Project 2 Final Report

This report showcases team 5’s entire design project to build an A/D converter using an Arduino the controller. It starts by giving a small introduction, background research, and then the technical requirements and restraints. Included in the background research is different types of A/D converters, then the report will discuss how we choose which converter to use based on our requirements. Next we reported the process we went through as well as the obstacles that we had to overcome. In the end of the report it details the building and testing portion of the project and an analysis our results from the demonstration.

Attachments: Picture of our Gantt chart/WBS, Equations, Circuit schematic, Flowchart, and Arduino code

**To:** Prof. Jie Yang, EE-286 Section 1

**From:** Team 5, EE-286 Section 1

Evan Genuise, Mason Gerace, William McGerty, Aidan Nash, and Joshua Pollock

**Date:** October 18, 2018

**Subject:** Final Report

**Introduction and problem statement:**

In this project we were tasked with creating an analog to digital converter. We were introduced to 3 separate techniques and circuits to accomplish our object at hand. The designs we were introduced to were: flash/parallel converter, ramp/sweep generator, and serial/successive approximation. The customer wanted this project to be tested by October 23rd, 2018 and completed by October 28th , 2018. This report details the requirements, the process of deciding, building and testing, the presentation, and how we used the design process. The appendixes will include the Gantt chart and Wbs charts, equations, circuit schematic, flowchart, and arduino code.

**Requirements and constraints:**

Requirements

* Visual two decimal display
  + This will improve readability of the measured voltage. This doesn’t limit the design, nor require a technical change.
* Be able to measure input voltage (0-10Vdc)
  + This will make sure our input voltage is correct, so this requirement doesn’t hinder the design or limit our possible solutions
* Accurate to ±2%
  + Because we can change the accuracy of our A/D converter, having a specific range of error helps our design over other designs, but does not affect how we create our solution.
* The test voltage measured by the Arduino will be displayed on a computer monitor.
  + This is more for “ease of use”, we will visually be able to see the test voltage via the command line
* The measured voltage derived from A/D circuit will be displayed by the Arduino.
  + This will show functionality, but does not limit and options.
* The percent difference between the voltage measured by the Arduino and the team’s circuit will be displayed on a computer screen.
  + This will help us show how accurate our circuit is, but doesn’t affect our solution.

* Constraints
* Will fit in a shoe box.
  + Sets the limit for space the project is allowed to take up.
* Must be controlled with an arduino
  + Limiting factor as other control methods cannot be used.

**Background research:**

Sweep/Ramp Generator:

The ramp generator was another all around good method. While it provides a lower speed, a ramp generator has the possibility of a very good resolution and only a medium cost. The circuit may require calibration and should be easy to build. A single power rail should work for this method as well instead of requiring positive and negative power rails.

Successive/Serial Approximation:

This ADC proved to be the most well-rounded method in our research. It has a medium speed, medium precision, and a medium cost to it.

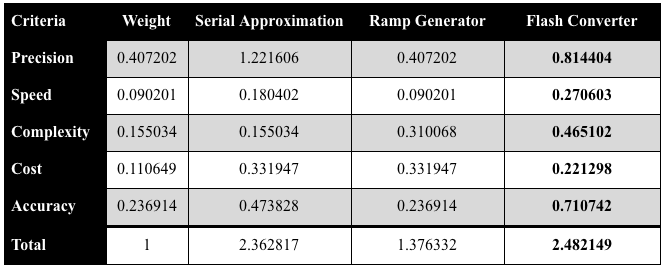
Flash/Parallel Converter:

A N-bit flash ADC has 2N-1 comparators and 2N resistors. Relatively easy to build but have a low resolution. These ADCs have a high speed but also have a high cost as they use many comparators and resistors as the circuits resolution gets higher.

**Pairwise and decision matrix**:



Here is the Pairwise matrix that we came up with. The criteria we used were precision, speed, complexity, cost, and accuracy.



The above decision matrix takes the values from the first decision matrix and multiplies those values by the weight that was calculated in the pairwise matrix. From this matrix, we concluded that the **Flash Converter method** would best satisfy the requirements and constraints for the circuit design. The flash converter won with a total of **2.482149**. This circuit design proved to be one of the most versatile methods given to us. While a little more costly than the other two, this ADC proved to be much more simple in concept than the other two.

**Discussion of op-amp choice:**

By using a decision matrix, our group decided that will be using the Flash converter as our ADC method. Through our research we found that the flash converter, although a little more costly, is very simple and can be semi-accurate. The flash converter was the best choice for our team and provided many benefits over the other methods that we researched. Our group is satisfied with this outcome, as the other methods of converting analog to digital used a clock and were a bit out of reach for us.

**Discussion of LED vs 7-segment display:**

We did not decide between these, and instead decided to just do both. We managed to wire it up so that they did not take too much power away from anyone one part, and in the end they were fully functioning. We came to this decision because we believed it would look better and overall be better if we had both options fully functioning and ready. In the event that a “client” were to not indicate whether they wanted one or the other, we felt it would be most appropriate to have both, and they could choose when it’s all working perfectly.

**Build and test:**

For the building portion, we first of course got the stack comparator designed first. We decided to go with the LM324, a 4 quad op amps (4 chips with 4 op amps in each), so that we could get a more confined board. To start, we first got the code (seen in Appendix V) and the first 10 op amps up and running to see if the first 4 bits/ digits would work correctly. After we did this, we moved on to get the bits after the decimal up and running, which ended up being 2 digits, and the rest of our op amps.

Once the op amps were up and running we moved to the LED’s and the LCD display, along with the 7 segment display. These were fairly simple to build as they were simple wiring and hooking up, and we used past code from the previous project. After that, it was just a matter of hooking them up the the arduino, and they were finished.

The code was mainly for the LCD and the 7 segment, as the stack did not need very much code. To get a read from the comparators we used a recursive method to iterate up an array of pins. This read if the pin was high and added 0.75 (the voltage step) to a total. If the recursive method detected a low pin, the method would return a 0 and stop recursion. For the LCD, we imported the LiquidCrystal library, and set the output to display “Voltage: “ plus whatever the output was. The contrast was set with a potentiometer. The bulk of the code was for the 7 seg, as it needed quite a lot to be able to function. We created two helper methods to output to the 7 segment display. Using a switch case statement, we checked which index the digit was at and then called the next helper method to display the correct number. This iterated through the 4 7 segments and outputted the correct digit as needed.

**Demo results:**

For the demo on Tuesday, October 23, we brought the fully functioning arduino in, with the presumably working code, and as we tried to test in class, the code would not upload to the board. As a result, we had to push the demo to the following Tuesday, October 30. Aside from the code not working, the rest of the board had been working and was receiving the correct output from a potentiometer input. The 7 seg and LED screen math with what the stack was outputting, so the only thing that needed to change was the code uploading so we could plug in a VRef and truly test it.

**Analysis of results:**

From what we currently understand, the mega simply did not want to take the code, but worked shortly after. We diagnosed the problem later to be the fact that the digital inputs were acting strange when using the serial pins, so we switched them to other digital pins and the code finally uploaded. After this, we started actually being able to test the board, and we ran into a new problem, which ended up being inputs switched with outputs. After this, the board passed all self ran tests, and is awaiting our follow up demo.

**Analysis of the project in terms of the design process:**

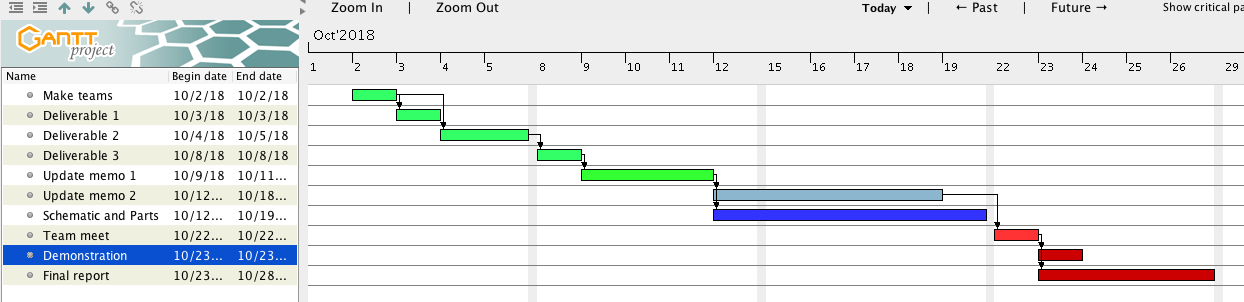
We were first asked to create an analog to digital converter. Guidelines were given to us that our device had to meet. We started brainstorming different methods we could use for this project, settling on a flash converter. We then took the guidelines given to us into consideration when creating the logic and circuitry for our product. Our product went through several prototypes before settling on our final design.

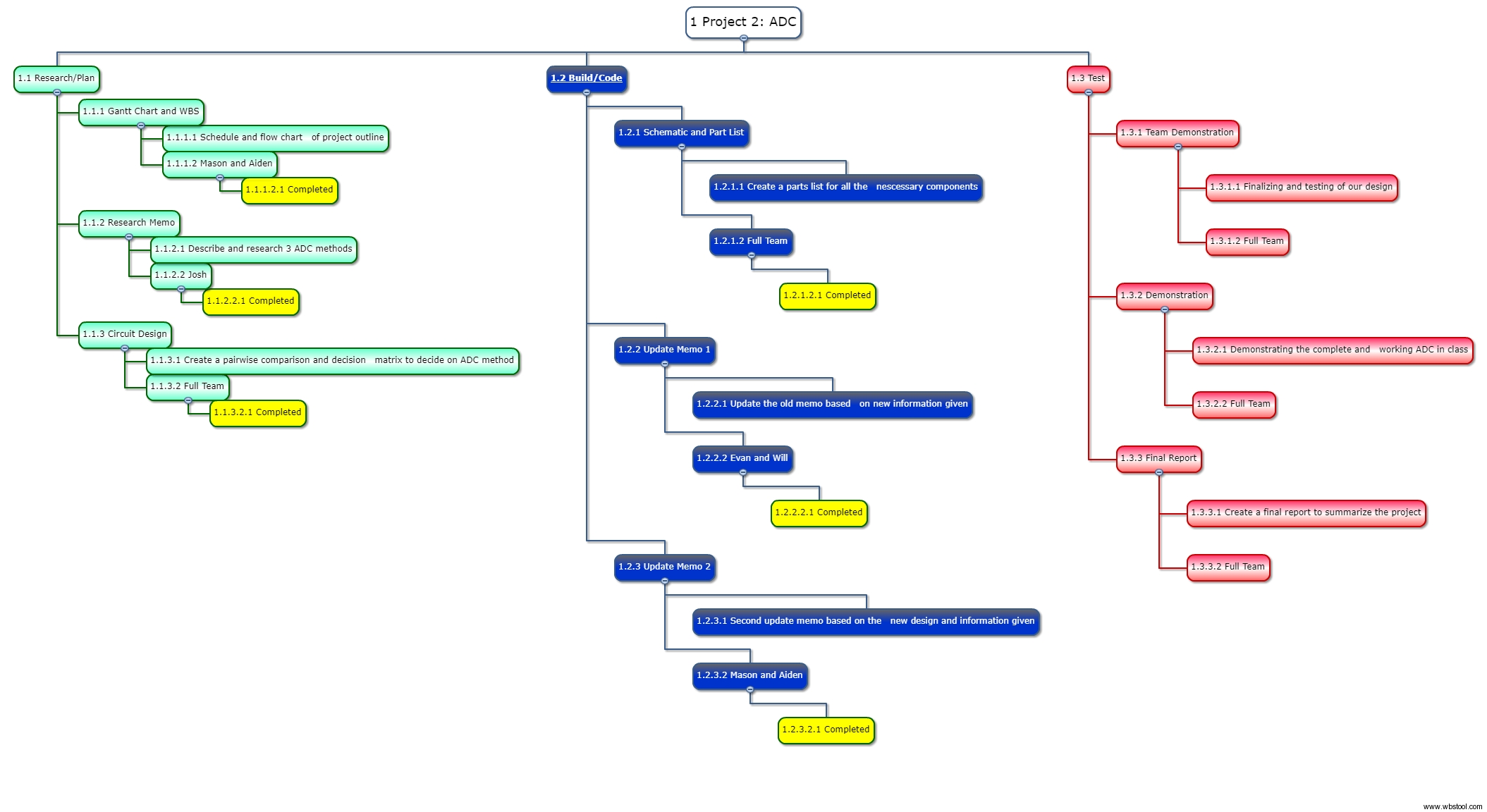
**Analysis of the team functions in terms of team development process:**

On the first day our project was assigned to us, we met one another to establish interpersonal relationships with one another and to discuss the task at hand. We also decided on who would be assigned to each task for the remainder of our project. We never reached step two of team formation, which arises conflict within the group. Each member knew what they were assigned and carried out what needed to be done individually in order to ensure that our group was successful. Throughout the entirety of the project, our group had clear roles and each member was responsible in completing the tasks assigned to them.

**Appendix A:**

Gantt chart, Wbs Chart





**Appendix B:**

Equations

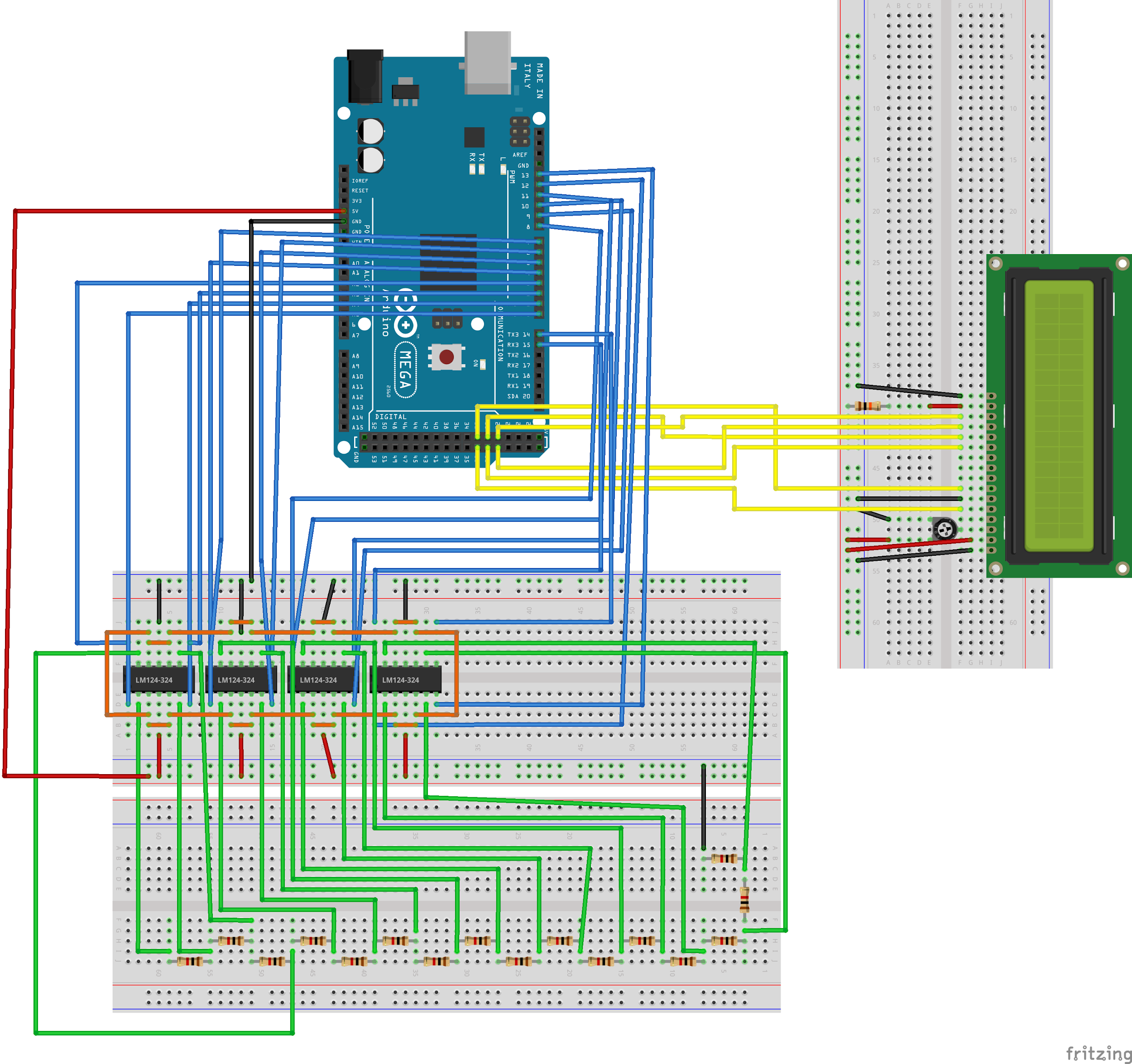
Voltage Divider Equation:

Math for each step:

1. 1000 \* (12/160000) = 00.75 V
2. 2000 \* (12/160000) = 01.50 V
3. 3000 \* (12/160000) = 02.25 V
4. 4000 \* (12/160000) = 03.00 V
5. 5000 \* (12/160000) = 03.75 V
6. 6000 \* (12/160000) = 04.50 V
7. 7000 \* (12/160000) = 05.25 V
8. 8000 \* (12/160000) = 06.00 V
9. 9000 \* (12/160000) = 06.75 V
10. 10000 \* (12/160000) = 07.50 V
11. 11000 \* (12/160000) = 08.25 V
12. 12000 \* (12/160000) = 09.00 V
13. 13000 \* (12/160000) = 09.75 V
14. 14000 \* (12/160000) = 10.50 V
15. 15000 \* (12/160000) = 11.25 V
16. 16000 \* (12/160000) = 12.00 V

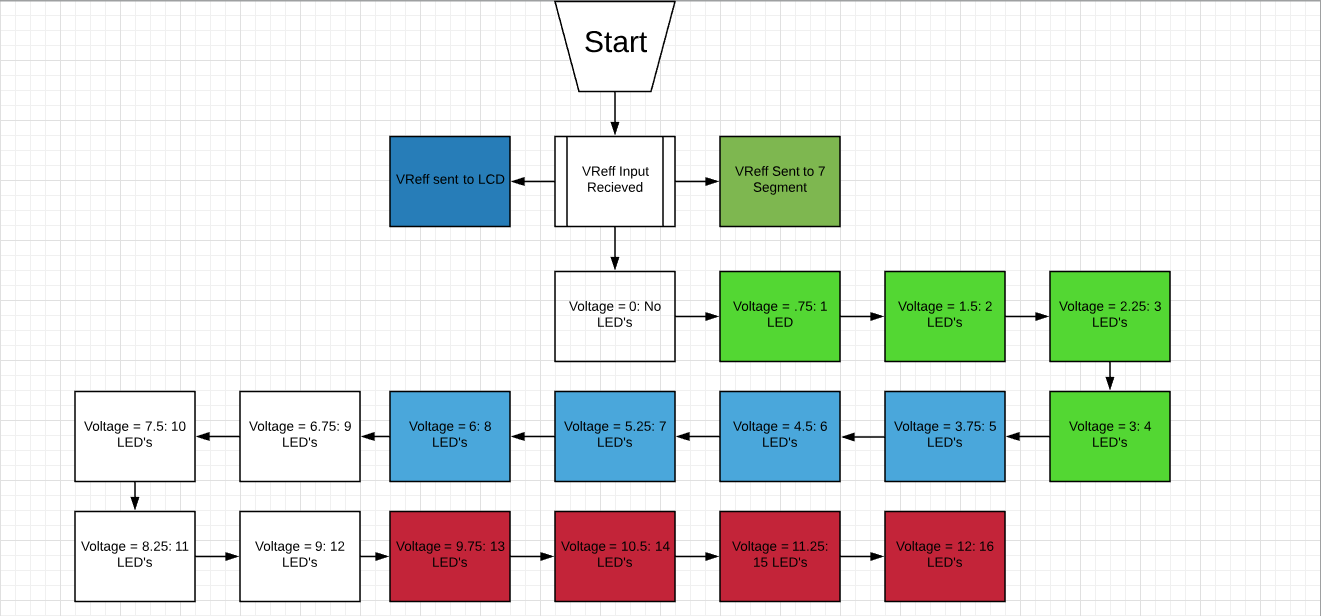
**Appendix C:**

Circuit schematic



**Appendix D:**

Flowchart

****

**Appendix E:**

Arduino code

#include <LiquidCrystal.h>

#define VCC 12

//array to hold our comparators

const int comparators[] = {38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53};

// Constant integers for 4 7-segment display

const int pinA = 22, pinB = 23, pinC =24, pinD = 25,

pinE = 26, pinF = 27, pinG = 28, pinDec = 29, D1 = 2,

D2 = 3, D3 = 4, D4 = 5;

// LCD Constants

const int rs = 35, en = 34, d4 = 33, d5 = 32, d6 = 31, d7 = 30;

const double VOLTAGE\_STEP = 0.75;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

String outVoltage;

void sevenSegmentHelper(char digit, int index);

void sevenSegmentNumbers(char digit);

double comparatorRead();

void setup()

{

//initialize comparator pins as inputs

for (int index = 0; index < 16; index++)

{

pinMode(comparators[index], INPUT);

}

pinMode(pinA, OUTPUT);

pinMode(pinB, OUTPUT);

pinMode(pinC, OUTPUT);

pinMode(pinD, OUTPUT);

pinMode(pinE, OUTPUT);

pinMode(pinF, OUTPUT);

pinMode(pinG, OUTPUT);

pinMode(pinDec, OUTPUT);

pinMode(D1, OUTPUT);

pinMode(D2, OUTPUT);

pinMode(D3, OUTPUT);

pinMode(D4, OUTPUT);

lcd.begin(16, 2);

Serial.begin(9600);

}

void loop()

{

int index;

char voltageChar;

// Get the whole number voltage

outVoltage = (String)comparatorRead(15);

// Adds a 0 to the front of the number if not 10, 11, or 12

if (outVoltage.charAt(1) == '.')

{

outVoltage = "0" + outVoltage;

}

// Iterate through outVoltage and use seven seg helpers

for (index = 0; index < 2; index++)

{

voltageChar = outVoltage.charAt(index);

if (voltageChar != '.')

{

sevenSegmentHelper(voltageChar, index);

}

}

for (index = 3; index < 5; index++)

{

voltageChar = outVoltage.charAt(index);

if (voltageChar != '.')

{

sevenSegmentHelper(voltageChar, index-1);

}

}

Serial.println("Voltage: " + outVoltage);

lcd.setCursor(0,0);

lcd.print("Voltage: " + outVoltage);

}

double comparatorRead(int index)

{

double calculatedVoltage;

if (index >= 0 && digitalRead(comparators[index]) == HIGH)

{

return (VOLTAGE\_STEP + comparatorRead(index-1));

}

return 0;

}

void sevenSegmentHelper(char digit, int index)

{

switch (index)

{

// If first digit

case 0:

digitalWrite(D1, LOW);

digitalWrite(D2, HIGH);

digitalWrite(D3, HIGH);

digitalWrite(D4, HIGH);

digitalWrite(pinDec, LOW);

sevenSegmentNumbers(digit);

break;

// If second digit

case 1:

digitalWrite(D1, HIGH);

digitalWrite(D2, LOW);

digitalWrite(D3, HIGH);

digitalWrite(D4, HIGH);

digitalWrite(pinDec, HIGH);

sevenSegmentNumbers(digit);

break;

// If third digit

case 2:

digitalWrite(D1, HIGH);

digitalWrite(D2, HIGH);

digitalWrite(D3, LOW);

digitalWrite(D4, HIGH);

digitalWrite(pinDec, LOW);

sevenSegmentNumbers(digit);

break;

// If fourth digit

case 3:

digitalWrite(D1, HIGH);

digitalWrite(D2, HIGH);

digitalWrite(D3, HIGH);

digitalWrite(D4, LOW);

digitalWrite(pinDec, LOW);

sevenSegmentNumbers(digit);

break;

}

}

void sevenSegmentNumbers(char digit)

{

switch (digit)

{

// if digit is 0

case '0':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, HIGH);

digitalWrite(pinF, HIGH);

digitalWrite(pinG, LOW);

delay(1);

break;

// if digit is 1

case '1':

digitalWrite(pinA, LOW);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, LOW);

digitalWrite(pinE, LOW);

digitalWrite(pinF, LOW);

digitalWrite(pinG, LOW);

delay(1);

break;

// if digit is 2

case '2':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, LOW);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, HIGH);

digitalWrite(pinF, LOW);

digitalWrite(pinG, HIGH);

delay(1);

break;

// if digit is 3

case '3':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, LOW);

digitalWrite(pinF, LOW);

digitalWrite(pinG, HIGH);

delay(1);

break;

// if digit is 4

case '4':

digitalWrite(pinA, LOW);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, LOW);

digitalWrite(pinE, LOW);

digitalWrite(pinF, HIGH);

digitalWrite(pinG, HIGH);

delay(1);

break;

// if digit is 5

case '5':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, LOW);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, LOW);

digitalWrite(pinF, HIGH);

digitalWrite(pinG, HIGH);

delay(1);

break;

// if digit is 6

case '6':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, LOW);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, HIGH);

digitalWrite(pinF, HIGH);

digitalWrite(pinG, HIGH);

delay(1);

break;

// if digit is 7

case '7':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, LOW);

digitalWrite(pinE, LOW);

digitalWrite(pinF, LOW);

digitalWrite(pinG, LOW);

delay(1);

break;

// if digit is 8

case '8':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, HIGH);

digitalWrite(pinF, HIGH);

digitalWrite(pinG, HIGH);

delay(1);

break;

// if digit is 9

case '9':

digitalWrite(pinA, HIGH);

digitalWrite(pinB, HIGH);

digitalWrite(pinC, HIGH);

digitalWrite(pinD, HIGH);

digitalWrite(pinE, LOW);

digitalWrite(pinF, HIGH);

digitalWrite(pinG, HIGH);

delay(1);

break;

}

}