Lab 1

Thermometer with Web Interface

Team 7:

Matthieu Stogsdill, Makenna Maguire, Tim Evans, Austin Wittenburg

1. Design Documentation

Hardware Design

i. Product Overview

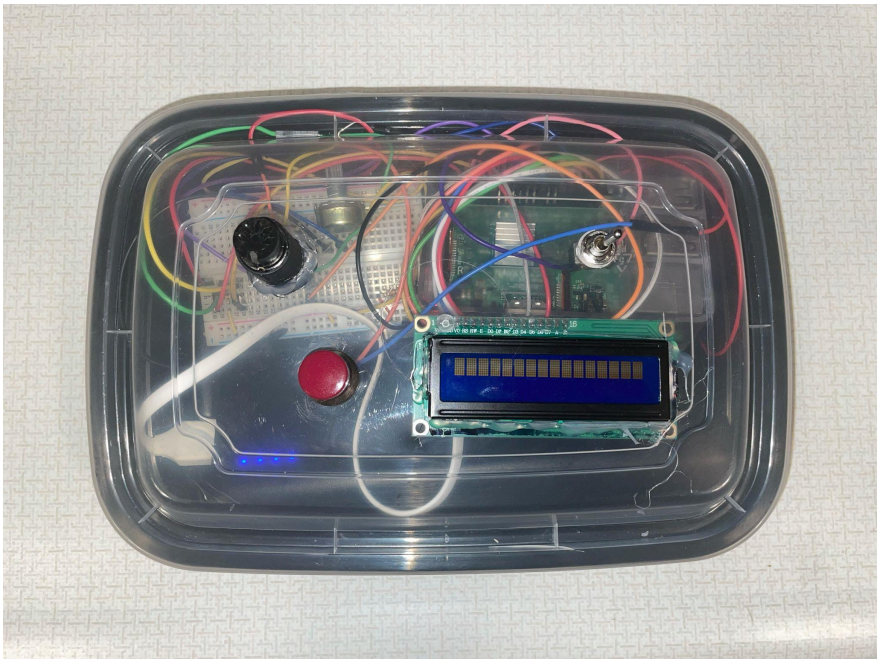


Figure 1: Final Prototype of Digital Thermometer with Web Interface

The final design for the 3rd box of the digital thermometer can be seen from **Figure 1**. Our final design consists of several components including a raspberry-pi as the brain and main processing unit for the product, a LCD screen where data and update messages are displayed, a toggle latch switch for turning the entire device on and off, a custom five pin female XLR input to be compatible with our custom temperature probe, a 20100 mAh battery, and lastly a momentary push button that allows the user to turn on the LCD screen for data readings. The raspberry-pi along with all wiring, breadboard, and battery are enclosed in a makeshift container made from a tupperware with a removable lid. The button, and toggle switch are fastened to the lid of the container with a washer and nut while the LCD and XLR cable input are hot glued to the lid of the container. The battery, raspberry-pi, and breadboard are all secured to the inside of the container with velcro.

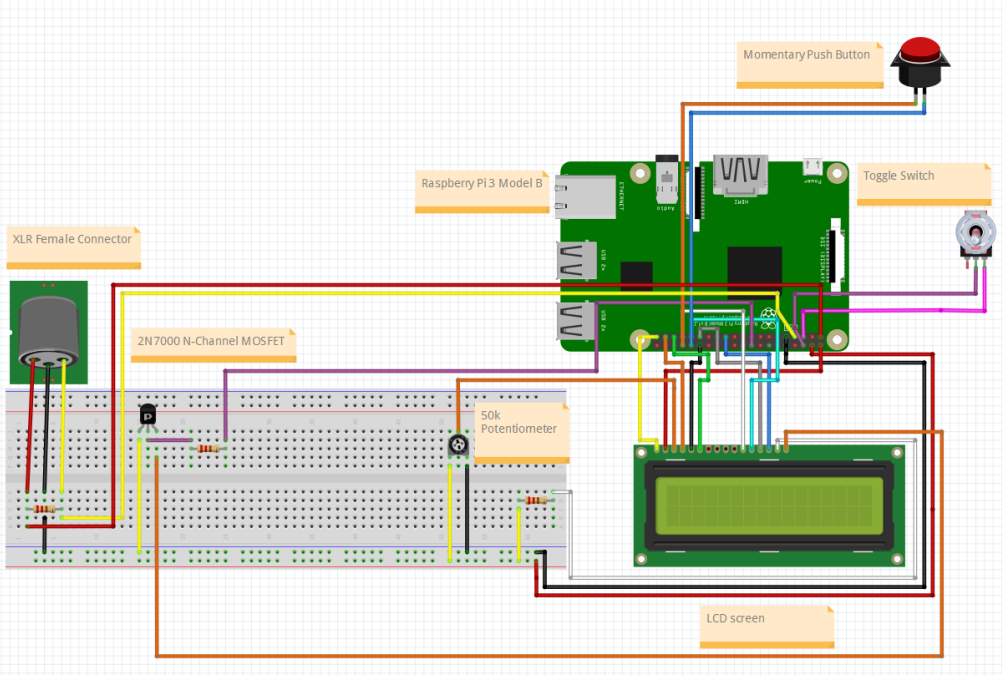


Figure 2: Final Circuit Schematic of Digital Thermometer with Web Interface

ii. List of Hardware Components

a. DS18B20 Waterproof Temperature Sensor Cable

Figure 3: DS18B20 Waterproof Sensor Cable (Unmodified)

The temperature sensor used for this design is the DS18B20 Waterproof temperature sensor cable. It is a digital thermo probe that uses 1-wire interface making it easy to communicate with raspberry-pi and send temperature data. It is able to measure temperatures ranging from (-55° C to +125° C) or (-67° F to +257° F) which meets the requirements for this lab. It has a stainless steel head allowing it to endure wet conditions. It connects via three wires (Data, Ground, Vcc) which can be seen in **Figure 3**. The operating supply voltage to Vcc is anywhere within the range of 3.0V - 5.5V which fits nicely with either of the raspberry-pis 3.3V or 5.0V supply pins.

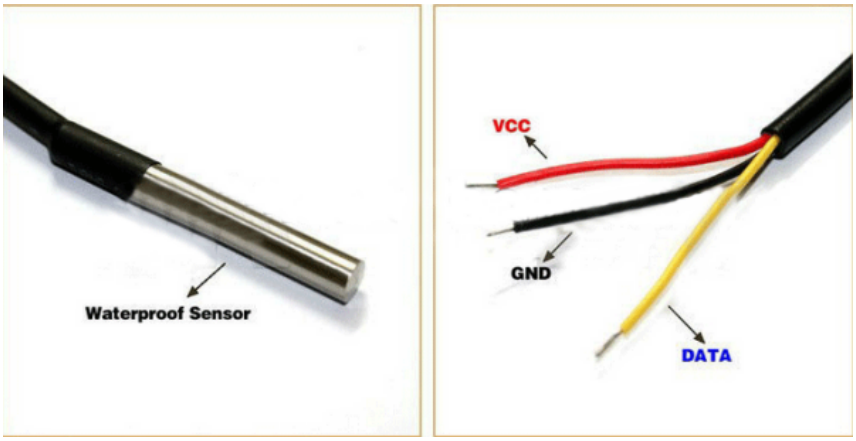


Figure 4: Stainless Steel Cap and Pin Definitions for DS18B20 Temperature Probe

b. 5-Pin XLR Cable Connectors Male/Female

Figure 5: 5-Pin XLR Cable Connectors Female (left) and Male (right)

As part of the requirement for this design the temperature probe must have a terminating connector. As can be seen from **Figure 4** the probe originally had free wires to make the connections so modifications needed to be made. In the final design of the product the data,

ground, and vcc wires were soldered to the 5-pin XLR male connector. On the female side wires were soldered onto the corresponding position so that the pins lined up when the temperature probe was plugged into the 3rd box via this new female XLR connector. The new modified temperature probe with both the male and female connectors can be seen in **Figure 6** and the corresponding pinout for both can be seen in **Figure 7**.

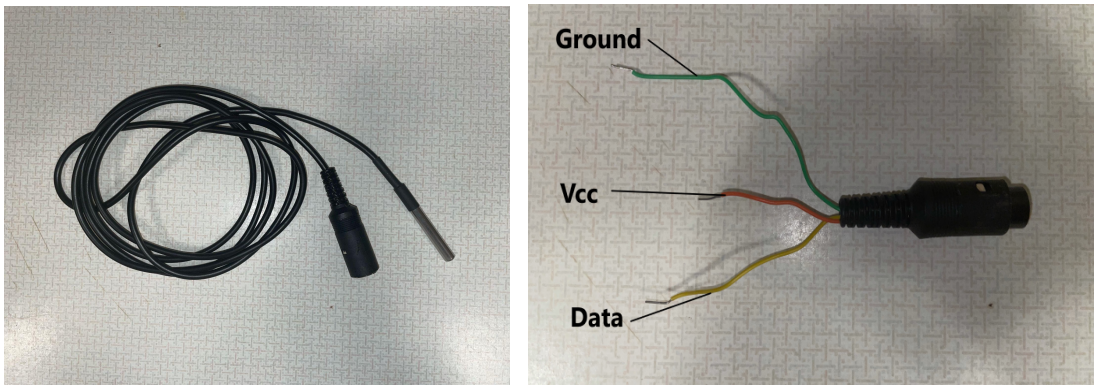


Figure 6: Modified Temperature Probe with Male XLR Connector (left) Modified Female XLR Connector (right)

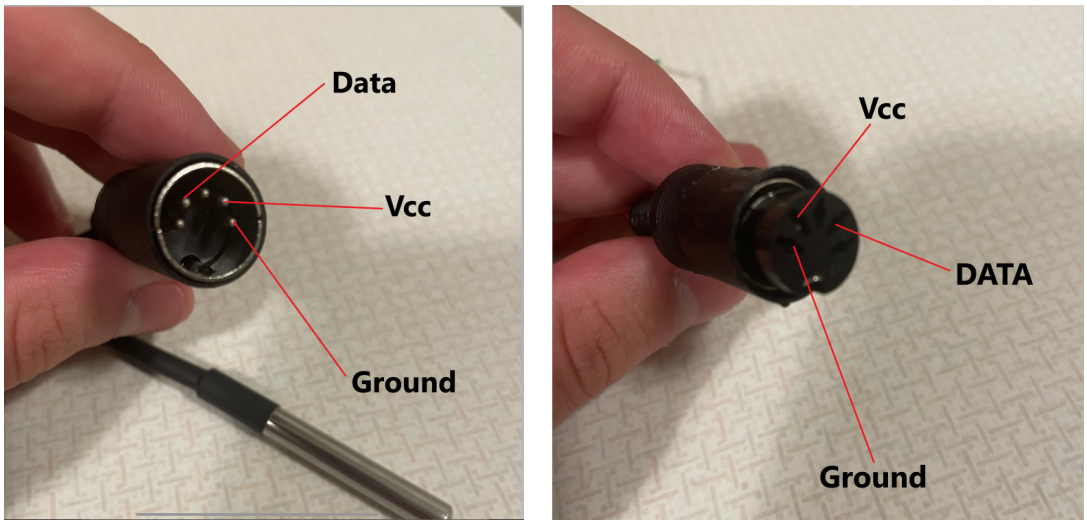


Figure 7: Pinout for Male XLR Connector (left) and Female XLR Connector (right)

c. Raspberry-Pi

Figure 8: Raspberry-Pi 3 Model B

The main processing unit for this design is the raspberry-pi 3 Model B. Since this project requires software and hardware there needs to be a central unit that connects all components of the design together. The raspberry pi is a microprocessor capable of connecting all of our hardware via the GPIO pins which a pinout can be seen in **Figure 16**. The raspberry pi runs all of the python files at bootup.It also has Wifi capabilities which allows us to remotely connect to the server it hosts and read the temperature data from another computer.

d. LCD 1602

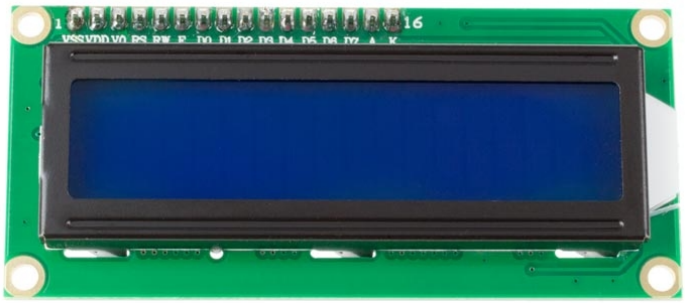


Figure 9: LCD 1602

The screen used to display data and error messages to the user is a simple 16 columns by 2 rows LCD. It is connected to the raspberry pi and other components on the breadboard via jumper wires from its pins on the upper back of the LCD. The corresponding pinouts and where they connect to on the device can be seen in **Table 1**.

e. Momentary Push Button

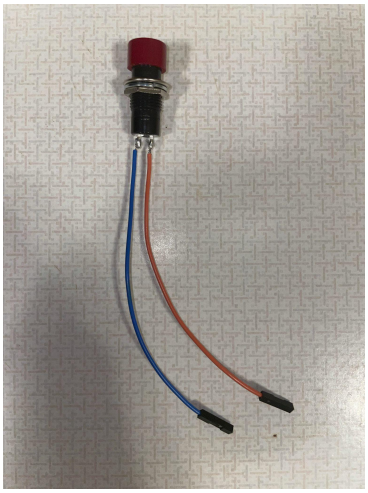


Figure 10: Momentary Push Button with Soldered Jumper Wires

The button used to control the LCD screen is a momentary push button. It is normally open and will only close the circuit if the user presses the button and only when the user is pressing it. Female Jumper wires were soldered onto the two pins so it connects straight to the raspberry pi GPIO pins.

f. Latching Toggle Switch

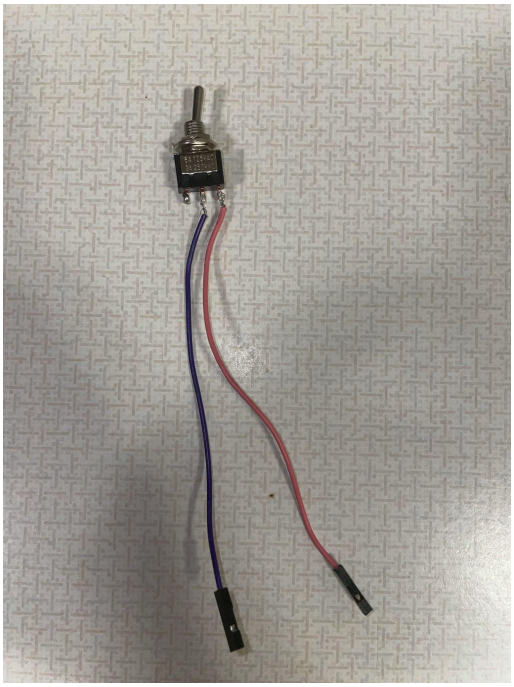


Figure 11: Latching Toggle Switch with Soldered Jumper Wires

The MTS-1 (mini toggle switch) was used to turn the device on and off. There are female jumper wires soldered onto two of the three pins so that it connects straight to the raspberry pi GPIO pins. The state of the switch and whether it corresponds to on or off is controlled by the shutdown.py python script that runs at boot up of the device.

g. 50k Ohm Potentiometer

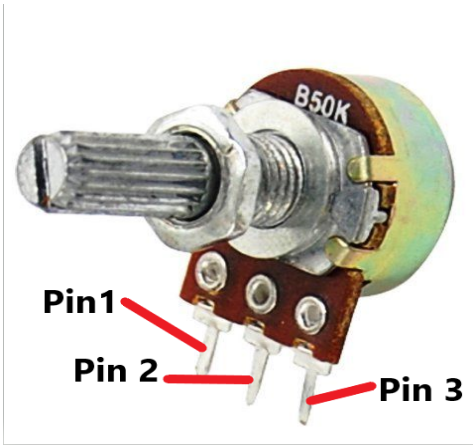
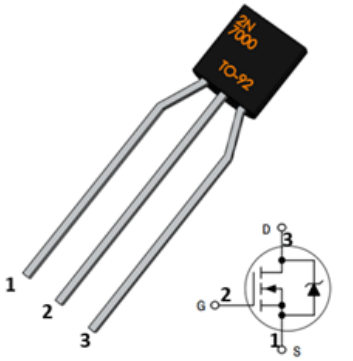


Figure 12: 50k Ohm Potentiometer

A 50k Ohm potentiometer is included with this device in order to control the contrast of the LCD. If the LCD is connected straight to the 5V supply pin of the raspberry-pi the screen can be difficult to read especially at wider angles. The potentiometer allows for contrast adjustment and is enclosed inside the tupperware to prevent changes once the resistance is set.

h. 2N700 Transistor



2N7000	
1	Source
2	Gate
3	Drain

Figure 13: 2N700 N-Channel MOSFET and Pinout

The 2N700 is a N-channel MOSFET which is used to physically control the LCD screen backlight. It is connected to GPIO-24 via its gate and the pin sends a voltage whenever the button is pushed to allow current flow and turn on the LCD backlight.

i. Anker 20100 Power Bank

Figure 14: Anker 20100 Power Bank

The Anker 20100 Power Bank is a 20100 mAh portable charger. This charger is what powers the entire device and it connects to the raspberry pi via a micro-USB cable. It is more than capable of powering the raspberry pi and all other components who use the pi as a power source as it can supply up to 5V and 4.8A.

j. Temperature Sensor Enclosure

Figure 15: Temperature Sensor Enclosure

The enclosure used to house the temperature sensor is a modified tupperware container with holes cut for the LCD display, button, toggle switch, and XLR cable connector.

k. Miscellaneous Components

There are a few components that are non-specific but still necessary for the successful operation of this device. These components are as follows

-Breadboard

-Velcro

- Micro-USB cable

-Jumper wires (female-female, male-male, female-male)

-3 Resistors (1k Ohm, 4.7k Ohm, 330 Ohm

i. GPIO Pinouts & Wiring Diagram/Block Diagram

Component	Component Pin	Raspberry Pi GPIO Pin	
		BCM	BOARD
LCD 1602	Ground (GND)	Ground (GND)	39
	Vcc	5V	2
	Contrast	Ground (GND)	Potentiometer
	RS	GPIO-26	37
	R/W	Ground (GND)	30
	Enable (E)	GPIO-19	35
	D4	GPIO-13	33
	D5	GPIO-06	31
	D6	GPIO-05	29
	D7	GPIO-11	23
	LED+ (A)	5V	4
	LED- (K)	Ground (GND)	Breadboard
Temperature Probe	Vcc	3.3V	1
	Ground (GND)	Ground (GND)	9
	DATA	GPIO-04	7
Button	Pin 1	GPIO-12	32
	Pin 2	Ground (GND)	34
Toggle Switch	Pin 1	GPIO-03	5
	Pin 2	Ground (GND)	6
50k ohm Potentiometer	Pin 1	5V	4
	Pin 2	LCD Contrast	3
	Pin 3	Ground (GND)	9
2N700 MOSFET	Drain	LED- (K)	15 on LCD
	Gate	GPIO-24	18
	Source	Ground (GND)	9

Table 1: Component Pinout and Corresponding Connections

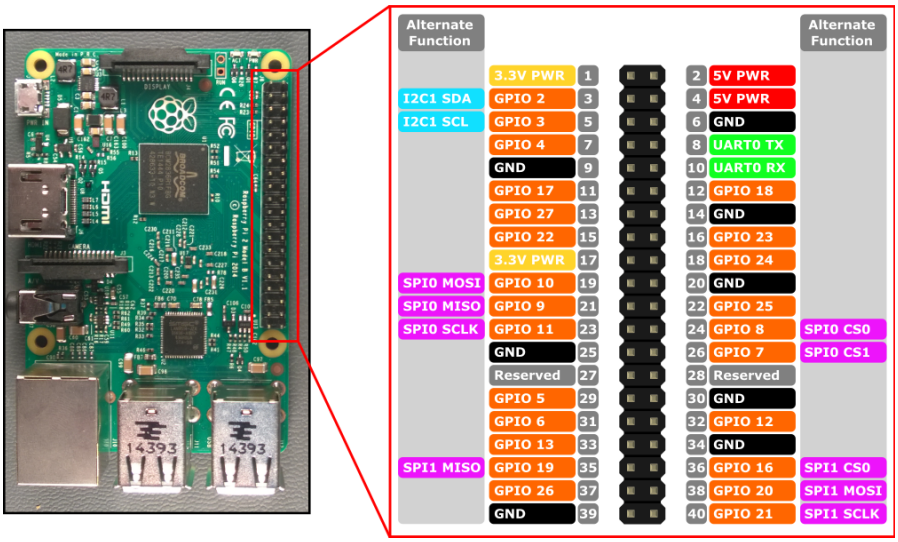


Figure 16: Raspberry-Pi 3 GPIO Pinout

ii. Software Design

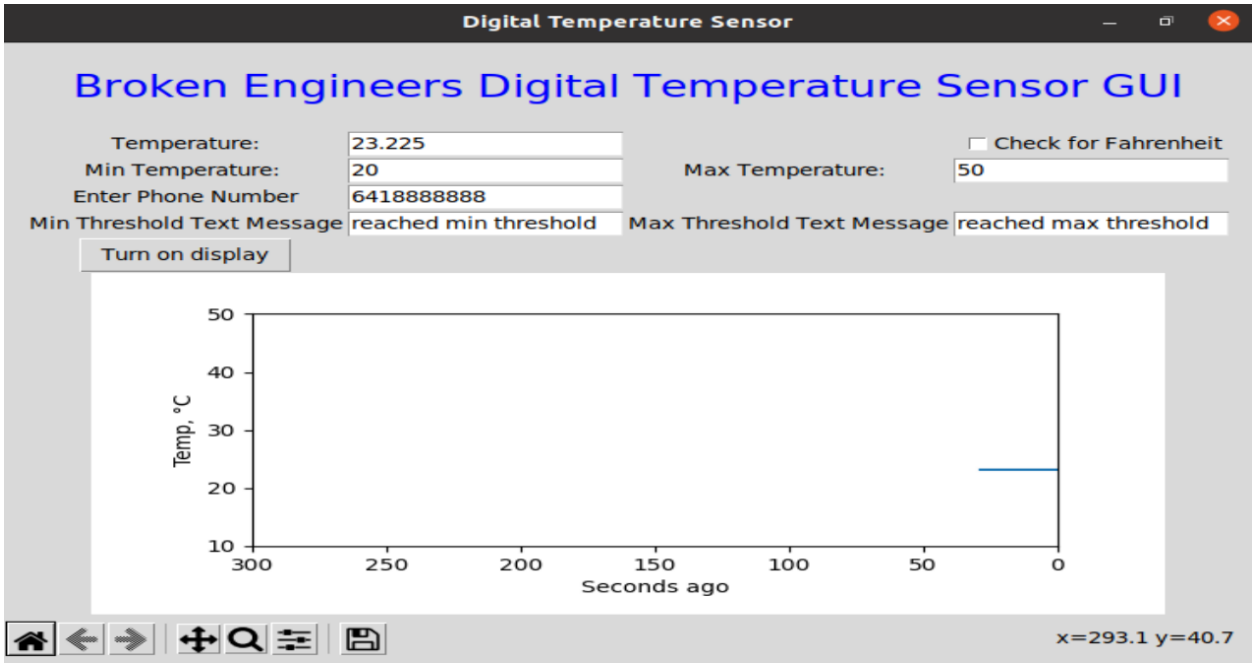


Figure 17: Client-Side Graphical User Interface

The software components of our product consists of two python scripts running on the Raspberry Pi and one python script running on a separate computer. The Pi automatically runs these two scripts on boot via rc.local file manipulation. The first script continuously checks two GPIO pins for a change in state due to the switch being turned on or off. If the switch is flicked off, the Pi runs a bash command to safely shutdown and vice versa. The second script starts a server, continuously checks for a button press, sends temperature data once a second, and continuously checks for client connection and messages. On the computer, a script runs a client which connects to the Pi server. The client receives temperature data at a rate of 1 reading per second. The client then displays this data on a graph in the graphical user interface as in **Figure 17.** The GUI provides several options such as changing the temperature data from fahrenheit to celsius, sending custom text messages to any phone number, and changing the upper and lower temperature thresholds which trigger this text message.

Server

shutdown.py:

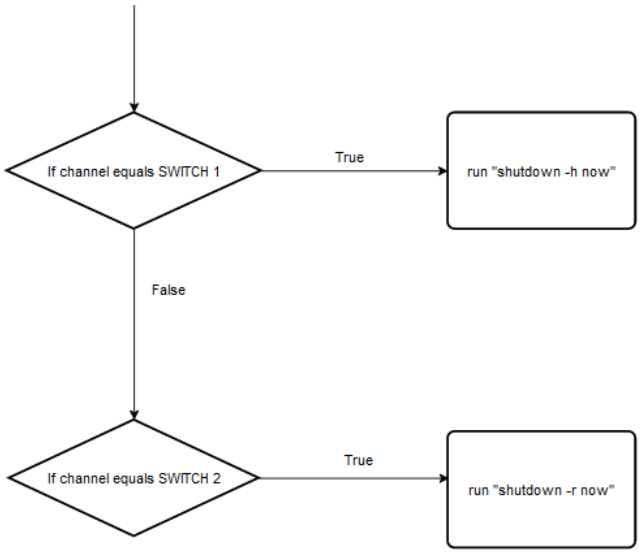


Figure 18: shutdown.py flowchart

Shutdown.py is one of the two files that runs at the startup of the raspberry pi and it monitors the state of the toggle switch. The file initially binds one pin of the toggle switch to a GPIO pin of the raspberry pi and the other switch pin to a ground pin of the raspberry pi. The script then constantly checks whether there is a falling edge or rising edge in the pin output state which corresponds to the physical position of the switch. When the switch is flipped into the on position it sends a boot up command to the operating system and powers on the pi and its attached components. When the switch is flipped into the off position it sends a shutdown command that ends all ongoing processes and safely turns the pi and its attached components off.

Tempsensorserver.py:

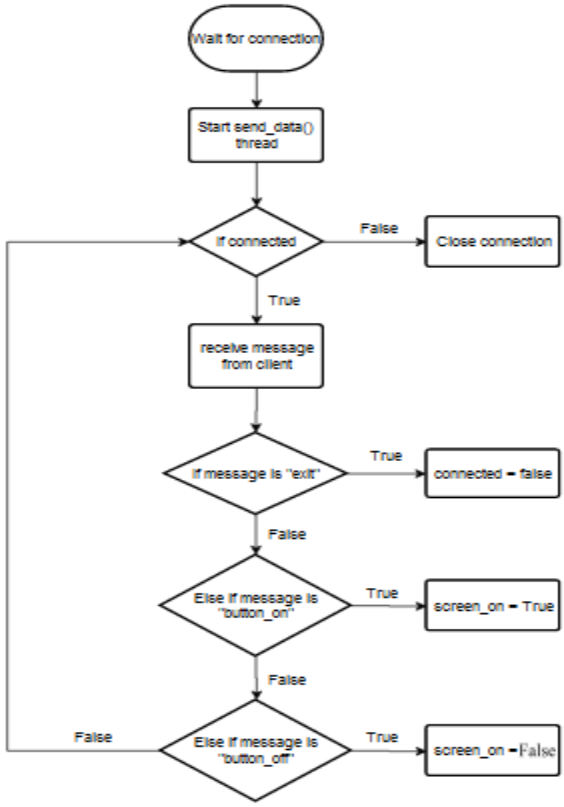


Figure 19: handle_client() thread conditional flowchart

Tempsensorserver.py starts by initializing the server socket on port 8090. After setting up GPIO settings and configuring directory information, the script starts the handle_client() and check_button() threads. The handle_client function first waits for a connection from the client. Then after the client connects, it starts the send_data() thread. This function reads temperature data from the file that is produced by the temperature sensor. If the last three characters are not "YES" then the function returns "NoData" as shown in **Figure 20**. Else it returns the temperature in a formatted way. After the send_data() thread is started, the handle_client() thread starts a while loop to receive messages from the client. If the message is "exit" it closes the connection. Else if the message is "button_on" it sets a global variable screen_on to True. Else if the message is "button_off" it sets the global variable screen_on to False. This global variable is used in the check_button thread. The check_button thread contains an infinite while loop. It checks to see if the button is pressed or if the screen_on global variable is true. If so, the lcd screen is powered and either "NoData" or the temperature is written to the screen depending on if there is data as in **Figure 19.**

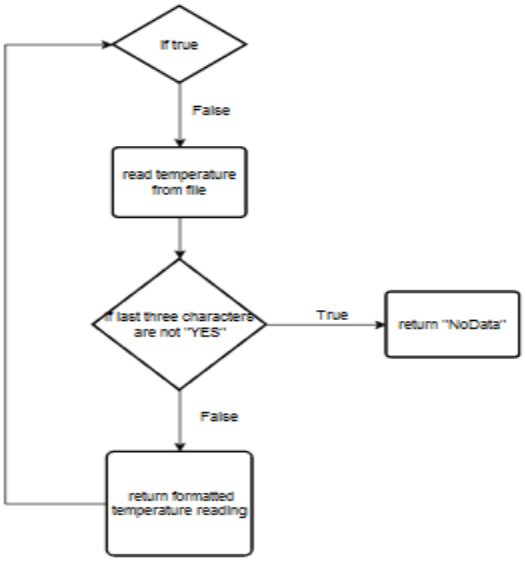


Figure 20: send_data() thread conditional flowchart

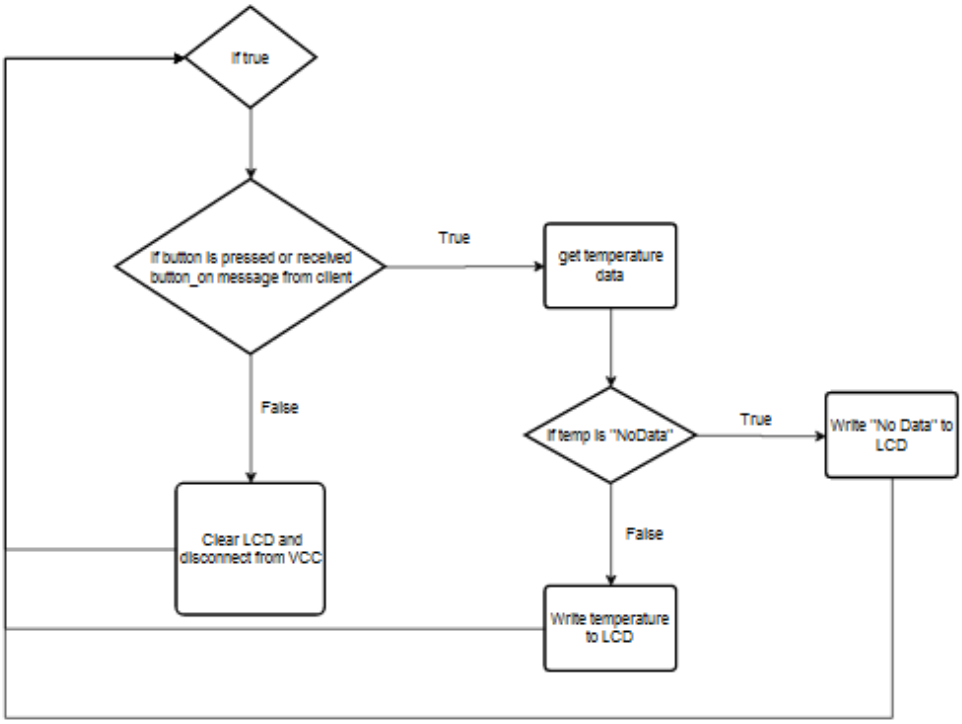


Figure 21: check_button() thread conditional flowchart

Client

graphclient.py:

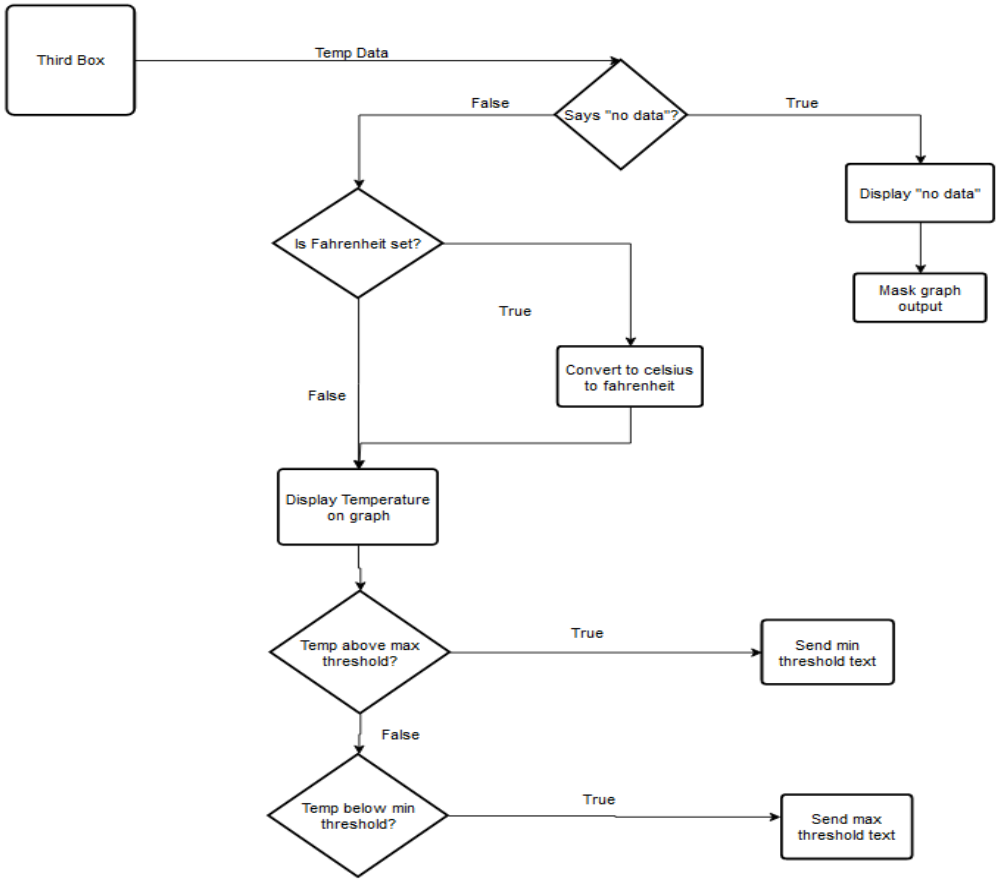


Figure 22: graphclient.py flowchart

Graphclient.py runs on a computer separate from the Raspberry Pi. First, the script connects to the server over socket 8090. Then it adds components to the graphical user interface using tkinter. The script then starts to call the next_step() function every second. This function receives a message from the server containing temperature data in celsius or a "no data" message. If the message is "no data" then the string "NoData" is displayed to the GUI. Otherwise, the temperature is displayed to the GUI in fahrenheit if the checkbox is checked and in celsius otherwise. To create the graph in an animated style from right to left, the temperature data is reversed before it is plotted. The temperature is then plotted and any missing data points are left out using a numpy masked array. At the end of next_step() a thread is created to run the textMessage() function. This function checks if the temp is above the max threshold or below the min threshold. If either of these conditions are met, an sms is sent to the specified phone number. The "Turn On Display" button widget on the GUI is bound to a function called turnOnDisplay(). If the button is pressed, a special message "button_on" is sent to the server and the button is reconfigured to show the words "Turn Off Display" as in **Figure 23** below. If that button is pressed, a message "button_off" is sent to the server" and the button is reconfigured to show the

words “Turn On Display.” After the gui is closed, the client sends the message “exit” to let the server close the connection properly and closes the connection itself.

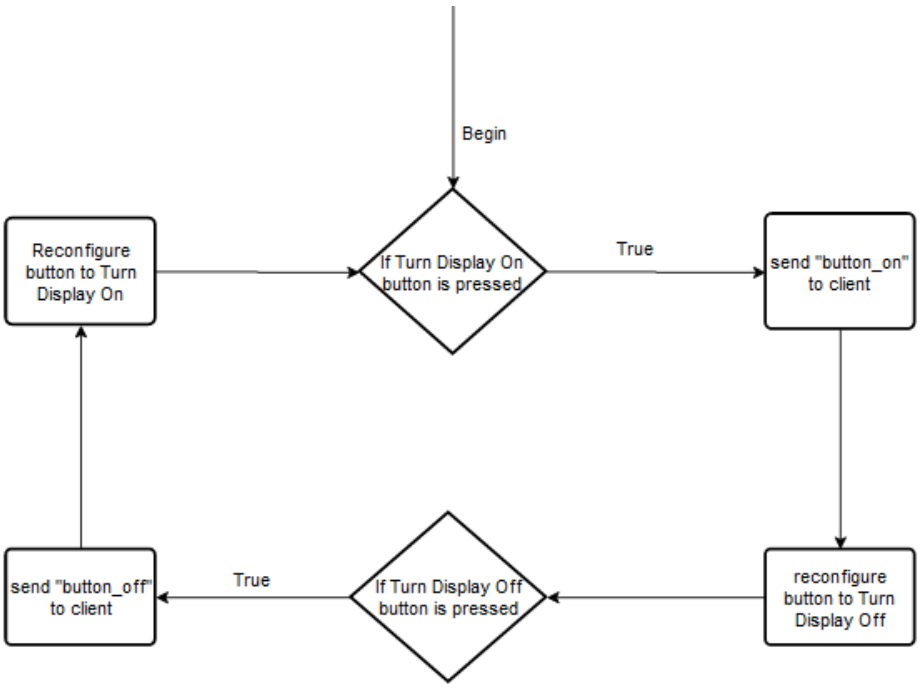


Figure 23: GUI button reconfiguration

2. Design Process and Experimentation

For our design we compared two different design choices. The major design pieces we compared were to use an Arduino or a Raspberry Pi. We ultimately decided on the raspberry pi for a couple reasons. These reasons are: a raspberry pi is 40 times faster than the Arduino when it comes to clock speed, the Pi also has 128,000 times more RAM, and it can multitask supporting two USBs and can wirelessly connect to the internet. The raspberry pi and arduino both were great options and some of the reasons for choosing an Arduino are: arduino’s are much simpler which makes hardware implementation better, also they have a real time and analog time property that pi’s do not have, another piece is the Arduino IDE is much more user friendly. We ultimately chose the Raspberry Pi for the reason that it is more useful for this project. We believe this because the pi is faster making it more efficient and this project was very complex so the faster clock speed on pi is needed to implement the amount of software and hardware we needed.

One of the hardware design tradeoffs we came to was what type of terminating connector to make for the temperature probe. The probe needed to still read temperature when plugged in and also notify the system if it was unplugged. There also needed to be at least three channels for the Vcc, ground, and data connections in this connector. We originally decided that we would solder the temperature probe to a 3.5mm Headphone jack and use that as our connector and wire an audio jack port onto the final design where the cable could be plugged in. The headphone jack had three channels which was perfect to fit the needs of our project as it had the tip, ring, and sleeve connections for vcc, ground, and data. When implementing this design we found that when you connect the temperature probe to the system there is a momentary contact between the 5V input and ground and it shorts the system causing the raspberry to reboot. The design choice we ended up going with is using male and female 5-pin XLR connectors which provide the same level as connection but without the contact of one pin touching the other.

The next step we took was to make sure we understood all the software requirements and made a detailed agenda for what we wanted to accomplish with our final prototype. We made a list of the requirements and which ones had higher priority and needed to be accomplished first. The software design choices we made first started with which program we all wanted to work with. After listing all the different languages including Python, C++ and Java we came to a mutual conclusion that we were all comfortable with Python. This design choice was made purely because it was best for our group. We needed to complete this lab in a short period of time and working in a language we all have worked on before was the reason we chose Python. The next software piece that we debated on was how to communicate between the pi and computer using the client-server connection. We ended up doing a graph client GUI which encompassed all the requirements into one python script. This allowed us to control all pieces of software from this file and keep it organized. The software components of this lab will be changed periodically over the course of the lab since with the testing protocols this lab requires, it has to be constantly updating. The choices we thought would work in the design portion are not always going to be the best to achieve the goal of the lab.

3. Test Report

Test Procedure	Expected Result	Pass/Fail	Value
Measure the temp sensor cable	The cable should be 2 meters long, ± 0.1 meters	Pass	2.03 meters
Turn the power switch to the on position	The system should boot up and connect to the internet	Pass	

Press the button	The display should turn on and show the user the temperature in degrees celsius	Pass	
Observe the data being displayed	The data should appear with no noticeable delay (less than 20 ms) after pressing the button	Fail	877 ms
Observe the data being displayed	The temperature should update about once a second	Pass	
Release the button	The display should turn off and no longer display anything	Pass	
Observe the display	When the button is released the display should turn off with no noticeable delay (less than 20 ms)	Pass	15 ms
Unplug the temp sensor and turn on the display	The display should notify the user that the temp sensor is not sending data to the system	Pass	“No Data”
Plug the sensor back in and turn on the display	The display should now continue to show the temperature in degrees celsius	Pass	
Dip the temperature sensor in ice water	The temperature readout should be 0 degrees celsius ± 2 degrees	Pass	2 ℃
Examine the exterior of the system and give the box a shake	The box should be physically robust and capable of withstanding reasonable amounts of force	Pass	
Examine the connections of the system	The components should have terminating connectors	Pass	
Connect to the temp sensor via wifi	The system should allow for wifi connected computers to access it and get the temperature data from it	Pass	
Use the computer to turn on the display	The computer should be able to remotely turn on the display and show the temperature	Pass	

Use the computer to turn off the display	The computer should be able to turn the display off remotely	Pass	
Examine the graph on the computer application	The computer application should have a graph displaying the temperature data over the last 300 seconds	Pass	
Examine the graphs bounds	The graphs upper bound and lower bound should be 50 and 10 degrees celsius respectively	Pass	Upper: 50 Lower: 10
Watch the graph as it collects temperature data	The graph should scroll from right to left when new data is collected.	Pass	
On the computer, switch the temperature from degrees celsius to degrees fahrenheit	The computer should be able to display the temperature in either degrees celsius or degrees fahrenheit and switch between them on command	Pass	
Unplug the temp sensor from the box again	The computer application should clearly show on the graph when there is missing data and the temperature readout should acknowledge the disconnected temp sensor	Pass	“No Data”
Plug the temp sensor back into the box	The computer application should now continue to get data from the box and display it	Pass	
Observe the data being collected	The computer application should be updating the temperature about once every second	Pass	
Set the minimum temperature on the computer application to 20 ℃, include a valid phone number in the box for a phone number and then cool the sensor to below that temperature	The computer application should send a text message to the phone number given	Pass	
Set the maximum temperature on the computer application to 20 ℃, include a valid phone number in	The computer application should a send text message to the phone number given	Pass	

the box for a phone number and then warm the sensor to above that temperature			
Change the desired message to be sent and then get the temperature above or below one of the min/max thresholds	The text message should now match the new message	Pass	
Turn the switch on the box to the off position	The box should power down and no longer do anything when the button is pushed	Pass	
Observe the computer application	The computer application should continue to update the temperature data	Fail	

4. Project Retrospective

a. Project Outcome

Overall, we stuck to our original plan well. From our first meeting to the final checkoff, we stayed on track and were able to complete the lab at a good pace. Looking at our design process, we were able to stay on point with most of our decisions. We think the only change we would add to our device would be a better design for the software component. We had a small delay in our readings and believe if we fixed the software design piece by using threads, our delay on the display would have been less. This design error could have been avoided if we spent more time on the software design portion of the lab. This would have provided a strong backbone for our prototype and would have kept us more organized. However, we think the end product we created was solid and we created a device that could be used everyday.

b. Changes for Future Labs

For future labs, we would like to change a few factors which will make our final project a more efficient device. In this lab, the one issue we would like to fix for next time is making sure we implement a better design for the delay on the temperature reading. We were able to get a reading on the display, however, the delay was longer than we wanted. In our software, we used threads to check if the button was pressed and this caused a minor timing delay on our display. We think if we did not use the threads and implemented the software of the button better the

delay for the temperature reading would have been much less than 20 ms. Another change we would make is making sure to read every requirement thoroughly. The one part we missed was that if the raspberry pi was shut down the graph would still read data. Our design failed for this test, because we used the raspberry pi and a computer with a server-client relationship and when the raspberry pi was shut down the data was not picked up from the pi. To fix this, we would need a different design choice and make sure to have software that would connect directly to the display so even when it shuts down the data is still received. These two changes are small and if we implement them for the next lab, we will be even more successful than we were with this one.

c. Roles and Responsibility

Throughout the lab, we all contributed a lot to the final outcome of the project. We were able to split the work and get the majority of the lab done together in our allotted time slots every week. As a team we met every week twice a week for about 2-3 hours in order to accomplish all of the necessary components of the lab. The first meeting for the lab was primarily planning; we set up a document to make sure we stayed on track and what we wanted to accomplish each week. We think this is what set us up for success, meeting early and getting a plan in place for the weeks to come. As for the roles and responsibilities of the group, Matt was in charge of most of the hardware components of the lab. He was able to get us started with the raspberry pi and temperature sensor to get us off to a quick start. Once we had the temperature sensor wired correctly, we were able to get started on the software to make sure the sensor was giving us data that we could work with. In one of our first meetings we were able to accomplish the simple code to make sure the sensor was providing the information we needed. Following this, we hooked up the LCD and we worked together on getting the display to work properly. Most of the software components of the lab were completed as a group during our meetings. Tim implemented the software behind implementing the text message. With the GUI component of the lab, we split that up between the three of us and we all worked on a separate part of the lab. Tim was in charge of the graph, Austin did the work with the button, and Makenna was in charge of putting it all together and making sure it all flowed together. The teamwork shown above is the reason we were able to get the lab done at an efficient pace.

The lab provided us with a few small challenges as a team. We believe the key to our success was starting early on the tasks for the lab. When we came

across an issue or challenge i.e, getting the temperature to display on the LCD, we worked together to find the best way to solve the error. With multiple people working on the one issue we were able to figure out the problem quickly. We realized that this was a lot of work and we would have to make progress each week so we had to use our strengths to our advantage. Matt being the main electrical engineer helped with the hardware components and then with the software pieces we shared the load during our meetings. We think there were not a ton of negative components of the lab as we worked well together for the past few weeks. Overall, we think we allotted the right amount of time to get the project done to the best of our abilities and we should continue to do this for the next few labs.

d. Project Management

Throughout the lab we implemented a waterfall and test driven methodology to complete most of the lab. This was because we completed most of the lab in phases and we needed to make sure all the requirements in each phase were met correctly. Phase one consisted of the hardware implementation. We knew we needed to complete this part first to then be able to implement the software. Also in phase one we completed a series of tests to make sure the hardware components were correctly working. This is when the test driven methodology was implemented. Once we knew all the hardware was working, we started phase two which was the bulk of the lab which involved the software. We implemented a server-client relationship between a computer and the raspberry pi and had to perform a series of tests to make sure the server and client communicated to each other correctly. Phase three was started after this, which consisted of the GUI that outputted the graph to the user and had inputs for the temperature and the minimum and maximum temperatures. This is the part that we had to key in on the major software design requirements. We had to include multiple different things on the graph and make sure the inputs were correctly implemented. This part of the process took a ton of time as we had to design the GUI to be user friendly and visually appealing. The tests for this part of the lab were specific and we had to make sure we were interfacing the GUI and the server-client in order to get the correct outputs. Phase four was a vital part of the lab which was a final run to make sure all the requirements were met and the final device was working properly. In this phase, we ran all necessary tests to make sure nothing broke the program and the device still worked. This phase was the biggest part of our plan and we would have missed a few requirements had we not completed this part.

e. Gantt Chart

Attached below is the Gantt chart we created on the first meeting we had as a group. We believe we stuck to this chart well and it was a major reason for our success in this lab. We decided to split the tasks up and we completed the components of the lab weekly in order to stay on track. We fell a little behind in the software for the GUI interface because we were trying to figure out the layout we wanted for the lab and it took us a little to agree upon the correct layout for the graph and user inputs. However, besides this the lab was split up well and assigning roles for the requirements was key. Another minor piece of the lab that could have gone better was spending more time on the design of the software. We felt that even though we knew the outcome we needed to slow down in the design portion so we could be more efficient with our testing and final product. This means that we should have spent more time in the planning part and then also adding a step in the Software Development that includes a better design component so we can stay more on top of the software. Besides these two minor points, we had great success and we want to carry this into future labs.

PROJECT TITLE	Lab 1: Digital Temperature Sensor	CLASS NAME	ECE: Principles of Electrical And Computer Engineering Design
TEAM NAME	Broken Engineers	DUE DATE	10/3/2021

							Week 1							Week 2							Week 3							Week 4					
	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	M	T	W	R	F	S	Su	M	T	W	R	F	S	S	M	T	W	R	F	S	Su	M	T	W	R	F	S
1	Project Conception and Initiation																																	
1.1	Project Planning	Everyone	9/1/21	9/30/21	29	100%			25%							25%							25%								25%			
1.1.1	Agenda	Makenna	9/1/21	9/8/21	7	100%			50%				20%		15%	5%																		
1.2	Research: Hardware	Matt	9/1/21	9/8/21	7	100%							85%	10%		5%																		
1.3	Research: Software	Tim	9/1/21	9/8/21	7	100%						30%		50%	15%	5%																		
1.4	Project Requirements	Austin	9/1/21	9/8/21	7	100%			75%				20%			5%																		
1.5	Project Initiation	Everyone	9/1/21	9/30/21	29	100%				25%							25%								25%							25%		
2	Hardware & Software Development																																	
2.1	Hardware Components & Circuit Building	Matt	9/7/21	9/26/21	19	100%										65%						20%								10%	5%			
2.2	Software: Server/ Client	Tim	9/10/21	9/14/21	4	100%										50%																		
2.3	Software: GUI	Makenna	9/23/21	9/27/21	4	100%				25%				25%							25%								25%					
2.3	Software: Graph	Tim	9/23/21	9/27/21	4	100%															50%								50%					
2.4	Software and Hardware Integration	Austin	9/15/21	9/27/21	12	100%																	25%						75%					
3	Test Planning																																	
3.1	Button Testing: Hardware	Matt	9/7/21	9/7/21	0	100%			50%		50%																							
3.1.1	Button Testing: Software	Tim, Austin, Makenna	9/7/21	9/9/21	2	100%					50%					50%																		
3.2	LCD Testing: Hardware	Matt & Austin	9/14/21	9/27/21	13	100%										50%		50%																
3.2.1	LCD Testing: Software	Tim & Makenna	9/14/21	9/27/21	13	100%												50%			50%													
3.3	GUI Interface Testing	Tim & Makenna	9/20/21	9/27/21	7	100%														50%						50%								
3.4	Server/Client Testing	Tim	9/16/21	9/27/21	11	100%										25%							50%								25%			
3.5	Software and Hardware Integration	Tim & Austin	9/16/21	9/27/21	11	100%										25%							50%									25%		
4	Documentation																																	
4.1	Design Documentation	Matt	9/29/21	10/3/21	4	100%																									50%	50%		
4.2	Design Process and Experimentation	Austin	9/29/21	10/3/21	4	100%																									50%			50%
4.3	Project Retrospective	Makenna	9/29/21	10/3/21	4	100%																									50%			50%
4.4	Test Plan	Tim	9/29/21	10/3/21	4	100%																									50%			50%
4.5	Final Overview	Everyone	9/29/21	10/3/21	4	100%																									50%			50%

5. Appendix & References

A. Circuits

This section will explain in further detail the multiple circuits that are apart of the final design of the temperature sensor.

1. Temperature Probe Circuit

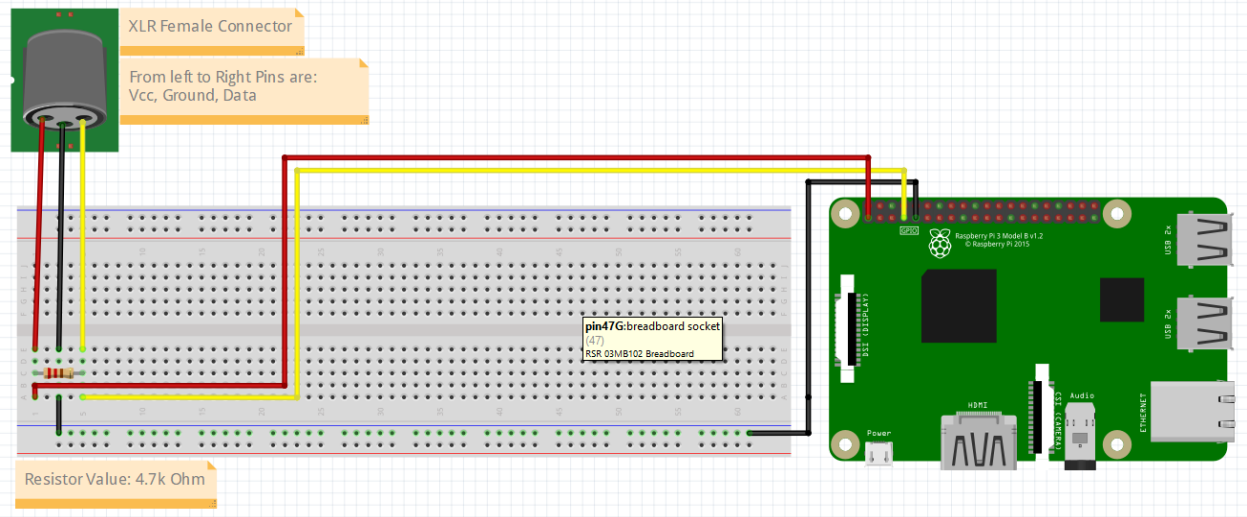


Figure A-1: Temperature Probe Circuit Diagram

The temperature probe circuit has 3.3V of power running from the 3.3V GPIO pin on the raspberry pi to the breadboard. There is no need to connect this voltage to the positive rail of the breadboard as it is the only component that uses the 3.3V source from the raspberry pi. Next, the wire from the 3.3V GPIO pin is connected to the Vcc wire of the female XLR connector for the temperature probe. The second connection is the ground connection coming from the ground GPIO pin on the raspberry pi and this runs from there to the negative rail on the breadboard. The third and final connection is the data connection which plugs into GPIO-04 on the raspberry pi. As can be seen on **Figure A-1** there is a 4.7k Ohm pull up resistor between the Vcc power and data connections. This resistor is necessary so that the DS18B20 can set the signal high and low to transmit temperature data. This works together with the one wire interface so that there is potential for multiple sensors to use the same wire.

2. Toggle Switch and Momentary Push Button

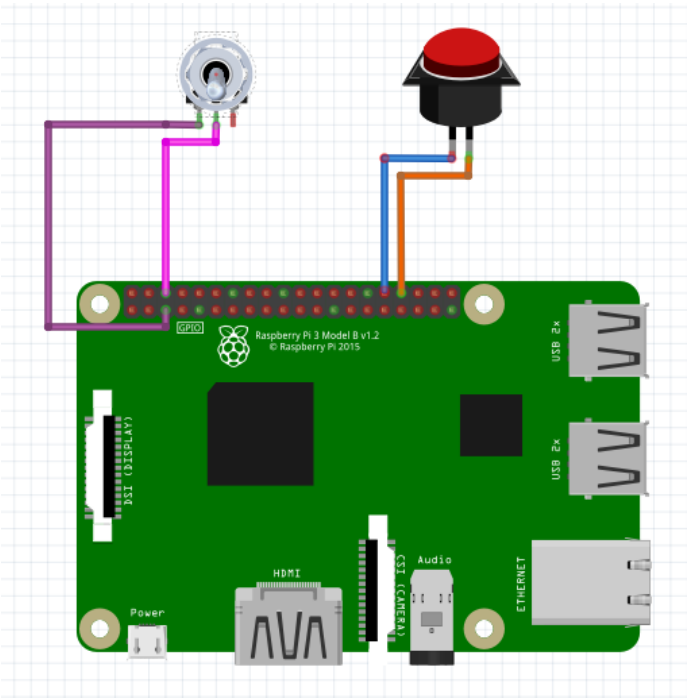


Figure A-2: Toggle Switch and Momentary Push Button Circuit Diagram

The toggle switch and push button are directly connected to raspberry pi via jumper wires. The button is a momentary push button and is normally open. The button when pushed by the user will close the circuit and the circuit will remain closed for the duration that the user is holding down the button. Once the user lets go of the button the circuit is again open and its close function will end. In this design when the button is held down the temperature data will display to the LCD screen if the temperature probe is plugged in or it will display “No Data” if the temperature probe is plugged in. The toggle switch controls the on/off state of the entire device. It is completely controlled by software and the controls are defined in the shutdown.py script that runs at the startup of the device. The corresponding GPIO pinouts for each of these components can be seen in **Table 1**.

3. LCD Circuit with Contrast Adjustment and MOSFET controlled Backlight

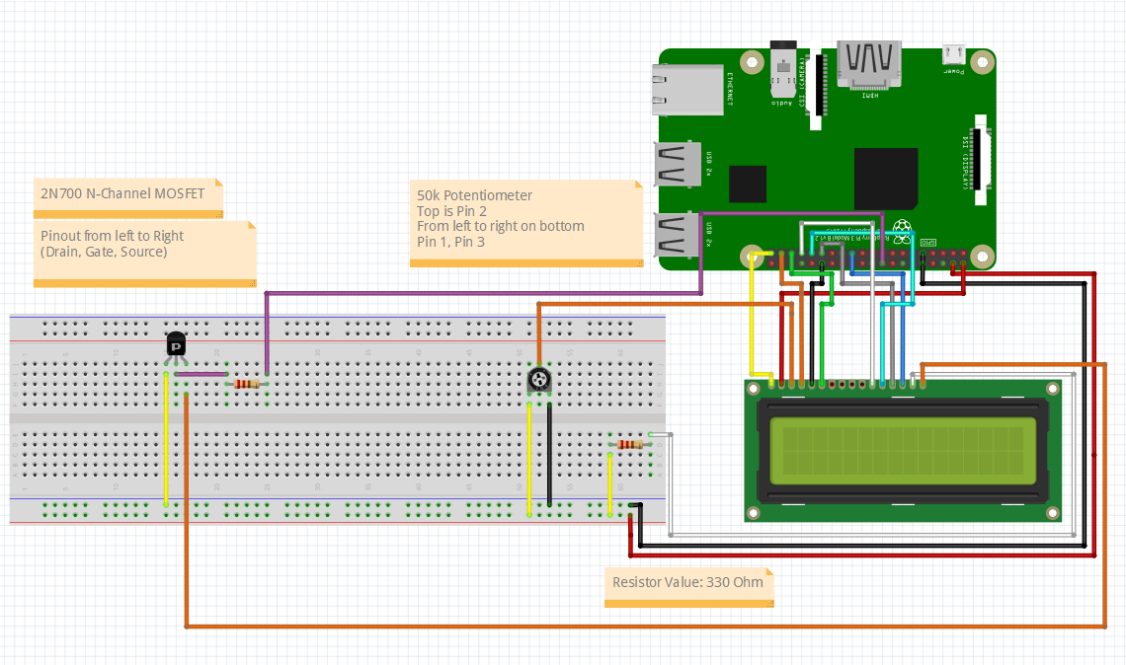


Figure A-3: LCD Circuit Diagram

In the circuit containing the LCD screen there are two minor circuits contained within the overall design. The first one being the contrast adjustment circuit. This circuit is composed of a 50k Ohm potentiometer connected to the contrast pin of the LCD. The 1st and 3rd pins are then connected to the 5V power rail and ground respectively. The potentiometer varies the resistance therefore varying the current to the contrast pin allowing the user to adjust the contrast of the LCD screen. The screen if not adjusted properly can be very difficult to see so setting this contrast to the perfect clarity is important for the temperature readings. The second circuit in this diagram is the MOSFET which controls the LCD backlight. When plugged into the raspberry pi with power being delivered the LCD backlight will always be on and there is no way through software alone to turn it on and off. A solution to this is to add a MOSFET into the design connected to the cathode or pin 16 of the LCD. This pin is connected to the drain of the MOSFET and the source is connected to ground via the negative rail on the breadboard. The gate of the MOSFET is connected to a GPIO pin on the raspberry pi with a 1k Ohm resistor in between so as to not exceed the MOSFETs gate conditions. Through software when the user pushes the button this sends a signal for the GPIO pin connected to the gate to go from low to high outputting a voltage and opening the MOSFET so that current can flow from drain to source allowing the backlight to come on. Once the user lets go of the button the GPIO pin is set from high to low the MOSFET gate closes and the backlight is turned off.

B. Datasheets

This section lists the data sheets for some of the components that have them available online.

DS18B20 Temperature Probe:
https://www.quick-teck.co.uk/Management/EEUploadFile/1420644897.pdf

2N7000 N-Channel MOSFET
https://www.onsemi.com/pdf/datasheet/nds7002a-d.pdf

LCD 1602
https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf