveTempReading();
   delay(1000);

//If the Temperature being read is greater than or equal
// to what is required
if(liveTemp >= reqTemp){

 //Do-While Loop to see if the next 20 Live
 // Temperature Readings are also greater than
 // or equal to what is required
 do{

   //Sets the value of the Live Temperature Reading
   // with the function call
   liveTemp = liveTempReading();
    delay(1000);

   //If the last 20 Temperature readings are less than
   // the required temperature
   if(cookedCounter < 20){

   //If the Temperature being read is greater than or
   // equal to what is required
   if(liveTemp >= reqTemp){

    //Increment the number of readings that exceed or
    // equal the required temperature
    cookedCounter++;

    //If the number of good readings is 15
    if(cookedCounter == 15){

     sounds(10);
    }

   //If the Live Temperature Reading is less than the
   // required temperature
   }else{

    //Reset the values of the variables used
     cookedCounter = 0;
     cookedExit = true;
   }

   //If the last 20 Live Temperature Readings prove the
   // food is cooked
 }else if(cookedCounter >= 20){

   //Changes the values of the variables used
   cookedExit = true;
   readingExit = true;

 }
```

```
    //Finishes the loop as the food has been cooked
    }while(cookedExit != true);
  }

  //Finishes the loop as the food has been cooked and
  // no reading is needed
  }while(readingExit != true);

  sounds(11);

  //Resets Variables if it needs to happen again
  cookedCounter = 0;
  cookedExit = false;
  readingExit = false;


   //Prints that the food is cooked on the User Interface
   Serial1.print("------------------\n");
   Serial1.print(" The ");
   Serial1.print(foodName);
   Serial1.print(" is cooked\n");
   Serial1.print("-----------------\n");

   //Sets the value to exit the loop
   optionExit = true;

 delay(1500);

 //Finishes the loop as the food has been cooked
 }while(optionExit != true);

 //Resets Variables if it needs to happen again
 optionExit = false;

  // Turning Off Option         ----------------------

  //The User chooses to turn off the Thermomeater
 }else if(mainMenuChoice == 3){

   //Calls on the Turning Off Function to print the
   // User Interface
   turningOff();

   //Creates an Infinite Loop so the user must reset
   // the Arduino Board to start the system from the
   // beginning
   while(0 != 1){
  delay(100000);
  }

  }

  //Resets the Main Menu Choice if the menu appears again
  mainMenuChoice = 0;
}


//Asks User if they want the Device Kept On
Serial1.println(" Keep Device On? ");
Serial1.println(" L: Yes | R: No ");
Serial1.println(" ---------- ");

//Do-While Loop to Take Button Input to Keep System On
do{

  //Left Button Choice
  if(digitalRead(BUTTON_LEFT) == HIGH ){

    //Option Select Sound
    sounds(0);

   //Sets Variable to Exit the Button Choice
   optionExit = true;

   //Displays that the program is restarting on User Interface
   Serial1.print("Now Restarting...\n");
   delay(3000);

   //Sets Variable to Enter the Menu Once Again
   menuExit = false;
  }

  //Center Button Choice

  else if(digitalRead(BUTTON_CENTER) == HIGH ){

    //Option Select Sound
    sounds(0);

    //Prints that the Users Choice is Invalid
    Serial1.print("Left or Right Only\n");
  }

   //Right Button Choice
   else if(digitalRead(BUTTON_RIGHT) == HIGH ){

    //Option Select Sound
    sounds(0);

   //Sets Variable to Exit the Button Choice and Enter the
   // Menu Once Again
   optionExit = true;
   menuExit = true;

   delay(1000);

   //Calls on the Turning Off Function to print the
   // User Interface
   turningOff();

   //Creates an Infinite Loop so the user must reset the Arduino
   //Board to start the system from the beginning
   while(0 != 1){
    delay(100000);
   }

   }

//Finishes as the user has chosen to reset the system/program
}while(optionExit!=true);
```

```
//Resets the Variable to be used again
optionExit = false;

//End of Loop ************************************
}

//Resets the Variables to be used again
menuExit = false;
optionExit = false;
}


//Main Menu Display ******************************
void mainMenu(){

  //Prints out the Main Menu for the User Interface
  Serial1.write("     Main Menu        ------------------");
  delay(250);
  Serial1.write("  | 1. Timer        ");
  delay(250);
  Serial1.write("  | 2. Food & Meat  ");
  delay(250);
  Serial1.write("  | 3. Turn Off       ------------------");
  delay(250);
}

//Food Menu **************************************
void foodMenu(){

    //Prints out the Food Menu for the User Interface
  Serial1.write("    Preset Food       ------------------");
  delay(450);
  Serial1.write("  | 1. Chicken      ");
  delay(450);
  Serial1.write("  | 2. Beef         ");
  delay(450);
  Serial1.write("  | 3. Other Foods    ------------------");
  delay(450);
}

//Increment Button Map ***************************
void incremButtonMap(){

  //Prints out the Increment Button Map for the User Interface
  Serial1.print("   Temp Increment   ");
  delay(250);
  Serial1.print(" [LEFT|CENTER|RIGHT]");
  delay(250);
  Serial1.print(" [ -5 |DONE| +5 ]   ------------------\n");
  delay(250);
}

//Timer Button Map *******************************
void timerButtonMap(){

  //Prints out the Timer Button Map for the User Interface
  Serial1.print("   Timer Increment   ");
  delay(250);
  Serial1.print(" [LEFT|CENTER|RIGHT]");
  delay(250);
  Serial1.print(" [ -1 |DONE| +1 ]   ------------------\n");
  delay(250);
}

//Turning Off Function ***************************
void turningOff(){

  //Prints out the Turning for the User Interface
  Serial1.write("Turning Off...");
    delay(1000);

   //For Loop to Space the Previous Text away from the Screen
   for(int i = 0; i <20;i++){
    Serial1.write("  \n");
    delay(10);
   }

   sounds(3);

   //Prints out my name and year as Credits
   Serial1.write("Michael Sandrin 2021");
}

//Minutes to Milliseconds Calculation for Delays ***
int minsToMillisec(int mins){

  //Calculation To convert Mins to Milliseconds
  int millisec = mins * 60 * 1000;

  //Returns the Millisecond Value
  return(millisec);
}

//Milliseconds to Minutes Calculation for Delays ***
double millisecToMins(int millisec){

  //Calculation To convert Milliseconds to Mins
  double mins = (((double)millisec) / 1000.00) / 60.00;

  //Returns the Minutes Value
  return(mins);
}

// Food Timer Function ***************************
void foodTimer(int milliseconds){

  //Calculates the Time of the First Half of the Timer
  double timerFirstHalf = milliseconds / 2.00;

  //Calculates the Time of the Second Half of the Timer
  //Not Including the Last Minute and Delays
  double timerSecondHalf = timerFirstHalf - 63200.00;

  //Sets the Time that would be Displayed in the User Interface
  double displayTime = 0.0;

  //Displays the Timer has Begun on the User Interface
  Serial1.println("Timer has Begun!");
  sounds(4);
```

```
  //Delays Based on the First Half of the Timer
  delay((int)timerFirstHalf);

  //Changes the display time to be printed out
  displayTime = (millisecToMins(milliseconds)) -
(millisecToMins(timerFirstHalf));

  //Prints out the Timer being Halfway and the Remaining
  //Time on the User Interface
  Serial1.print("Timer is Halfway! \n");
  Serial1.print(" Time left: ");
  Serial1.print(displayTime);
  Serial1.println(" Mins");
  sounds(5);

  //Delays Based on the Second Half of the Timer
  delay((int)timerSecondHalf);

  //Prints that there is One Minute Left on the User Interface
  Serial1.println("One Minute Left!");
  sounds(6);

  //Delays the program for one minute (60000 Milliseconds)
  delay(60000);

  //Prints that the timer is done on the User Interface
  Serial1.println("Timer is Finished!");
  sounds(7);
}

// Converts the Analog Value to Fahrenheit *********
double tempConversion(int tempReading){
  double realTemp = (100 + ((double)tempReading)/50 + 3);
  return(realTemp);
}

// Live Temperature Reading Function ***************
int liveTempReading(){

  // Takes Analog Signal from the Temperature Probe
  int tempReading = analogRead(temp_Sensor);

  //Converts the Analog Signal to the Temperature
  double realTemp = tempConversion(tempReading);

    //Prints the Live Temp to the User Interface
    Serial1.write("Live Temp: ");
    Serial1.print(realTemp);
    Serial1.write("°F \n");

    //Returns the Temperature to where it was called
    return(realTemp);
}

// Sounds Function *********************************
void sounds(int song_Choice){

//Option Select Sound
if(song_Choice == 0){
analogWrite(audio_Pin, 50);
delay(50);
analogWrite(audio_Pin, 0);
delay(50);
analogWrite(audio_Pin, 50);
delay(50);
analogWrite(audio_Pin, 0);
}

//Error Sound
else if(song_Choice == 1){
analogWrite(audio_Pin, 80);
delay(1000);
analogWrite(audio_Pin, 5);
delay(1000);
analogWrite(audio_Pin, 80);
delay(1000);
analogWrite(audio_Pin, 5);
delay(1000);
analogWrite(audio_Pin, 80);
delay(1000);
analogWrite(audio_Pin, 5);
delay(1000);
analogWrite(audio_Pin, 0);
}

//Start Up Song
 else if(song_Choice == 2){
analogWrite(audio_Pin, 1);
delay(100);
analogWrite(audio_Pin, 5);
delay(100);
analogWrite(audio_Pin, 10);
delay(100);
analogWrite(audio_Pin, 15);
delay(100);
analogWrite(audio_Pin, 20);
delay(100);
analogWrite(audio_Pin, 25);
delay(100);
analogWrite(audio_Pin, 30);
delay(100);
analogWrite(audio_Pin, 35);
delay(100);
analogWrite(audio_Pin, 40);
delay(100);
analogWrite(audio_Pin, 45);
delay(100);
analogWrite(audio_Pin, 50);
delay(100);
analogWrite(audio_Pin, 0);
delay(1500);
analogWrite(audio_Pin, 50);
delay(50);
analogWrite(audio_Pin, 0);
delay(50);
analogWrite(audio_Pin, 50);
delay(50);
analogWrite(audio_Pin, 0);
}
```

```
//Shut Down Song
else if(song_Choice == 3){
analogWrite(audio_Pin, 50);
delay(50);
analogWrite(audio_Pin, 0);
delay(50);
analogWrite(audio_Pin, 50);
delay(50);
analogWrite(audio_Pin, 0);
delay(800);
analogWrite(audio_Pin, 50);
delay(100);
analogWrite(audio_Pin, 45);
delay(100);
analogWrite(audio_Pin, 40);
delay(100);
analogWrite(audio_Pin, 35);
delay(100);
analogWrite(audio_Pin, 30);
delay(100);
analogWrite(audio_Pin, 25);
delay(100);
analogWrite(audio_Pin, 20);
delay(100);
analogWrite(audio_Pin, 15);
delay(100);
analogWrite(audio_Pin, 10);
delay(100);
analogWrite(audio_Pin, 5);
delay(100);
analogWrite(audio_Pin, 1);
delay(100);
analogWrite(audio_Pin, 0);
}

//Timer Start
else if(song_Choice == 4){
analogWrite(audio_Pin, 65);
delay(250);
analogWrite(audio_Pin, 0);
delay(750);
analogWrite(audio_Pin, 65);
delay(250);
analogWrite(audio_Pin, 0);
delay(750);
analogWrite(audio_Pin, 65);
delay(250);
analogWrite(audio_Pin, 0);
delay(750);
analogWrite(audio_Pin, 40);
delay(700);
analogWrite(audio_Pin, 0);
}

//Timer Halfway
else if(song_Choice == 5){
analogWrite(audio_Pin, 5);
delay(350);
analogWrite(audio_Pin, 25);
delay(350);
analogWrite(audio_Pin, 45);
delay(350);
analogWrite(audio_Pin, 65);
delay(1000);
analogWrite(audio_Pin, 0);
}

//Timer Last Minute
else if(song_Choice == 6){
analogWrite(audio_Pin, 25);
delay(150);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(150);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(700);
analogWrite(audio_Pin, 0);
}

//Timer Finish
else if(song_Choice == 7){
analogWrite(audio_Pin, 25);
delay(250);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(250);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(250);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(250);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(250);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(250);
analogWrite(audio_Pin, 0);
delay(150);
analogWrite(audio_Pin, 25);
delay(800);
analogWrite(audio_Pin, 0);
delay(1500);
}

//Temperature Detected
else if(song_Choice == 8){
analogWrite(audio_Pin, 65);
delay(150);
analogWrite(audio_Pin, 5);
```

```
delay(150);
analogWrite(audio_Pin, 0);
delay(1000);
analogWrite(audio_Pin, 65);
delay(150);
analogWrite(audio_Pin, 5);
delay(150);
analogWrite(audio_Pin, 0);
}

//Food is Chosen
else if(song_Choice == 9){
analogWrite(audio_Pin, 65);
delay(50);
analogWrite(audio_Pin, 5);
delay(60);
analogWrite(audio_Pin, 65);
delay(1000);
analogWrite(audio_Pin, 0);
}

//Food is Almost Done
else if(song_Choice == 10){
analogWrite(audio_Pin, 65);
delay(150);
analogWrite(audio_Pin, 5);
delay(150);
analogWrite(audio_Pin, 0);
delay(400);
analogWrite(audio_Pin, 65);
delay(150);
analogWrite(audio_Pin, 5);
delay(150);
analogWrite(audio_Pin, 0);
delay(400);
analogWrite(audio_Pin, 65);
delay(150);
analogWrite(audio_Pin, 5);
delay(700);
analogWrite(audio_Pin, 0);
}

//Food is Cooked
else if(song_Choice == 11){
analogWrite(audio_Pin, 8);
delay(200);
analogWrite(audio_Pin, 40);
delay(400);
analogWrite(audio_Pin, 8);
delay(200);
analogWrite(audio_Pin, 40);
delay(400);
analogWrite(audio_Pin, 8);
delay(200);
analogWrite(audio_Pin, 135);
delay(1500);
analogWrite(audio_Pin, 0);

}
}
```

Program Details:

- 994 Lines of Code
- Last Run on Arduino 1.8.13 Software
- Run on Arduino DUE Board