Digital Thermometer

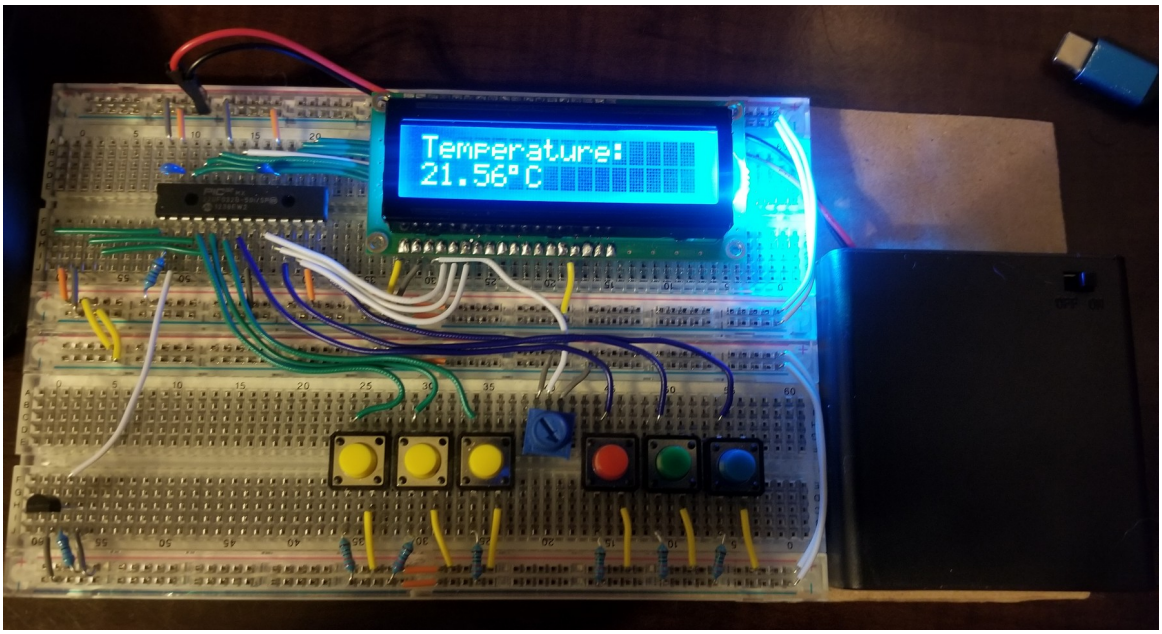
Introduction
This report goes over what this digital thermometer can do. It list in detail how it works and what it does by providing descriptions, and images of to help understand these concepts. It discusses the process of making a digital thermometer as well as understand how to use it. It also covers how to use and test a microcontroller. The important components for this project are PIC32MX microcontroller, DS18S20 temperature sensor, and a LCD.

Context
This project was made to create a simple digital thermometer using a PIC32MX microcontroller, a DS18S20 temperature sensor, and a LCD. This project is able to accuratley get the temperature of a room or even outside. The project deals with communication to the sensor, communication to the LCD, and communication to the microcontroller. The sensor is able to read from about -55°C to 125°C and display that value on the LCD as a Celsuis value, as a Farenheit value, and as a Kelvin value. The project is also able to change and mix the colors of the LCD to provide a nice display. The reason to create this project is to read the temperature and display it in a nice format. It allows us to bring a small and portable themometer. It also was used to learn and better understand circuits, components, and how to code so that we can control a microcontroller.

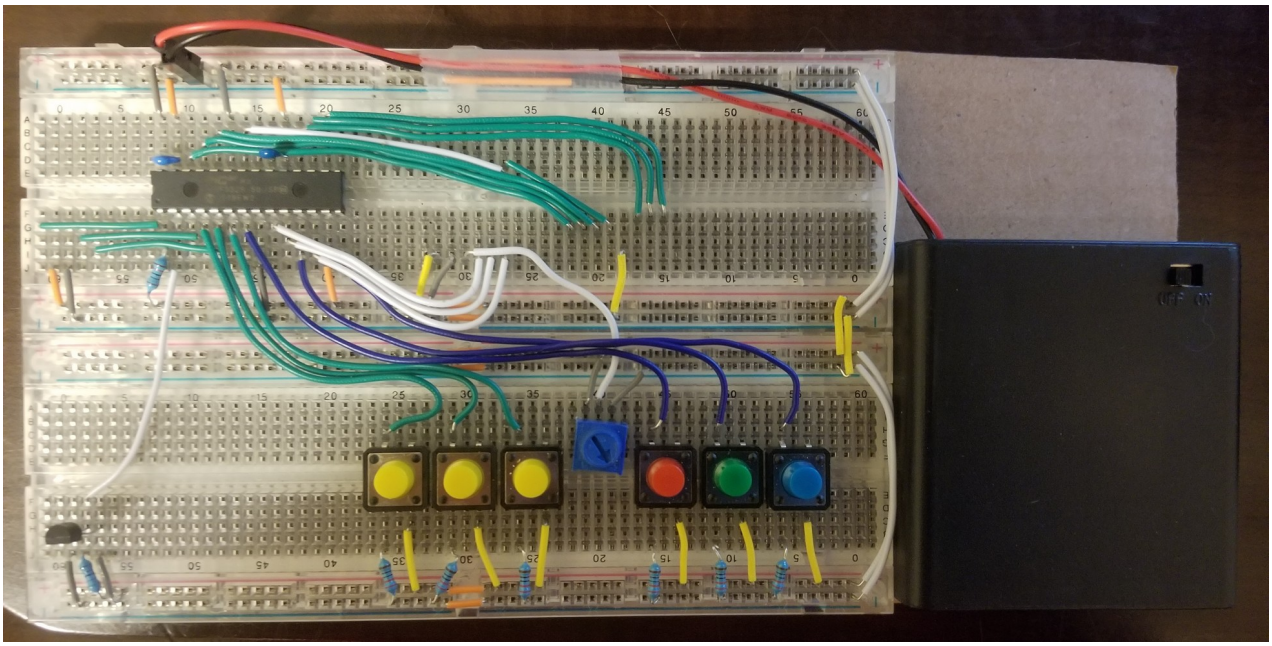
Technical Requirments / Specifications
- Commnuicate to a DS18S20 temperature sensor using one wire
- Communicate with an LCD
- Display the temperature to a LCD
- Accuratley get the temperature and have a high precision.
- Be able to change how it displays to the LCD
- Be able to to change and mix colors on the LCD
- Display to the LCD in different ways
- Know when the temperatuer sensor is not connected
- Convert between differt units
- Change precision while running
- Easy to use, change, access and understand
- Use a microcontroller to handle the communications between devices
- Portable and light
- Easy to modify and reproduce

Components List

2 * Breadboards (Anything to connect your components)
Any amount of wires or jumper cables
1 * PIC32MX220F032B (I use SPDIP for breadboard)
1 * PICKIT 3 programmer (Any programmer for the PIC32MX can work, but I use the PICKIT3 here)
1 * DS18S20 High-Precision 1-Wire Digital Thermometer
1 * NHD-0216K1Z-NS(RGB) LCD (Any lcd may do, but this contains different backlight colors)
1 * Battery Pack for about 4.5 V
3 * AA Batteries
1 * Potentiometer
6 * Push buttons
1 * 5KΩ resistor
1 * 10KΩ resistor
6 * 220Ω resistor
1 * 0.1UF capacitor
1 * 10UF capacitor



Finished circuit and final product showing a 9 precision temperature (sensor bottom left)



Cicuit under the LCD

Procedure

Getting started

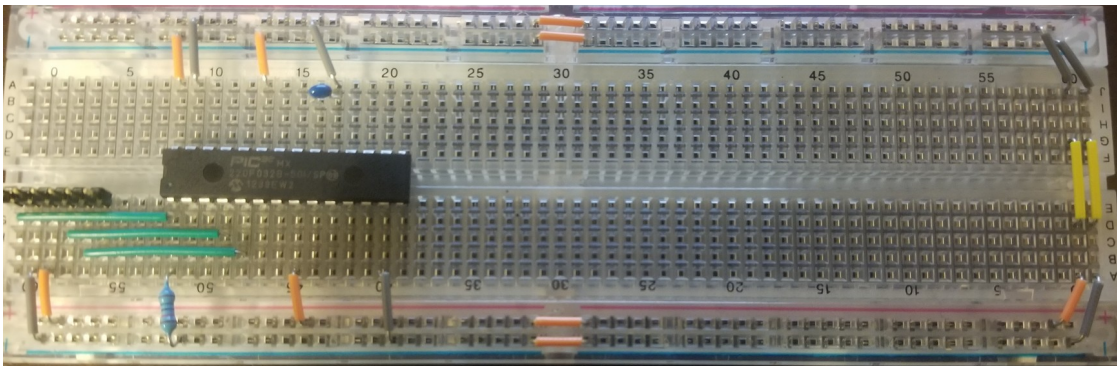
Begin by researching the required components for this project. Proceed to order all required components when properly researched. While waiting for all components, read and take note of all important details in the datasheets for each component. It is important that the datasheet is read and understood for the **LCD** and the **Temperature sensor**. The datasheets will provide a lot of information on how to properly communicate with the device, they will include commands, timings, tips, and techniques to use them. It is also important to read certain parts of the **PICKIT 3** and **PIC32MX** data sheets. Once all parts have arrived proceed to start by using one breadboard.

Circuit for the PICKIT 3

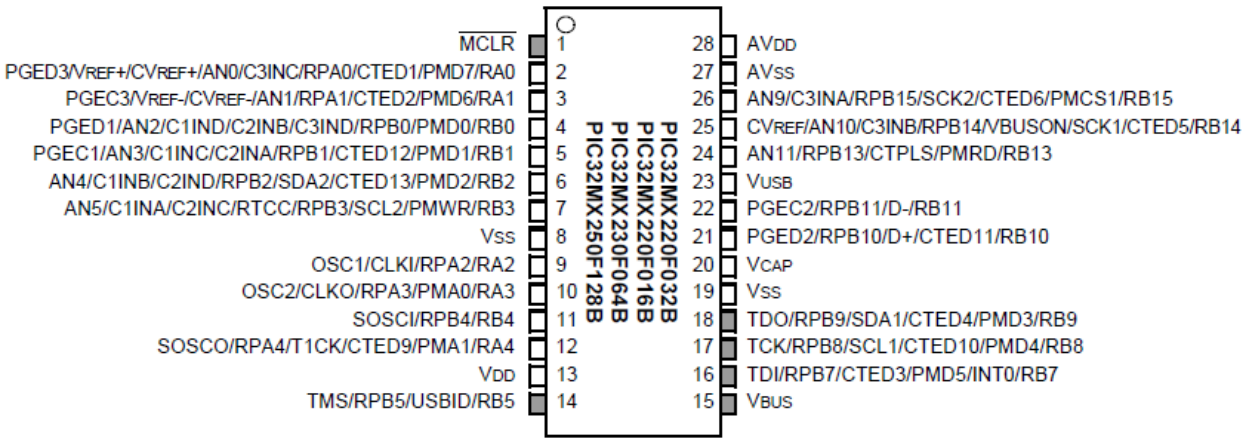
Begin the circuit with three wires of the same length. Place the first wire at the very edge of the breadboard (anywhere will do but make sure to provide room for other components). Place the second wire below the first wire such that there are two empty rows from where the first wire was placed. Place the third wire below the second wire such that it is one row forward from wher the second wire was placed. Where the first wire ends, place one 10KΩ resistor to the positve (+) terminal. Next place 2 wires between the two empty rows, between the first and second wires. The wire closest to the first wire must connect to the positive (+) terminal of the breadboard, and the second must connect to the negative (-) terminal of the breadboard. Now at the other end of the breadboard, have two wires connect both sides of the breadboards to there respective parts. This concludes the start of the circuit for the programmer.

Circuit for the PIC32MX

Place the PIC32MX where the first wire and the 10KΩ resistor are, such that PIN1 (MCLR) is connected. Place one 10UF capacitor between PIN20 (VCAP) and PIN19 (VSS). On PIN19 (VSS) place one wire to the negaitve (-) terminal. On PIN23 (VUSB) place one wire to the positive (+) terminal. It is recommended that to place one 0.1UF capacitor between PIN28 (AVDD) and PIN27 (AVSS). On PIN28 place one wire to the positive (+) terminal, and another wire on PIN27 to the negative (-) terminal. On PIN8 (VSS) place one wire to the negative (-) terminal. On PIN13 (VDD) place one wire to the positive (+) terminal. This concludes the beginning circuit for the PIC32MX. This is based off of this **website**.



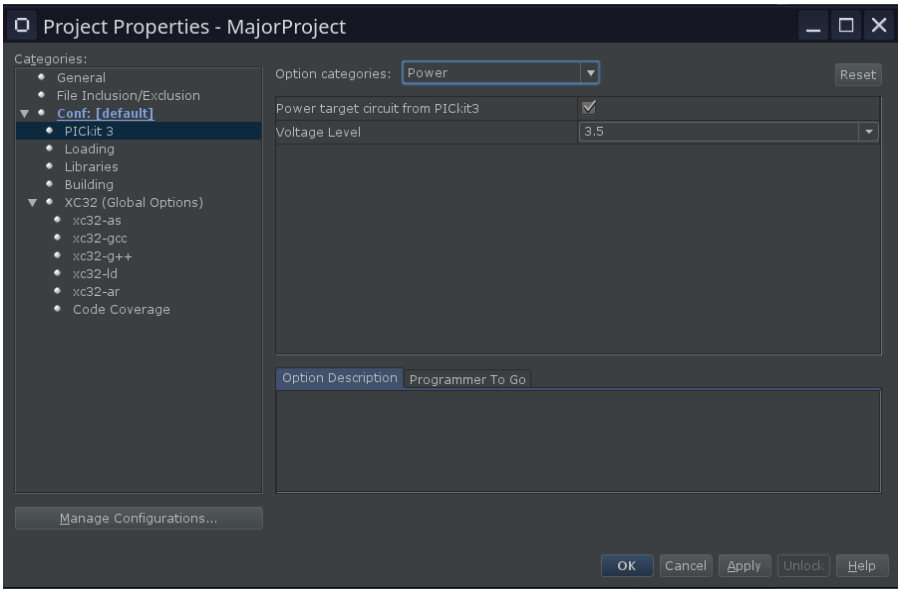
Starting circuit, 0.1UF capacitor not shown



Pin layout for the PIC32MX

Setting up the PICKIT 3

From here it is important to test that the programmer and circuit are working. Place the programmers pins into the programmer and keep an eye for the colored wire with the arrow, This Is your first programmer pin. Place the programmer in the breadboard such that the first pin is connected to the the first wire, and the sixth pin is connected to nothing. Connect the programmer to the computer that will assist in programming, it may light up first time. Next start up any programmer or ide that supports the PIC32MX and the PICKIT 3, for this project it is best to use MPLABX IDE. This procedure will not go over how to use MPLABX, there are many tutorial online that can go in depth. This will only cover key components to setting it up. The PICKIT 3 supports both powering the circuit using the programmer between 2.625 V and up to 3.5 V, or by an external power souce, but by using an external power source can be difficult and unreliable. For powering with the programmer using MPLABX, follow these steps. First create any standalone project for the PIC32MX that you are using. Next in project properties select PICKIT 3, click option catagories, select POWER, and check “Power Target”. You may leave the voltage level at defualt or change depending on how much is needed. For first time use you may need to wait as the PICKIT 3 downloads new firmware, if connected to the internet.If everything was done correctly it is possible to start programming, or debugging future projects!



Project properties options to provide power to the PICKIT 3

List of pins for components

It is finally time to get to the fun part of this project, connecting all components so that we can work and test with them! The code that will be made will be as general as possible so that any pins we decide to connect to may work, provided that the datasheet states those pins can be used for the desired purpose. For this project it has been decided that the folowing pins on the PIC32MX would be used:

Temperature sensor

PIN2 (RA0) → DS18S20 temperature sensor. Alias name is DQ.

Buttons

PIN4 (RB0) → Button 1. Alias name is BUTTON1.
PIN5 (RB1) → Button 2. Alias name is BUTTON2.
PIN6 (RB2) → Button 3. Alias name is BUTTON3.
PIN7 (RB3) → Button 4. Alias name is BUTTON4.
PIN11 (RB4) → Button 5. Alias name is BUTTON5.
PIN14 (RB5) → Button 6. Alias name is BUTTON6.

LCD

PIN9 (RA2) → Register select. Alias name is RS
PIN10 (RA3) → Read/Write. Alias name is RW
PIN12 (RA4) → Enable. Alias name is E

****** Our LCD will be in 4 bit mode ******
PIN26 (RB15) → Data bus 4. Alias name is DB4
PIN25 (RB14) → Data bus 5. Alias name is DB5
PIN24 (RB13) → Data bus 6. Alias name is DB6
PIN22 (RB11) → Data bus 7. Alias name is DB7
****** Optional pins for backlight control ******
PIN18 (RB9) → Red backlight. Alias name is RED
PIN17 (RB8) → Green backlight. Alias name is GREEN
PIN16 (RB7) → Blue backlight. Alias name is BLUE

Connecting the LCD

In order to connect the LCD the following must be read to better understand what each pin on the LCD does. Most LCD's come in this form, so it should be simple to adapt to another LCD. Make sure to read the LCD datasheet for more information.

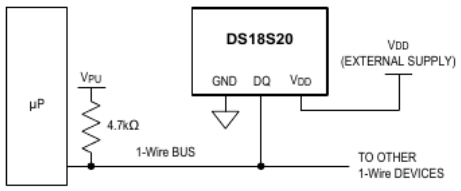
Pin No.	Symbol	External Connection	Function Description
1	V_{SS}	Power Supply	Ground
2	V_{DD}	Power Supply	Supply Voltage for Logic (+5.0V)
3	V_0	Adj. Power Supply	Supply Voltage for Contrast (approx. 0.5V)
4	RS	MPU	Register Select signal, RS=1: DATA RS=0: COMMAND
5	R/W	MPU	Read/Write select signal RW=1: READ, RW=0: WRITE
6	E	MPU	Operation Enable signal Falling Edge Triggered
7-10	DB0 – DB3	MPU	Four low order bi-directional three-state data bus lines. These four are not used during 4-bit operation
11-14	DB4 – DB7	MPU	Four high order bi-directional three-state data bus lines.
15	LED-	Power Supply	Backlight Cathode (Ground)
16	LED-RED	Power Supply	Backlight Anode Red (20 mA @ 2.1V)
17	LED-GREEN	Power Supply	Backlight Anode Green (20 mA @ 3.1V)
18	LED-BLUE	Power Supply	Backlight Anode Blue (20 mA @ 3.1V)

LCD pin descriptions. Color may not be available for all LCD's

Our external connection will be the PIC32MX. Start by connecting pin 1 of the LCD to the negative (-) terminal of the breadboard. Connect pin 2 to the positive (+) terminal of the breadboard. For pin 3 it is optional to place a potentiometer, if you do not use one, connect it to the positve (+) terminal. Connect a wire from pin4 to our named RS pin. Connect a wire from pin5 to our named RW pin. Connect a wire from pin6 to our named E pin. This project will only use 4 bit mode on the LCD, so pins 7 to 10 will not be connected. Connect pins 11 to 14 to there respective pins that were named. Connect pin15 to the negative (-) terminal. It is optional to connect pins 16 to 18, if not connect the colors desired to the positive (+) terminal. Some LCD come with pre-soldered header pins, unfortunatley our LCD does not. Some soldering will have to be done to receive proper and reliable connection. This will not cover how to solder, there are many tutorials and tips online on how to solder properly.

Connecting the DS18S20 temperature sensor

Connecting the DS18S20 is not too difficult. It only contains 3 pins, pin 1 being connected to the negative (-) terminal, pin 2 being our "One wire" for communaction between it and its master device is connected to our named DQ pin, and finally pin 3 being connected to our positive (+) terminal. It is very important that a resistor that of about 4.7KΩ is placed on pin 2 and connected to the positive terminal (+). This will act as our pullup resistor. This project will use a 5KΩ resistor. The DS18S20 does support a mode called parasite power, where we connect pin 1 and pin 3 to ground, but it requires even more specific timing, and will not be explained here. Do try to isolate the sensor from other devices so that those devices do not interfere or block it.



DS18S20 pin connections

Connecting the buttons

Connecting the buttons are very simple. Each one requires the same steps. Place the button between the middle gap of the breadboard, such that each side is connected to a seperate row of the breadboard, similar to the how the PIC32MX was connected. Connect the desired button to its respective pin on the PIC32MX. Place a 220Ω resistor (any small resistor may do) to the same pin that the button that we just connected and connect it to the negative (-) terminal. Finally connect the remaining pin to the positive (+) terminal. All buttons are connected like this. Do not that this project places these buttons on the second breadboard under the LCD for neatness.

Connecting a battery pack

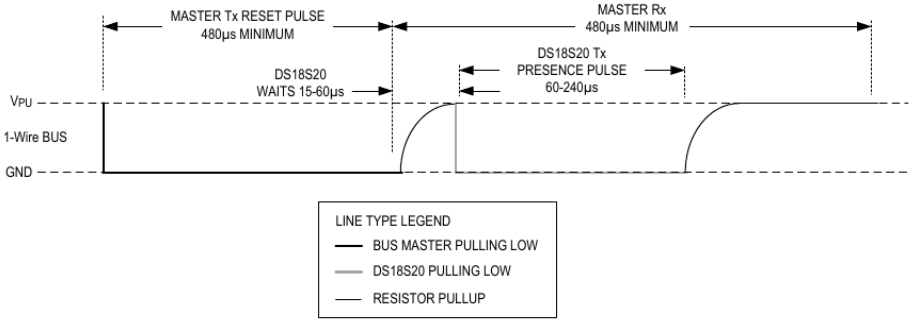
A battery pack is optional to connect. It allows the circuit to be moved around easier, so that it can be taken just about anywhere. In order to connect the batter pack, make sure that the requested voltage will not excede 5.5V as the PIC32MX will take a max of 5.5V. It is recommended to use way less then that, as the operating temperatures are between 2.3V to 3.6V. This project will use 3 AA batteries of 1.5V connected in series to create 4.5V. This voltage is a good one because it will not destory our PIC32MX and provides our LCD enough voltage to be visible easily. The batter pack must be connected to its respective positve (+) and negative(-) terminals to the breadboard. To avoid dangling it, place the project in a box, or attach it to anything that can hold both, such as a piece of sturdy cardboard or wood.

Communicating with the DS18S20 temperature sensor

Communicating with the DS18S20 temperature sensor comes down to two things, timing and order. The DS18S20 temperature sensor only allows us to communicate to it using one wire. To start commications we must always do the following steps (1) Initialization, (2) ROM Command (followed by any required data exchange), and (3) DS18S20 Function Command (followed by any required data exchange).

Initialization

We must beiging by performing an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse, that inidcites if the DS18S20 is ready to operate. The following picture shows the required timings as well as what the initialization sequence does.



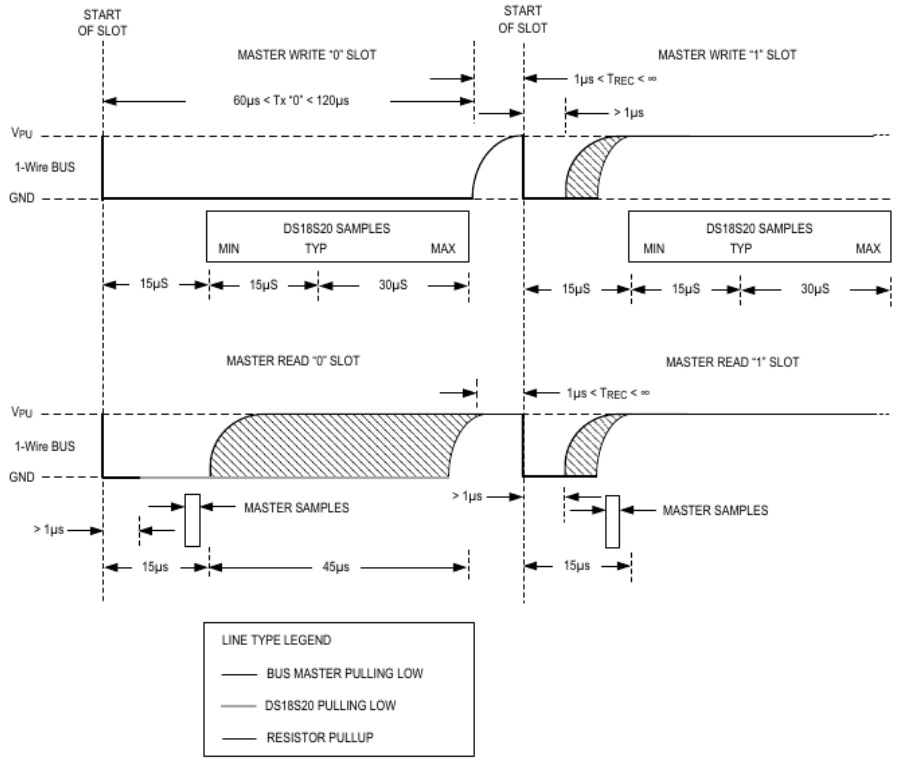
DS18S20 Initialization

ROM Command

The DS18S20 allows us to connect multiple 1-wire devices to a signle wire. Each one contains a 64-bit ROM code that we can use to determine which device we want to communicate with. These commands operate on the unique 64-bit ROM codes of each device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. This project will skip any ROM commands for simplicity and because we will only ever use one device. It is important to read the datasheet for each command, as they have a detailed description and correct timings.

DS182S0 Function Commands

In order to write commands to the we need to understand how writing and reading works. The bus master writes data to the DS18S20 during write time slots and reads data from the DS18S20 during read time slots. One bit of data is transmitted over the 1-Wire bus per time slot. Using the following image we can better understand how to write/read the DS182S0



DS18S20 write/read descriptions and timings

The timings must be followed or else communication will not work as desired. Now comes the differente commands. The following table from the datasheet describes each command and there hex code.

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED
TEMPERATURE CONVERSION COMMANDS			
Convert T	Initiates temperature conversion.	44h	DS18S20 transmits conversion status to master (not applicable for parasite-powered DS18S20s).
MEMORY COMMANDS			
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18S20 transmits up to 9 data bytes to master.
Write Scratchpad	Writes data into scratchpad bytes 2 and 3 (T_H and T_L).	4Eh	Master transmits 2 data bytes to DS18S20.
Copy Scratchpad	Copies T_H and T_L data from the scratchpad to EEPROM.	48h	None
Recall E^2	Recalls T_H and T_L data from EEPROM to the scratchpad.	B8h	DS18S20 transmits recall status to master.
Read Power Supply	Signals DS18S20 power supply mode to the master.	B4h	DS18S20 transmits supply status to master.

DS18S20 commands and descriptions

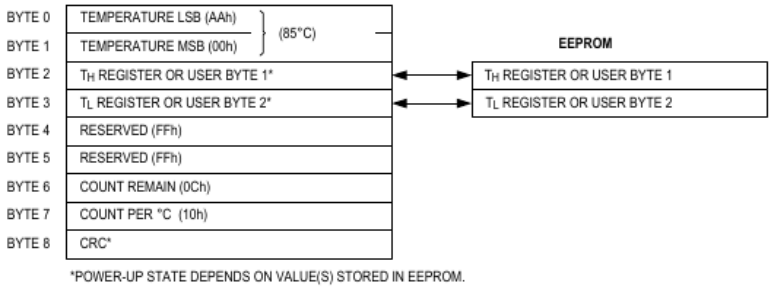
DS182S0 Getting the temperature

Now that we have our timings and order set we can now ask the sensor for our temperature! The DS18S20 has been designed to provide us the temperature, and convert it to °C. We can even get up to 12 bit precision. Each precision requires its own timings and how much each bit increases the temperature data.

Resolution	9 bit	10 bit	11 bit	12 bit
Conversion Time (ms)	93.75	187.5	375	750
LSB (°C)	0.5	0.25	0.125	0.0625

DS18S20 precision timings and detail

Now its time to start getting our temperature. We must first do our sequence, initialization, ROM, then a command. The command we will send is a Convert Temperature command (0x44). This command will convert the temperature for us, where each bit corresponds to 0.5°C steps. We must wait the required time based on the precision desired. Once the time is done we must do our sequence again and our command will be Recall E2 (0xB8). This command tells the DS18S20 to send its whole scratchpad containing 9 bytes. The bytes are described below.



DS18S20 scratchpad byte table

Byte 0 contains the temperature data, while Byte 1 only contains our sign bit. These are the two most important bytes that contain the information we need. To stop reading from scratchpad early we can send a reset pulse to stop the read time slot. To calculate the temperature we use the following formula.

$$Temperature = (\text{Byte0 shifted 1 bit to left}) - 0.25 + \frac{CountPerC - CountRemain}{CountPerC}$$

This will give us the proper temperature in °C, where we can convert it to other units if desired.

Communication with a LCD

In order to communicate to the LCD we must read the datasheet. Fortunatley for this project the datasheet provided contains semi ready C code to get started. For more information about LCD displays I will provide a useful **website**. This project uses a 4-bit mod that will need to a nybble of information (4 bits) twice, each time it needs to communicate. This is useful to know because in 8-bit mode we can send/recieve information all at once. The reason to use an 4-bit over a 8-bit is because it requires less wires, less pins used on our microcontroller, which can provide neatness and extra functionallity for other components. Most LCD's have the same process and commands in order to write/read to it. Everytime we wish to send information we must set our E pin to 1 or High. This tells the LCD to prepare information. The datasheet provides a list of commands in order to tell the LCD what to do. Writing letters to the LCD is not too difficult, all we really have to do is to tell it we will be sending data (RS high), we will be writing (RW low), then write to it, using the provided code in the datasheet. In C we can often time get away with directly writing array of characters, because most characters are built in by defualt.

11.2 Instruction Table

Instruction	Instruction code										Description	Execution time (fosc= 270 KHZ
	RS	R/W	DB7	DB6	DB 5	DB4	DB3	DB2	DB 1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRA and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" From AC and return cursor to Its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display	39us
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.	
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.	39us
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.	39us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39us
Read busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal Operation or not can be known By reading BF. The contents of Address counter can also be read.	0us
Write data to Address	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data From RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

The Code and final product

This project took many hours, and I am very glad that it is working and functional. Everything described has been written and performed to the best of my ability. The code can be found **here**. The way this project with is by the following. The Sensor reads the temperature and displays it to the LCD. We have 6 buttons that can different task. The first changes how the LCD shows data, the second changes precision of the temperature, the third changes our units (Celsuis, Farenheit, Kelvin). The last 3 buttons are dedicated for color, we can also mix colors here.

Testing

The way I tested that this system was properly working was by just trying to see what would cause it to stop working. For the temperature sensor to ensure that I was actually reading valid data, the data sheet says that CountPerC will always be hard coded to 0x10, and when debugging it always is that amount. That tells me that I am communicating with it correctly. For checking if the temperature is correct I did two things. One was downloading a temperature app online, and two was manually debuggin and plugging in random values, then converting using online calculators. After that I would attempt by either heating the sensor or cooling the sensor to test if it worked past room temperature, and lower then 0. Communicating with the LCD had a similar process. I would spend many hours trying to understand what was wrong and fixing them based on what I understood.

Contigency

I had a few ideas in mind, one included automatically cooling the device when heated, but the fans require too much money and require more power then I can safely provide. Another idea I had was making multiple smaller circuits and sending them to a server, but due to my limited knowledge on networking and servers I had to stop it for now. I have learned a lot of lessons while doing this. One includes how to properly use and understand a breadboard circuit and its components. Another includes learning how to solder on my own. I would love to apply my knowledge from this project on to future projects that I either do on my own of for a future school project.

Additional Material

I have done my best to make this project as open and easy to understand so that just about anyone can reproduce, test it and play around with. I think it is very important for projects like these are made open and easy to access so that it is easy for anyone from anywhere to learn from it and improve it. This project has been made a general as possible so that someone can use it in another solution if required. The product deals with the recording of temperatures, which has been becoming important as the temperatures around the world have been drastically changing and have been altered to the point that it is becoming harder to predict. This can hopefully be used to keep people safe indoors, and possibly outdoors with a proper case. It can also be modified to keep people at a decent temperature indoors, such that the temperature in a room does not excede anything that is seen as unsafe.

Conclusion

This project was a very fun project to do. I learned many things while doing it. Controlling the components seemed very daunting and hard, often times I was thinking why can I not figure it out, why is every explanation online only about arduino and other microcontrollers that are not like the pic32. It helped me better understand how to read datasheets and collect the information to make these components work. It truly hope just about anyone can reattempt this project and learn from it. I would really like for more of theses projects be made open as finding some info for certain components was difficult to find. Overall this was a very fun project and I would like to try another more difficult one soon!