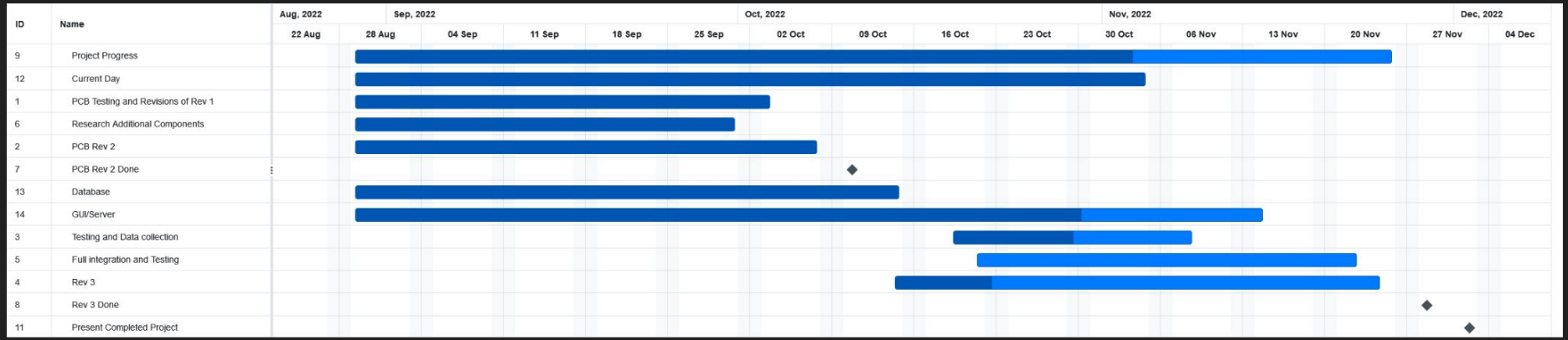
Security PUFfins
Design Presentation 3

Client: Julie H.
GTA: Jordan Beverly

Benjamin Assmann, Sharley Fabro, and Traigh Kirkeeng
4 November, 2022

Gantt Chart



Big Milestones Coming Up:

- Fixing ADC
- Revision 3
- Completed GUI
- Full testing

Project Overview

Circuit

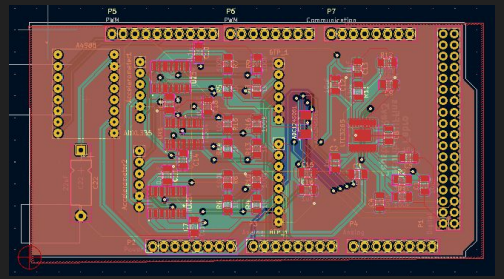
- 2nd Revision is working besides ADC
- Working on 3rd revision
- Test points are reading data
- Working on actually being able to execute our test process with real data

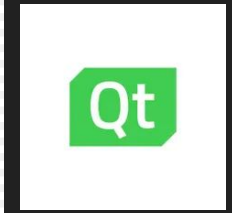
Arduino

- Scratch old ADC
- Learn new ADC
- Use SPI Arduino code to make circuit work
- Get real data from ADC to excel that isn't Analog

Database/GUI

- Wait for response from client about server deployment
- Fix GUI statistic functions
- Have group be comfortable with Documentation
- Help deploy ADC
- Get real data from Excel to Database

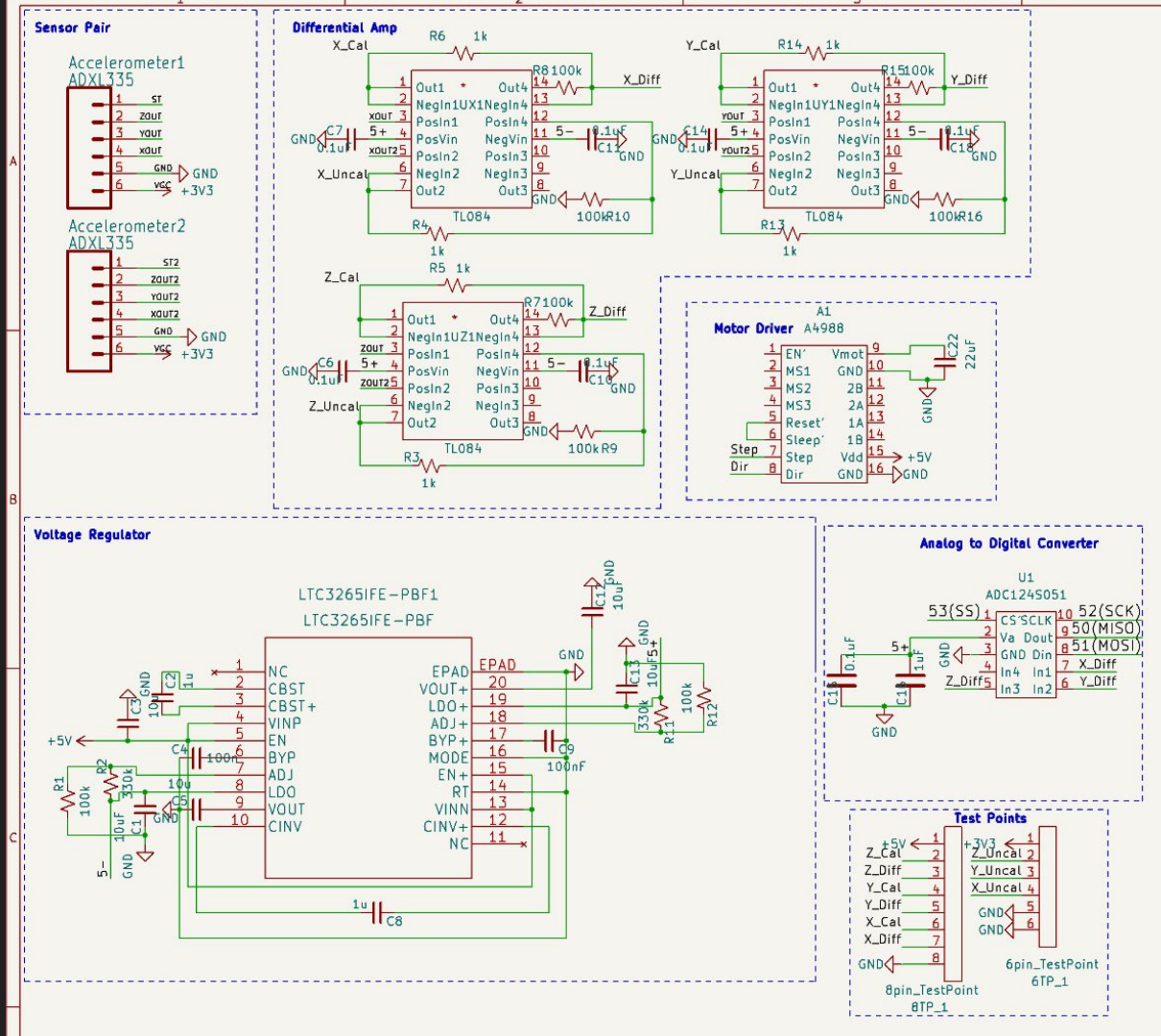




PCB

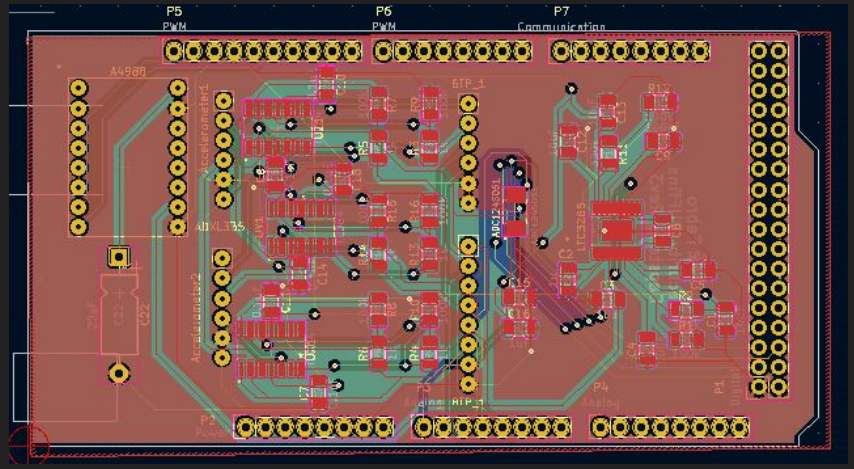
Modifications

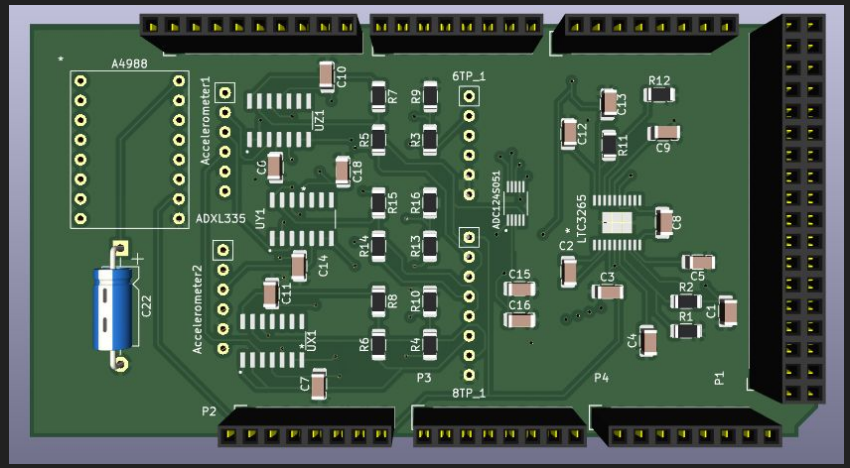
- Changed ADC to ADC124S051
- ADC is connected to SPI ports
- Fixed trimming capacitors



PCB

- New ADC is connected to SPI pins
- Trim capacitor is fixed to be grounded on one side
- Via is added to the charge pump ground to dissipate heat
- First layer grounding plane is expanded to cover the entire board





Data Collection

- Analog connections were utilized with the built in test points
- Differences between sensors are amplified with a gain of 100
- The values are relative close being within 1V difference

1	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.34	ZDiff= 1.35
2	YDiff= 1.72	YDiff= 1.72	YDiff= 1.72	YDiff= 1.72	YDiff= 1.67	YDiff= 1.69
3	XDiff= 2.08	XDiff= 2.09	XDiff= 2.09	XDiff= 2.09	XDiff= 1.78	XDiff= 1.83
4	XCal= 2.09	XCal= 2.09	XCal= 2.08	XCal= 2.09	XCal= 1.79	XCal= 1.84
5	YCal= 1.76	YCal= 1.76	YCal= 1.76	YCal= 1.76	YCal= 1.69	YCal= 1.71
6	ZCal= 1.71	ZCal= 1.71	ZCal= 1.71	ZCal= 1.71	ZCal= 1.36	ZCal= 1.37
7	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.34	ZDiff= 1.35
8	YDiff= 1.72	YDiff= 1.72	YDiff= 1.72	YDiff= 1.72	YDiff= 1.67	YDiff= 1.69
9	XDiff= 2.08	XDiff= 2.09	XDiff= 2.09	XDiff= 2.09	XDiff= 1.78	XDiff= 1.83
10	XCal= 2.09	XCal= 2.09	XCal= 2.08	XCal= 2.09	XCal= 1.79	XCal= 1.84
11	YCal= 1.76	YCal= 1.76	YCal= 1.76	YCal= 1.76	YCal= 1.69	YCal= 1.71
12	ZCal= 1.71	ZCal= 1.71	ZCal= 1.71	ZCal= 1.71	ZCal= 1.36	ZCal= 1.37
13	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.34	ZDiff= 1.35
14	YDiff= 1.72	YDiff= 1.72	YDiff= 1.72	YDiff= 1.72	YDiff= 1.67	YDiff= 1.69
15	XDiff= 2.08	XDiff= 2.09	XDiff= 2.09	XDiff= 2.09	XDiff= 1.78	XDiff= 1.83
16	XCal= 2.09	XCal= 2.09	XCal= 2.08	XCal= 2.09	XCal= 1.79	XCal= 1.84
17	YCal= 1.76	YCal= 1.76	YCal= 1.76	YCal= 1.76	YCal= 1.69	YCal= 1.71
18	ZCal= 1.71	ZCal= 1.71	ZCal= 1.71	ZCal= 1.71	ZCal= 1.36	ZCal= 1.37
19	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.69	ZDiff= 1.34	ZDiff= 1.35

Arduino and ADC

- Stepper Motor code has been completed
- Simple 40 step rotation, with stops to read values from accelerometers
- Currently reads values and develops PUFs, but not very precise

```
void loop()
{
  int steps;
  while (steps < 40) {
    if (steps == 0){
      delay(3000);} //wait 3 seconds. It gives enough time to read again once it comes back to the original position.

  // Set motor direction clockwise.
    digitalWrite(dirPin, HIGH);

  for(int x = 0; x < stepsPerRevolution; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }

  delay(1000); // Wait 1 second
  //reset the values before every read
  float ZDiff = 0.0;
  float YDiff = 0.0;
  float XDiff = 0.0;
  float ZCal = 0.0;
  float YCal = 0.0;
  float XCal = 0.0;

  for(i=0; i< 10; i++) {

  ZDiff = analogRead(A0);
  ZCal = analogRead(A1);
  YDiff = analogRead(A2);
  YCal = analogRead(A3);
  XDiff = analogRead(A4);
  XCal = analogRead(A5);

  //store 10 values for each axis
  ZDiff_values[i]= ZDiff;
  YDiff_values[i]= YDiff;
  XDiff_values[i]= XDiff;
  ZCal_values[i]= ZCal;
  YCal_values[i]= YCal;
  XCal_values[i]= XCal;
   }
```

Arduino and Old ADC

- Old ADC did not work
  - Timing too fast
  - Even in the default state, output was always 0
  - Attempted to use Serial Port Interface (SPI) but also did not work
  - Extremely complicated with internal control registers
- Changed ADC to a much slower and less complex ADC

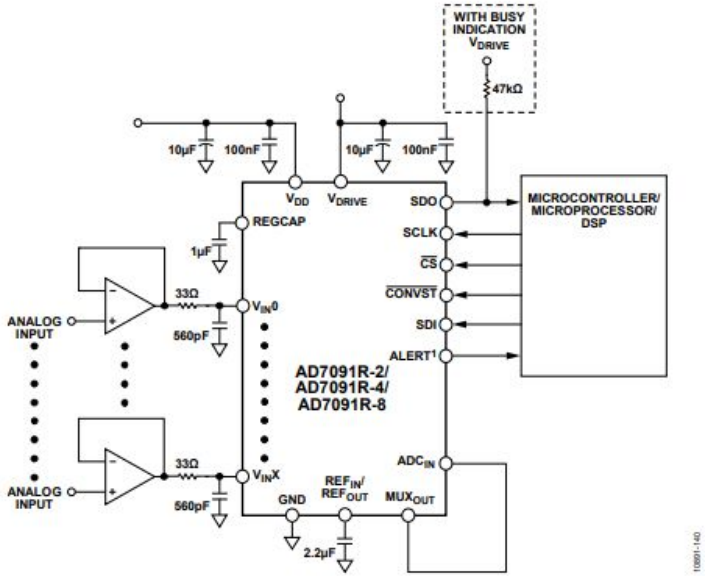


Table 10. Register Description

Address	Register Name	Default	Access		
			AD7091R-8	AD7091R-4	AD7091R-2
0x00	Conversion result	0x0000	R	R	R
0x01	Channel	0x0000	R/W	R/W	R/W
0x02	Configuration	0x00C0	R/W	R/W	R/W
0x03	Alert indication	0x0000	R	R	R
0x04	Channel 0 low limit	0x0000	R/W	R/W	R/W
0x05	Channel 0 high limit	0x01FF	R/W	R/W	R/W
0x06	Channel 0 hysteresis	0x01FF	R/W	R/W	R/W
0x07	Channel 1 low limit	0x0000	R/W	R/W	R/W
0x08	Channel 1 high limit	0x01FF	R/W	R/W	R/W
0x09	Channel 1 hysteresis	0x01FF	R/W	R/W	R/W
0x0A	Channel 2 low limit	0x0000	R/W	R/W	NOP
0x0B	Channel 2 high limit	0x01FF	R/W	R/W	NOP
0x0C	Channel 2 hysteresis	0x01FF	R/W	R/W	NOP
0x0D	Channel 3 low limit	0x0000	R/W	R/W	NOP
0x0E	Channel 3 high limit	0x01FF	R/W	R/W	NOP
0x0F	Channel 3 hysteresis	0x01FF	R/W	R/W	NOP
0x10	Channel 4 low limit	0x0000	R/W	NOP	NOP
0x11	Channel 4 high limit	0x01FF	R/W	NOP	NOP
0x12	Channel 4 hysteresis	0x01FF	R/W	NOP	NOP
0x13	Channel 5 low limit	0x0000	R/W	NOP	NOP
0x14	Channel 5 high limit	0x01FF	R/W	NOP	NOP
0x15	Channel 5 hysteresis	0x01FF	R/W	NOP	NOP
0x16	Channel 6 low limit	0x0000	R/W	NOP	NOP
0x17	Channel 6 high limit	0x01FF	R/W	NOP	NOP
0x18	Channel 6 hysteresis	0x01FF	R/W	NOP	NOP
0x19	Channel 7 low limit	0x0000	R/W	NOP	NOP
0x1A	Channel 7 high limit	0x01FF	R/W	NOP	NOP
0x1B	Channel 7 hysteresis	0x01FF	R/W	NOP	NOP
0x1C	Reserved	0x0000	NOP	NOP	NOP
...	...	...	...	...	...
0x1F	Reserved	0x0000	NOP	NOP	NOP

ADC124S051

- Half as many pins as previous
- Straightforward programming
  - SPI connections already pre programmed into Mega
  - byte dataOut = SPI.transfer(byte dataIn)
  - Challenging part will be reading 12 bits out before changing channels
- Easier testing

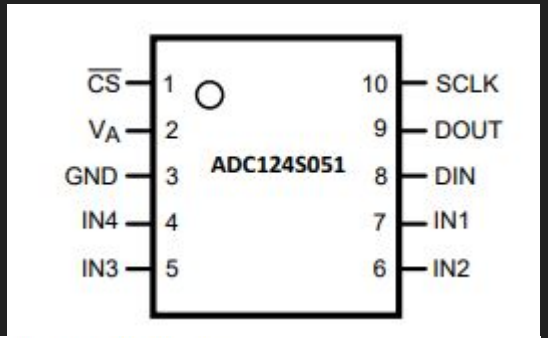


Table 2. Control Register Bits

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DONTC	DONTC	ADD2	ADD1	ADD0	DONTC	DONTC	DONTC

Table 3. Control Register Bit Descriptions

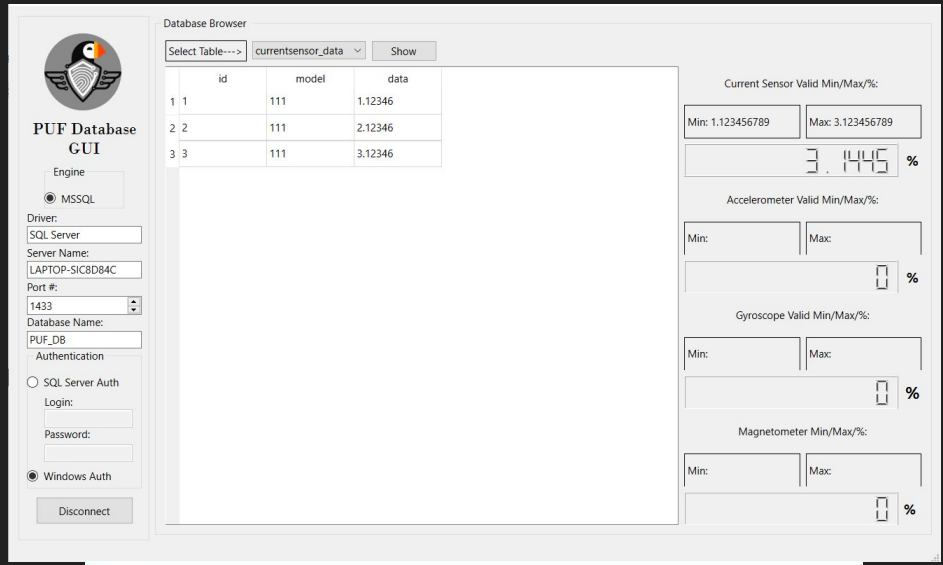
Bit #:	Symbol:	Description
7 - 6, 2 - 0	DONTC	Don't care. The value of these bits do not affect device operation.
5	ADD2	These three bits determine which input channel will be sampled and converted in the next track/hold cycle. The mapping between codes and channels is shown in Table 4.
4	ADD1	
3	ADD0	

Table 4. Input Channel Selection

ADD2	ADD1	ADD0	Input Channel
x	0	0	IN1 (Default)
x	0	1	IN2
x	1	0	IN3
x	1	1	IN4

Graphic User Interface(GUI)

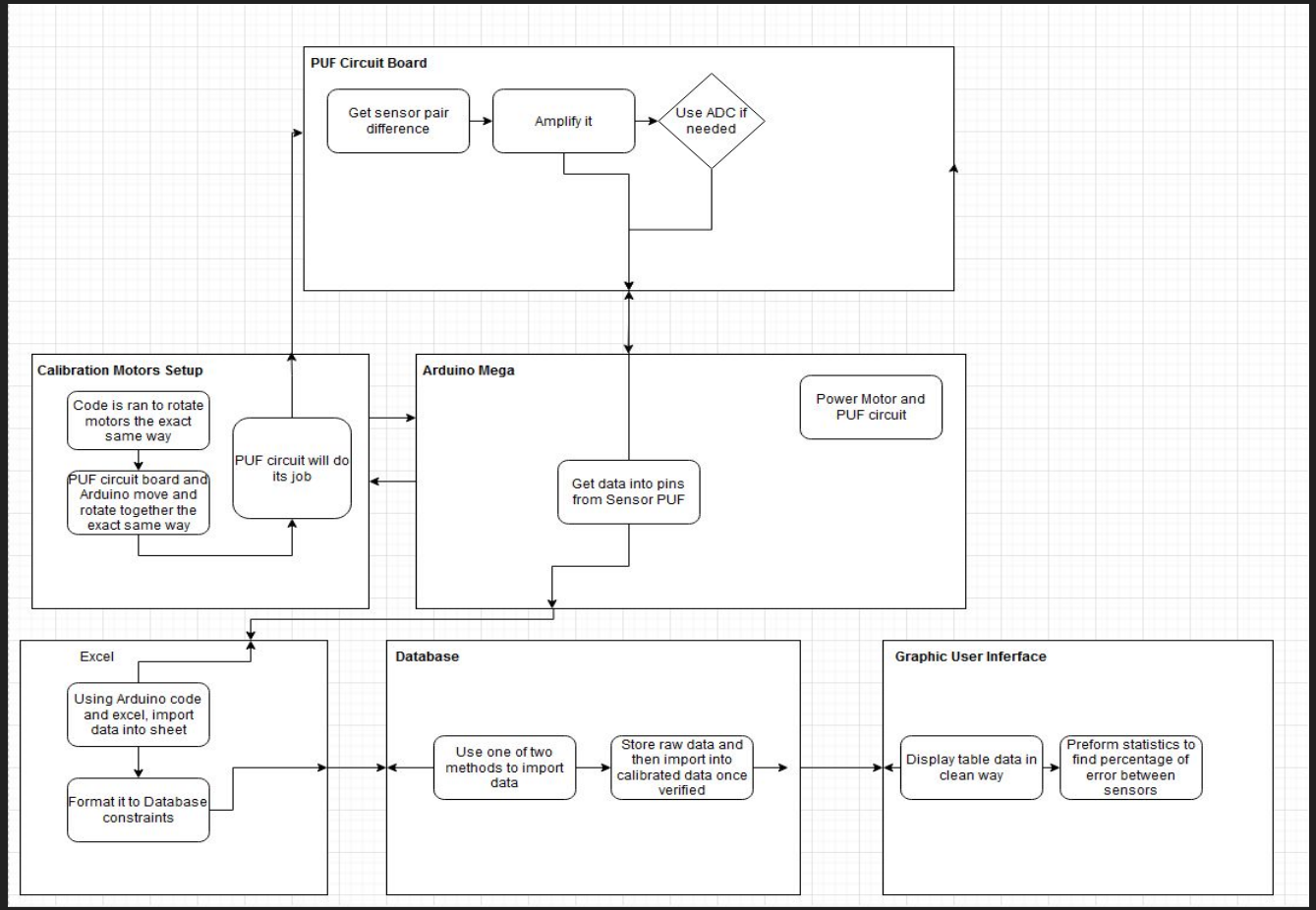
- Everything is done except the MIN/MAX/% Functions
- Have to do conversions 3 times on the data and its causing problems
- QVariant-> QString -> double and back for comparison loop
- Finally it will be deployed to server hardware bought and put in the lab



$$\%\ \text{Error} = \left| \frac{\text{Theoretical Value} - \text{Experimental Value}}{\text{Theoretical Value}} \right| \times 100$$

Theoretical Value = Actual ... Known ...True Value

Testing Process



What needs to be Done

- PCB
  - Revision 2 is working perfectly besides ADC
  - Working on Revision 3 to support newly picked ADC
    - Test everything as this might be the last board we can get before the end of the semester
- Integrate all components and test the overall product together collecting the converted values into excel using Arduino code
- Database and GUI
  - Fix final functions for Min, Max, % Error
  - Ensure the system as a whole works with real data from our circuit
  - Compile final build to .exe
  - Deploy server on purchased hardware and start collecting and using real data

Thank you for your time

Any Questions?