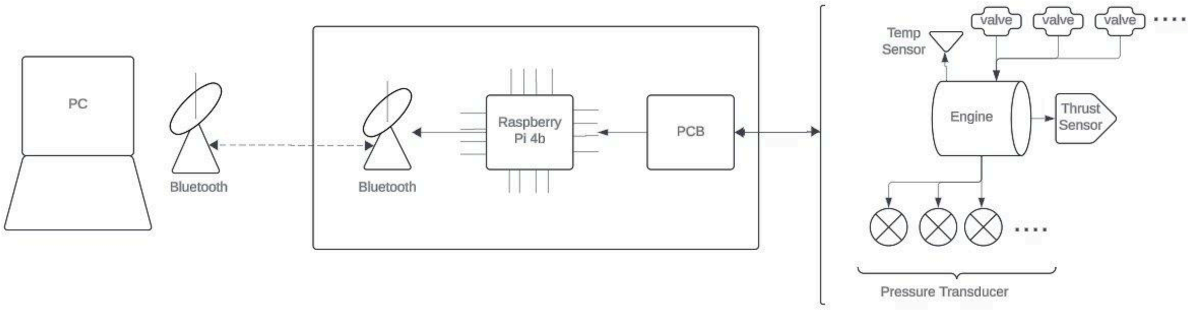
Electrical System Details

Table of Contents

High-Level Technical Overview

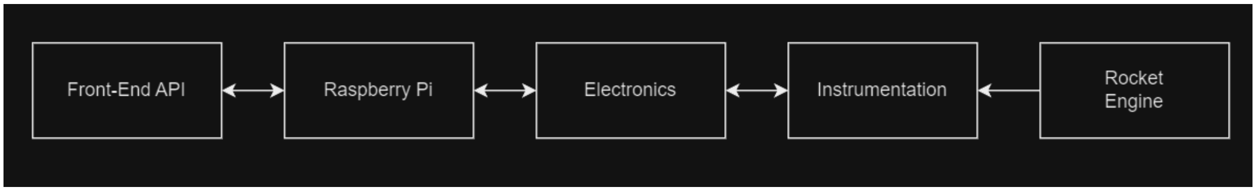
High-Level Electrical System Design

- Highlights the flow of data in the system



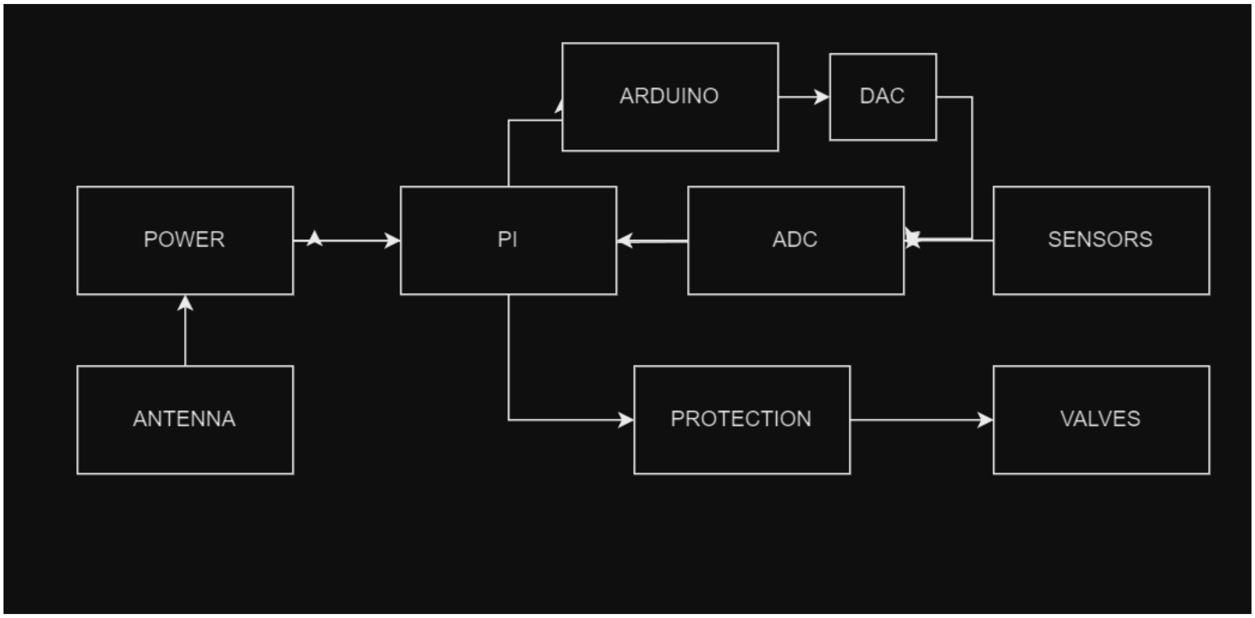
High-Level Block Diagram of Electrical System

- Adds more official terminology



Functional Diagram of Hardware Box

- Highlights the major hardware systems interacting with each other
- Missing are the LED’s and Sensor Filters

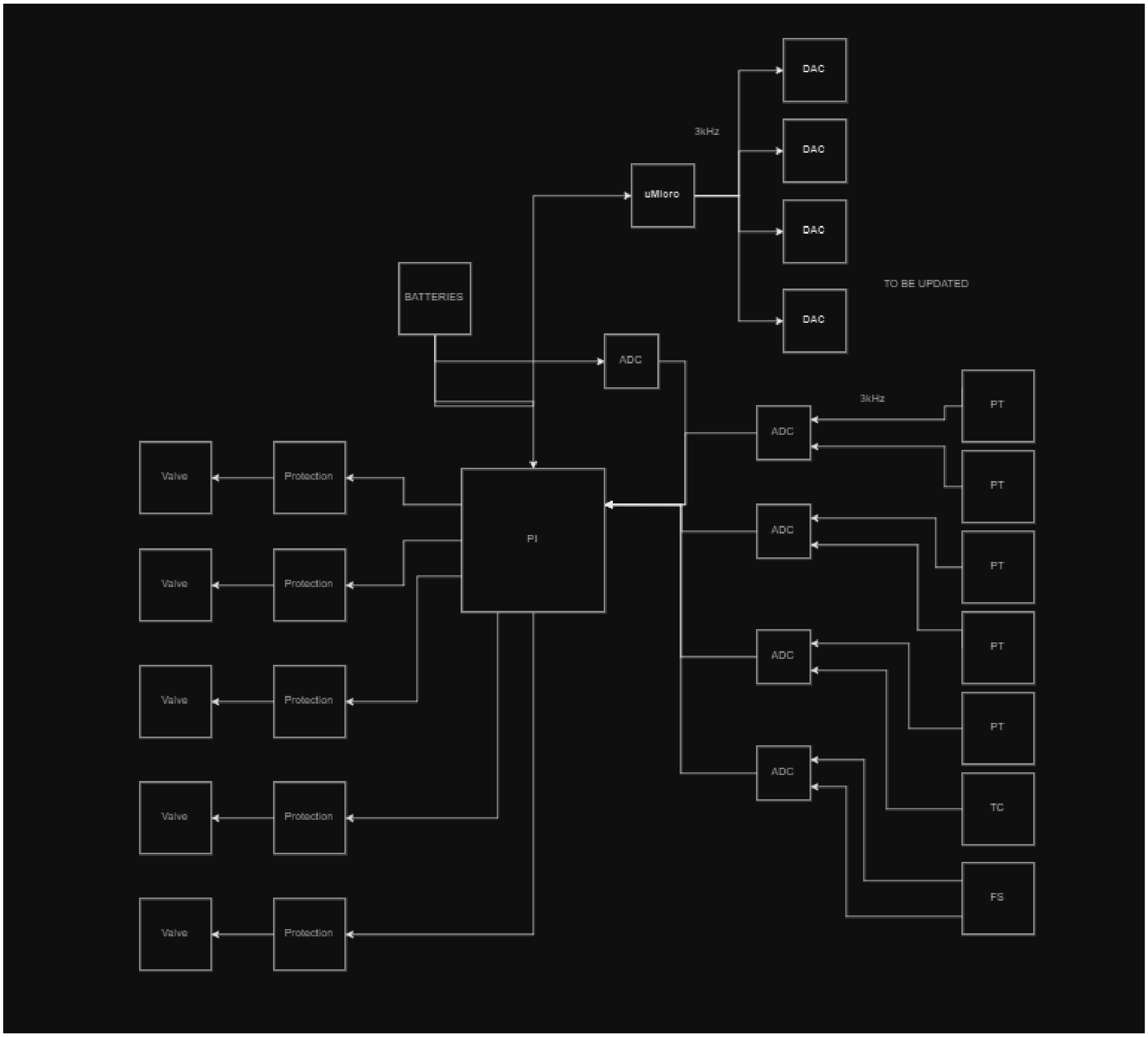


Box Hardware Details

Functional Diagram

Mid-Level Function Diagram

- I do not know the testing and sensor switching for the system correct
- Missing antenna and testing/sensor switching will be updated

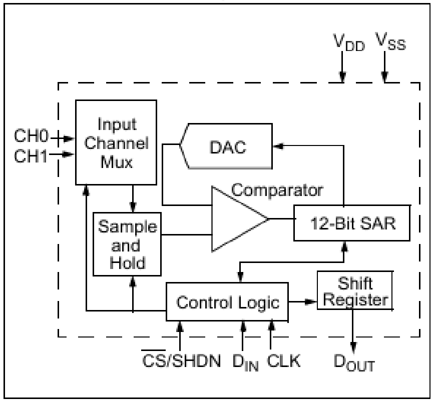
- 

ADC 1 and 2 for Pressure Transducers

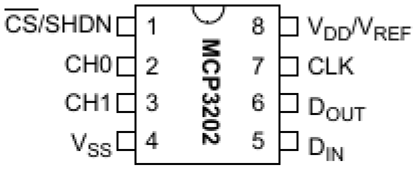
- MCP3202
- SPI
- 12-bit resolution
- SAR
- 5V

- <= 100ksps
- will use PI 5v rail as a reference voltage
- Adding diodes to sensor outputs for protection and isolation
- Will update design when available

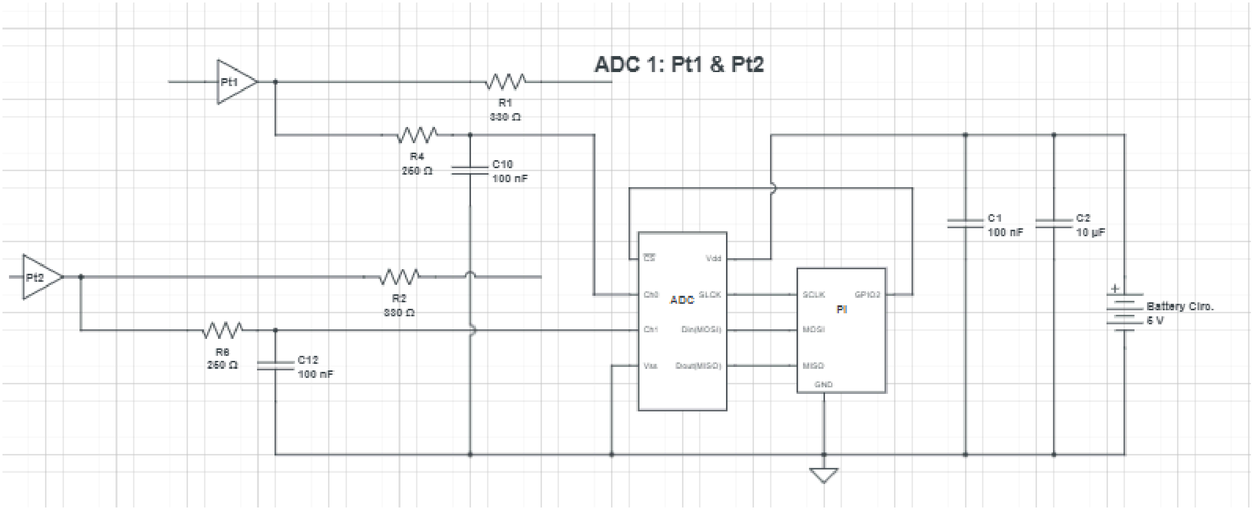
Functional Block Diagram



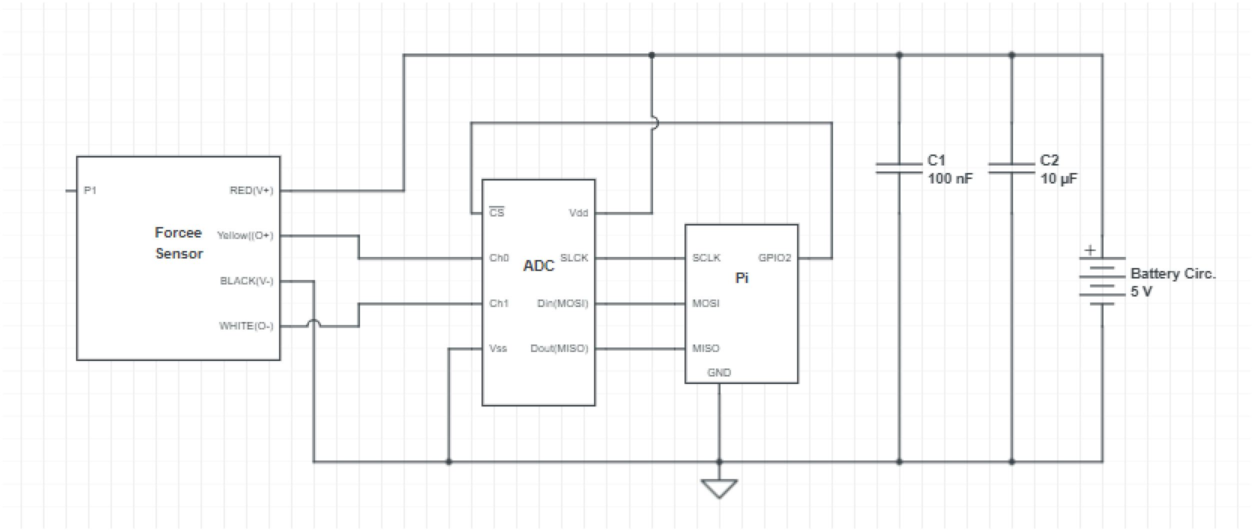
-



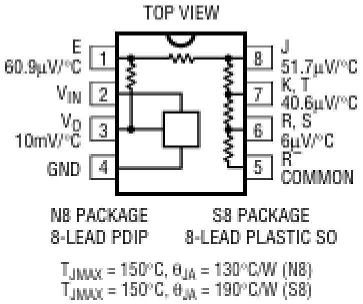
-

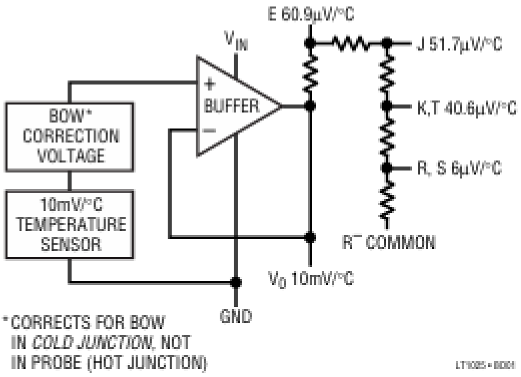


Force Sensor ADC



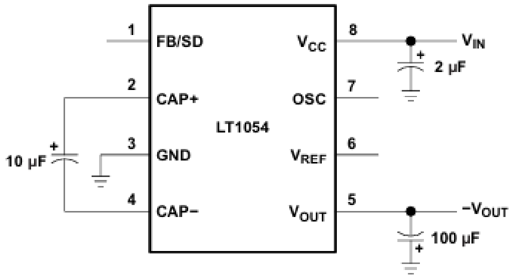
ThermoCouple

- Cold Junction Compensator (LT11025)
    - Thermocouples work by using 0C as a reference if the system isnt at 0C we need a way to artificially show that 0V is OC and thats a CJC
- 0.5C accuracy
- 10mV/C output
- CJC
    - 

    - 

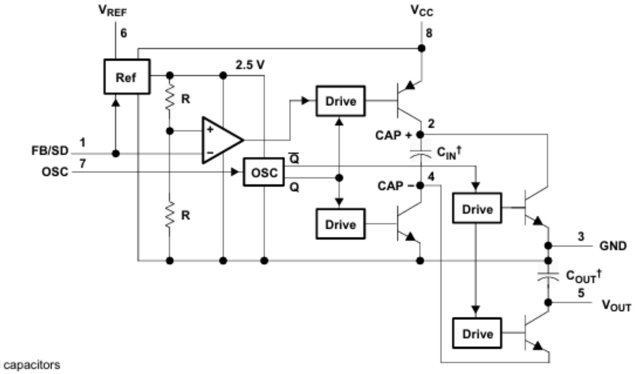
- Inverter

Basic Voltage Inverter



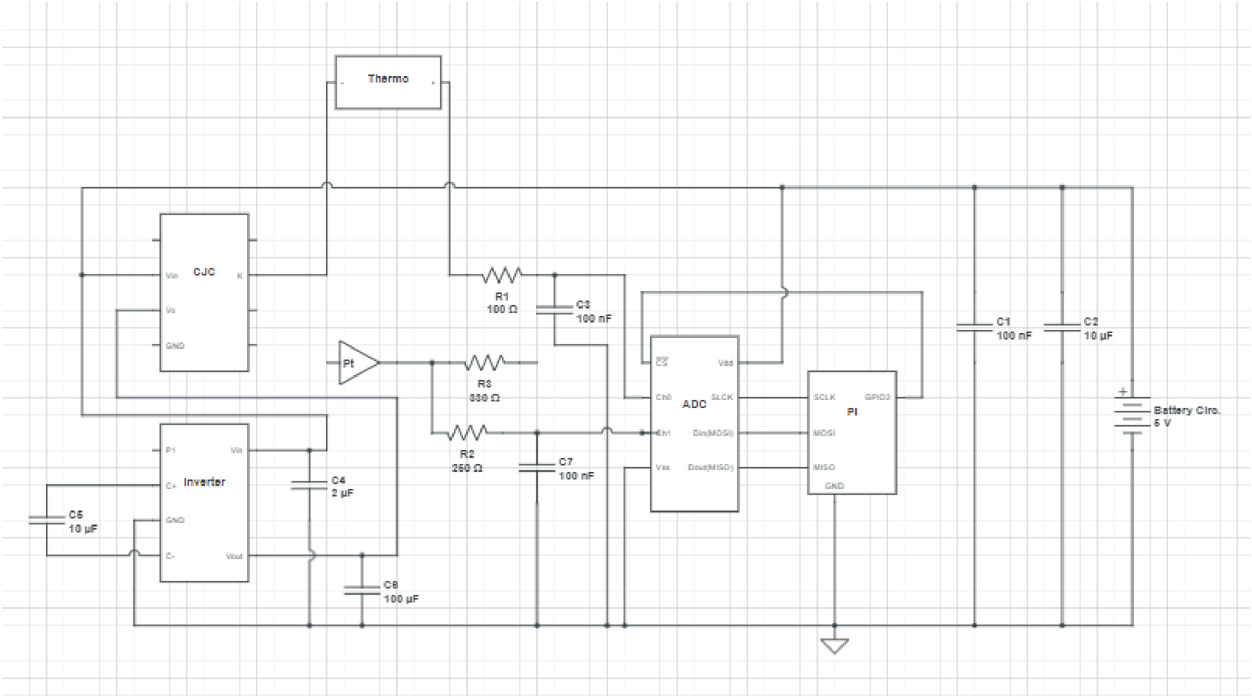
○

7.2 Functional Block Diagram

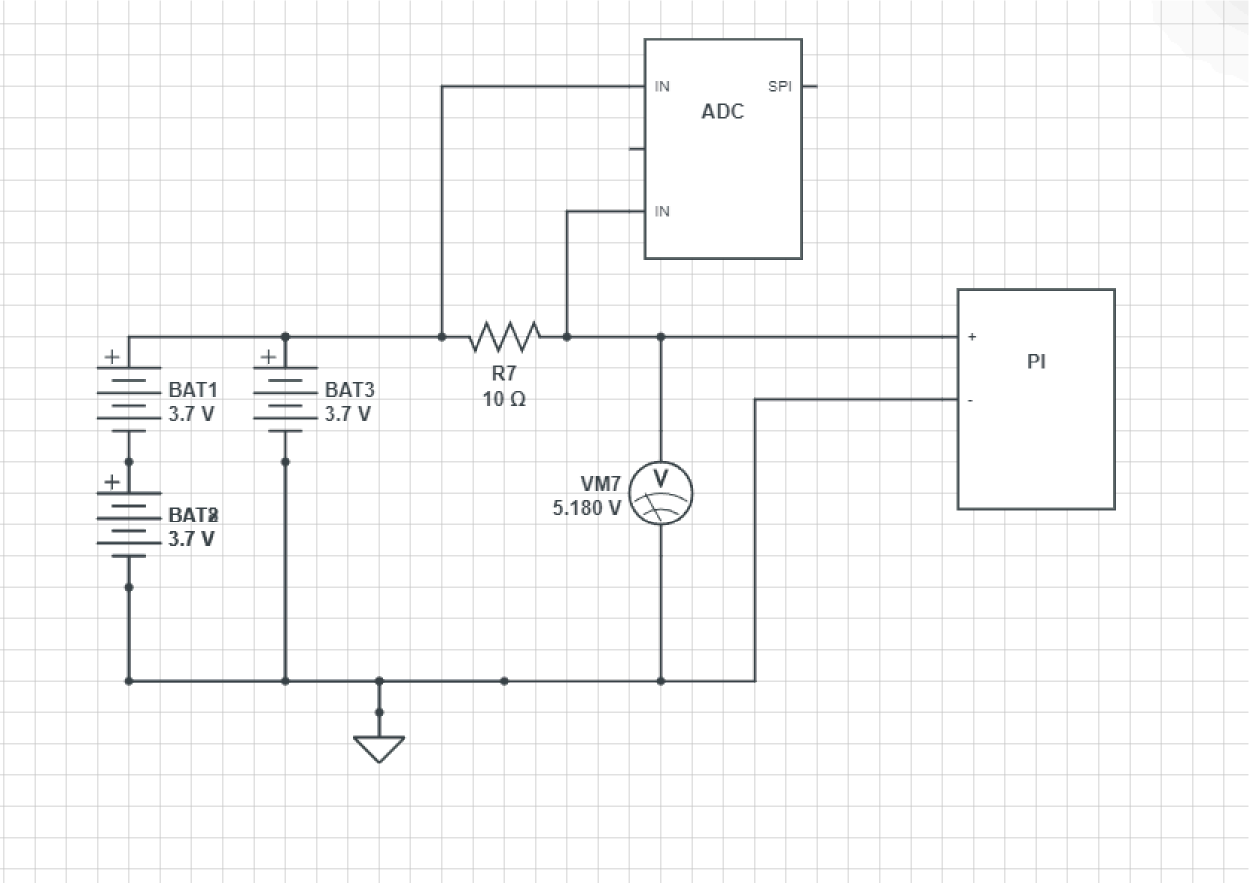


† External capacitors

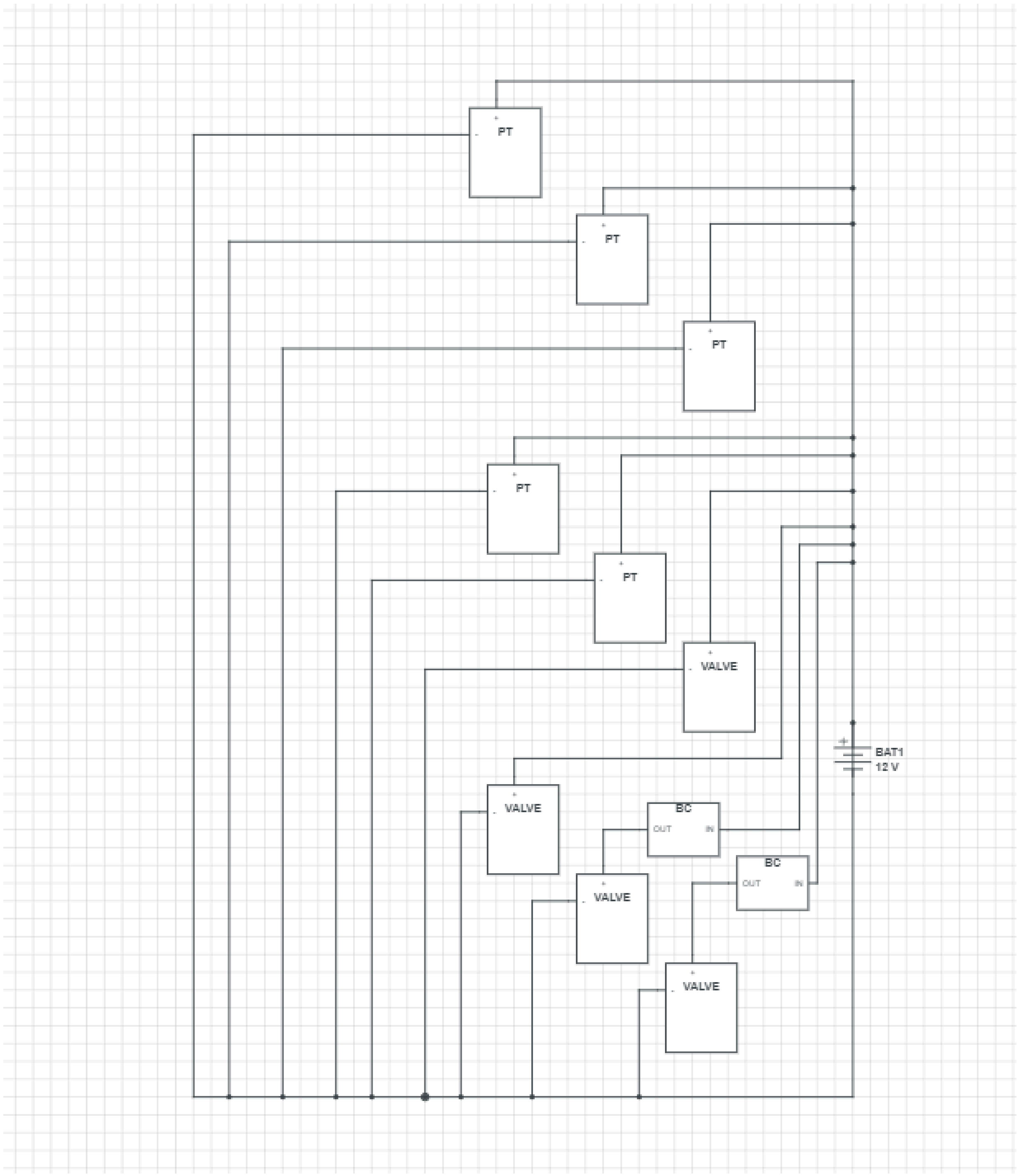
○ Pin numbers shown are for the P package.



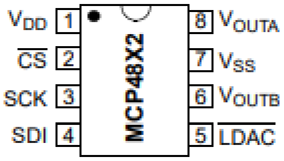
Power System

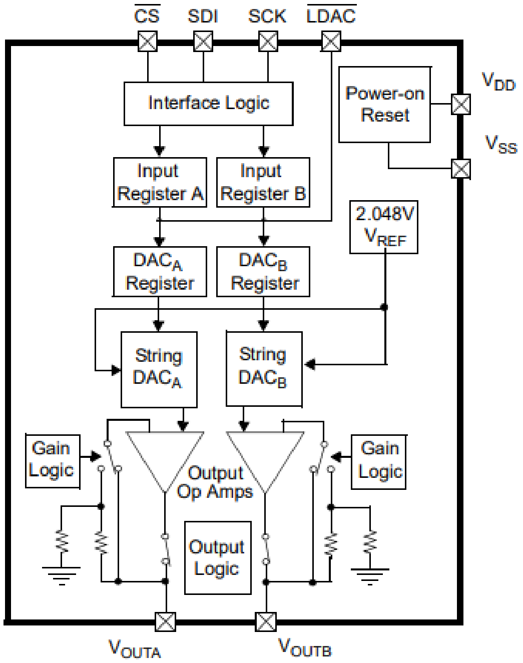


-
- Look for a way to connect pi usb-c to power jack and power jack to batteries


PT
PT
PT
PT
PT
VALVE
VALVE
BC
OUT
IN
VALVE
BC
OUT
IN
VALVE
BAT1
12 V

DAC For Testing

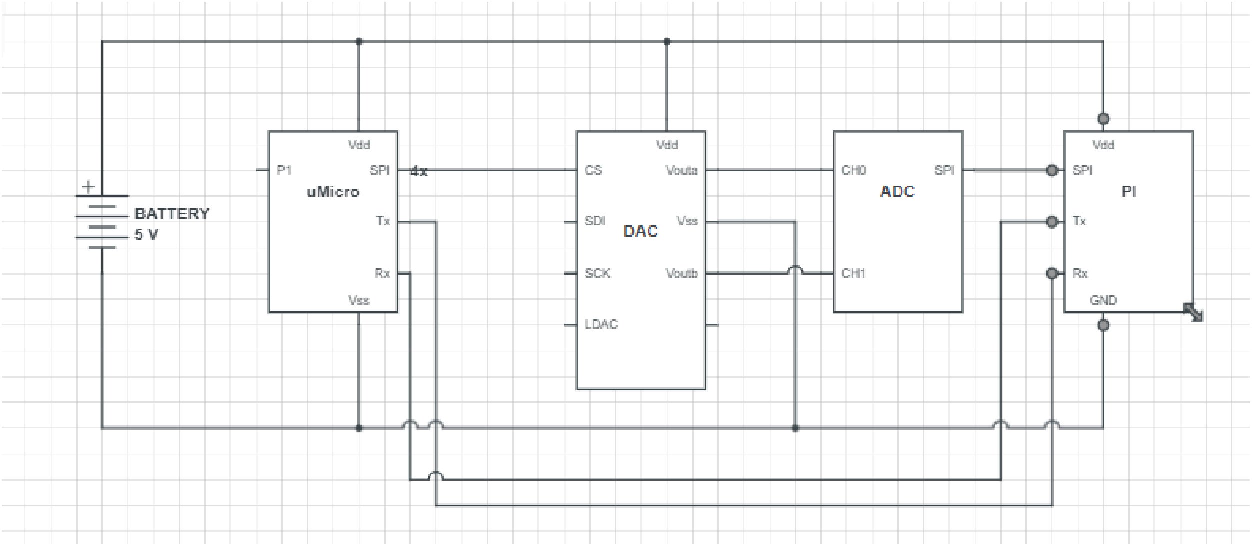
- MCP4802
- 12-bit resolution
- 3-wire SPI
- Internal Voltage Reference
- Arduino uMicro
- 

  MCP4802: 8-bit dual DAC
  MCP4812: 10-bit dual DAC
  MCP4822: 12-bit dual DAC
- 

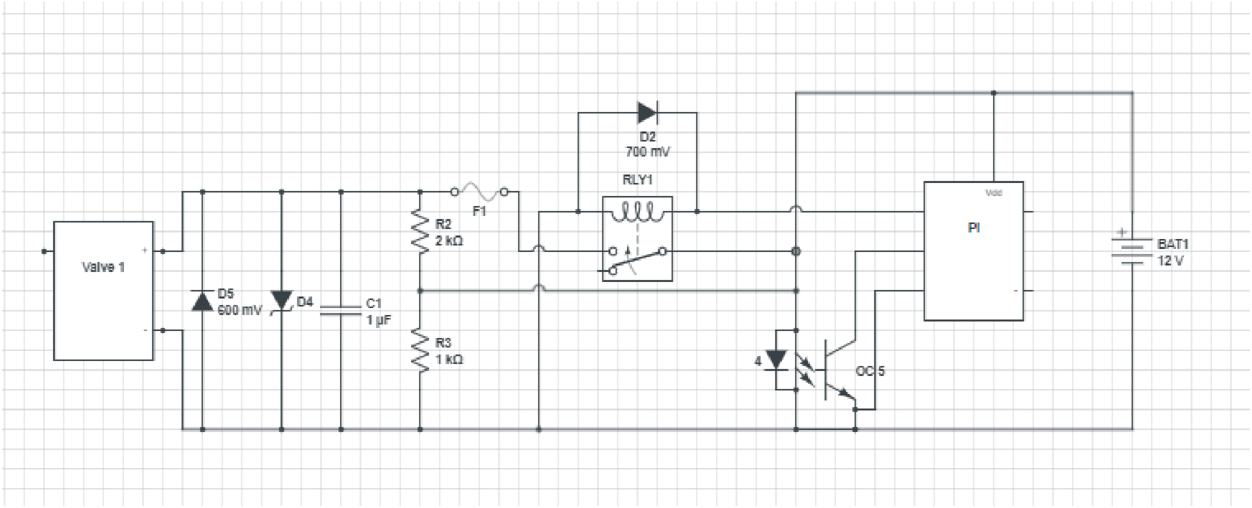
- Details
    - Using a microcontroller to control the SPI lines of the Arduino to control the DAC hardware
    - I want a async way to send data
        - variable pwm to analog converter was not viable
        - Arduino or esp32 does not have enough memory

    - Same SPI network would not have for mux for tx and rx data

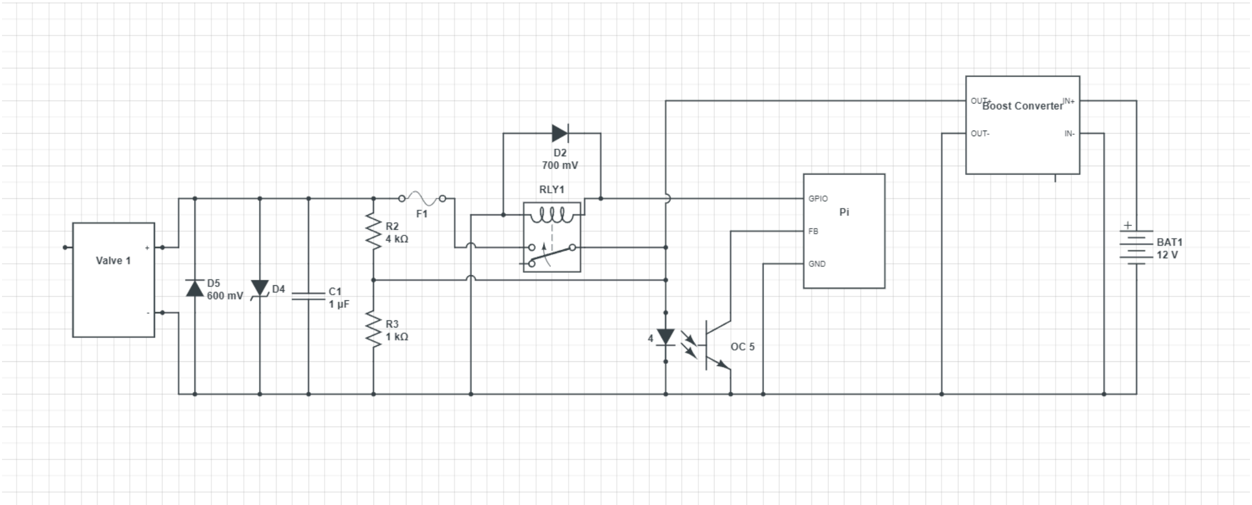
  Landed on ucontroller controlling for asynch spi lines and pre flashing with test data

- 

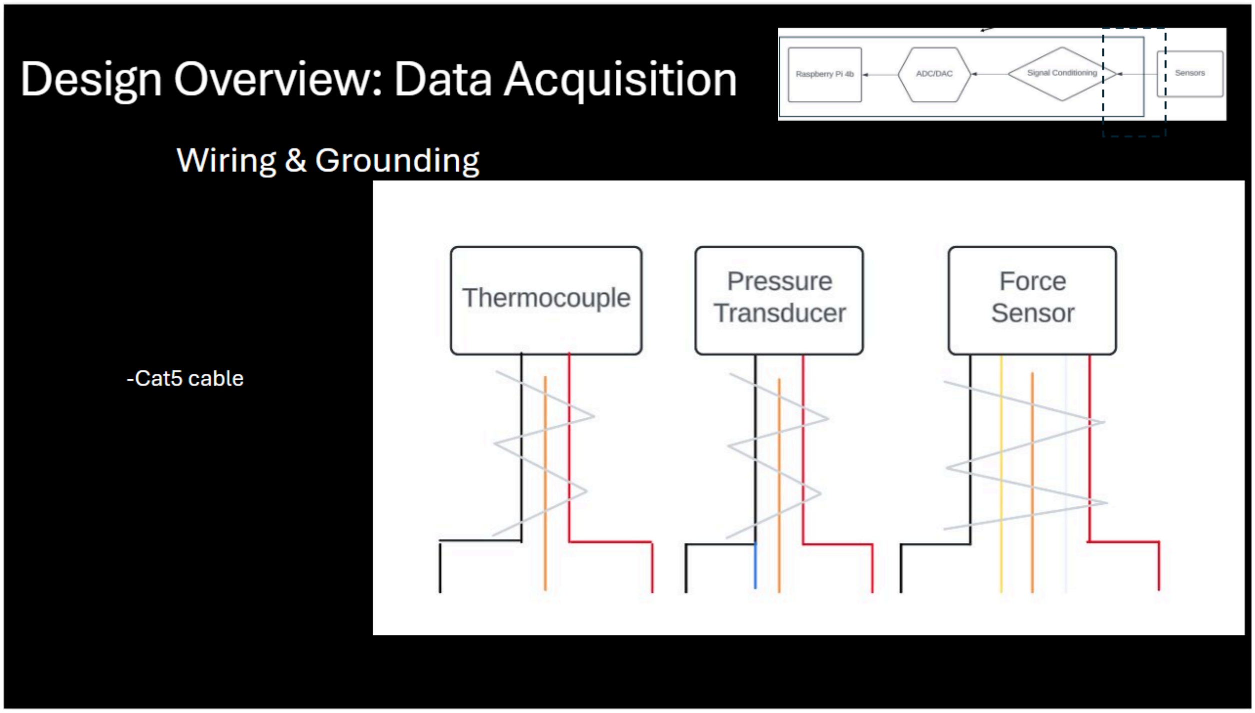
Valve Circuits

- **Look into replacing Relays with MOSFETs**
- **12V**
- 

- **24V**
    - **Look into boost converter for 24V valve**

- 

Instrumentation Hardware Details

- 

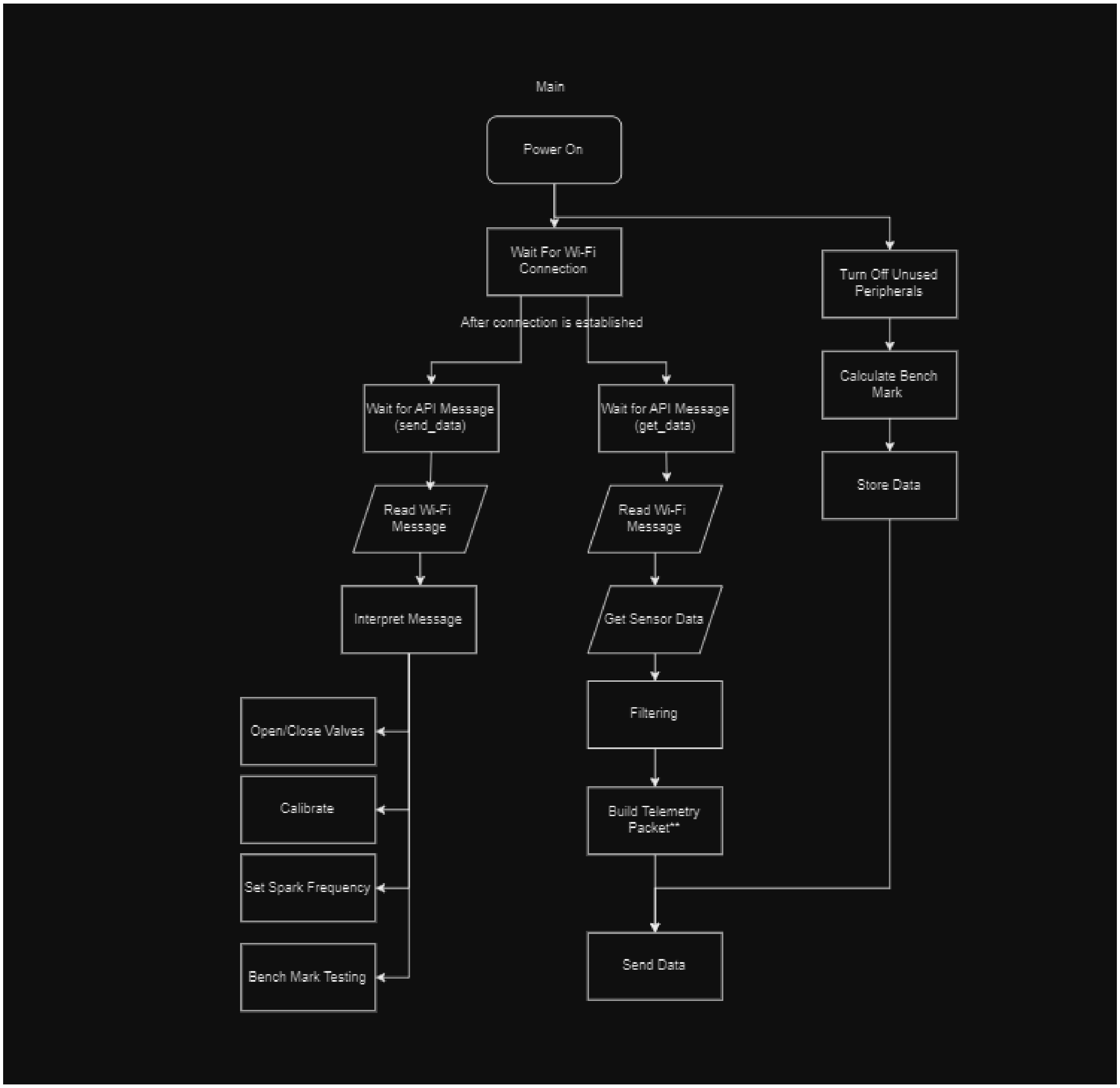
Design Overview: Data Acquisition

Wiring & Grounding

-Cat5 cable

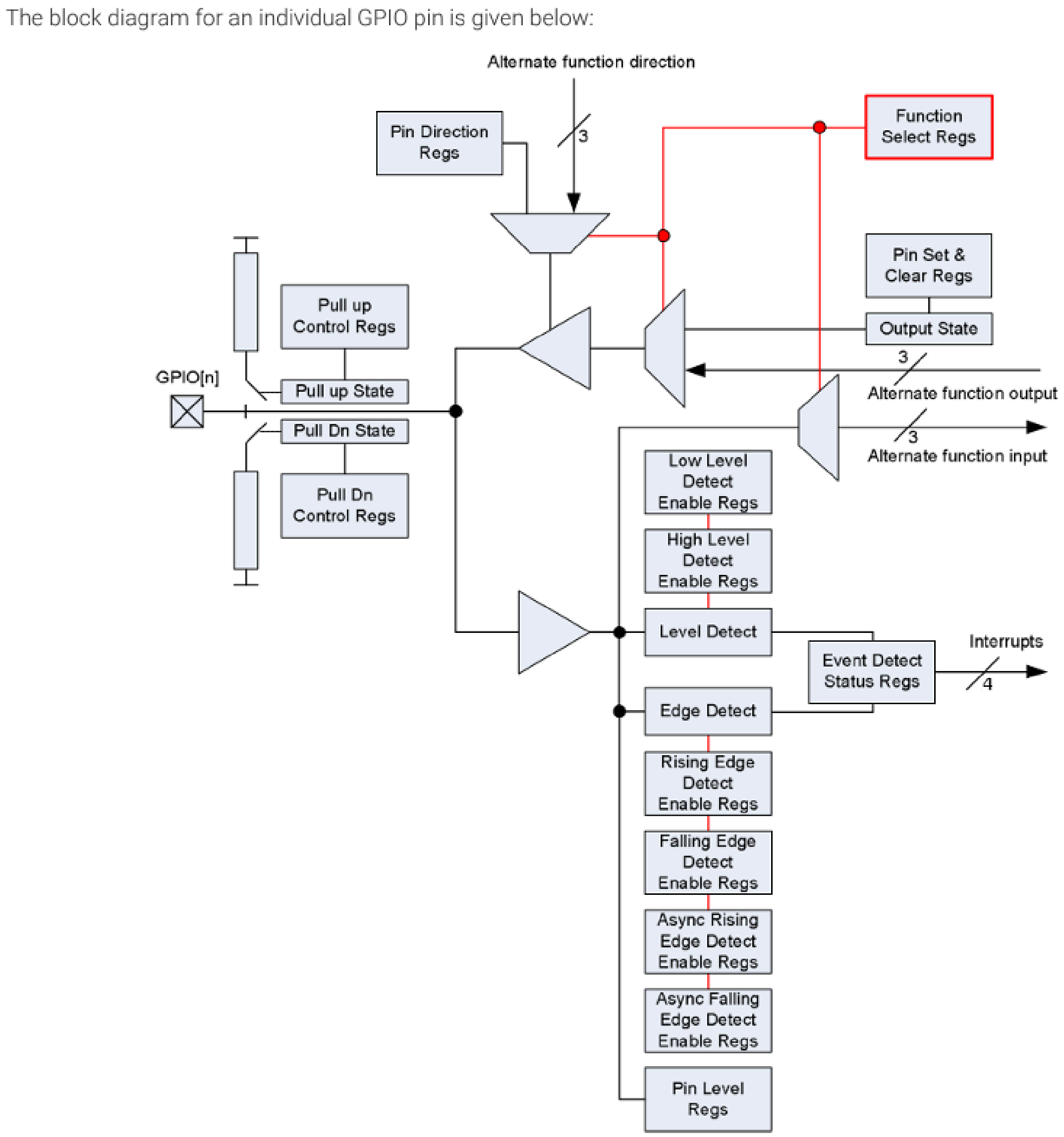
Software Details

- Each section is labeled a level the lower the level the lower level the software and vice versa it ranges from 0-4
- **High-Level Idea of the Software System**



GPIO Driver
https://github.com/izukaike/V2_Electrical_System/blob/main/src/gpio.hpp
Pin Hardware

- No current protection
- No overvoltage protection -> optocoupler with 3.3 volts is used
    - How much current does 3.3V use?

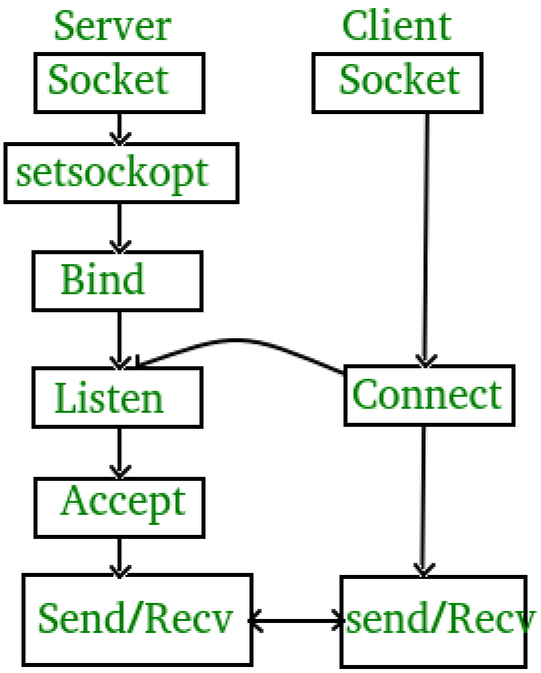


The block diagram for an individual GPIO pin is given below:

Alternate function direction
3
Pin Direction Regs
Function Select Regs
Pin Set & Clear Regs
Output State
Pull up Control Regs
Pull up State
Pull Dn State
Pull Dn Control Regs
GPIO[n]
3
Alternate function output
3
Alternate function input
Low Level Detect Enable Regs
High Level Detect Enable Regs
Level Detect
Event Detect Status Regs
Interrupts
4
Edge Detect
Rising Edge Detect Enable Regs
Falling Edge Detect Enable Regs
Async Rising Edge Detect Enable Regs
Async Falling Edge Detect Enable Regs
Pin Level Regs

Client Driver

https://github.com/izukaike/V2_Electrical_System/blob/main/src/client.hpp



- Utilizes sys/socket.h -> POSIX (portable operating system interface) ->
- Connect
    - Checks if sockfd is valid
    - Validates rest structure -> how? idk
    - Client send synch to host
    - Host sends sync-ack
    - Client send ack

Notes

- IPv4 vs IPv6
- TCP vs UDP
- IPROTO_TCP (default proto)?

- OOP Structure
    - Pi Side
        - Client

        - Telemetry/WifiF
        - GPIO
        - PWM
        - ADC
        - DAC
        - Coil
        - Linux
    - Py Side
        - Metrics
        - Telemetry
        - Wi-Fi Host
        - System Health

Level 1

ADC: MCP3202

Driver: https://github.com/izukaike/V2_Electrical_System/blob/main/src/adc.cpp

- Datasheet SPI interface
- SPI Mode 0: falling clock edge, rising data edge
- 8 bits per word

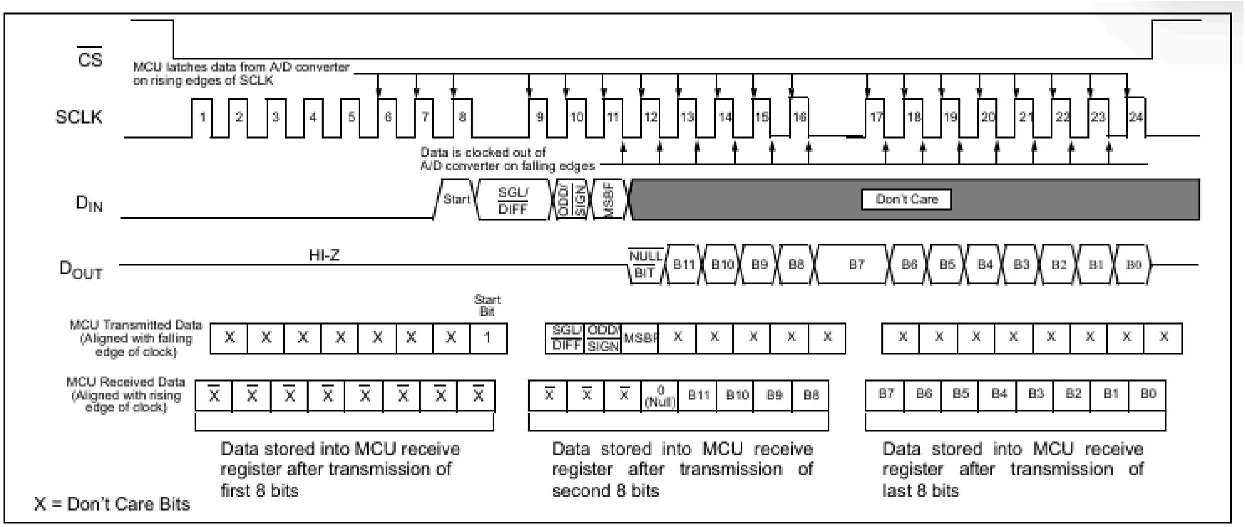

CS
MCU latches data from A/D converter on rising edges of SCLK
SCLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Data is clocked out of A/D converter on falling edges
DIN
Start
SGL/DIFF
ODD/SIGN
MSBF
Don't Care
DOUT
HI-Z
NULL BIT
B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
MCU Transmitted Data (Aligned with falling edge of clock)
X X X X X X X 1
Start Bit
SGL/DIFF ODD/SIGN MSBF X X X X X
X X X X X X X X
MCU Received Data (Aligned with rising edge of clock)
X X X X X X X X
X X X 0 (Null) B11 B10 B9 B8
B7 B6 B5 B4 B3 B2 B1 B0
Data stored into MCU receive register after transmission of first 8 bits
Data stored into MCU receive register after transmission of second 8 bits
Data stored into MCU receive register after transmission of last 8 bits
X = Don't Care Bits

FIGURE 6-1: *SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).*

- My software driver
    - Why Is it not working
        - use volatile?
        - Proper 5V reference voltage?
        - Read Datasheet
        - Proper Read/Write Command?
        - Arduino driver and translate to C++

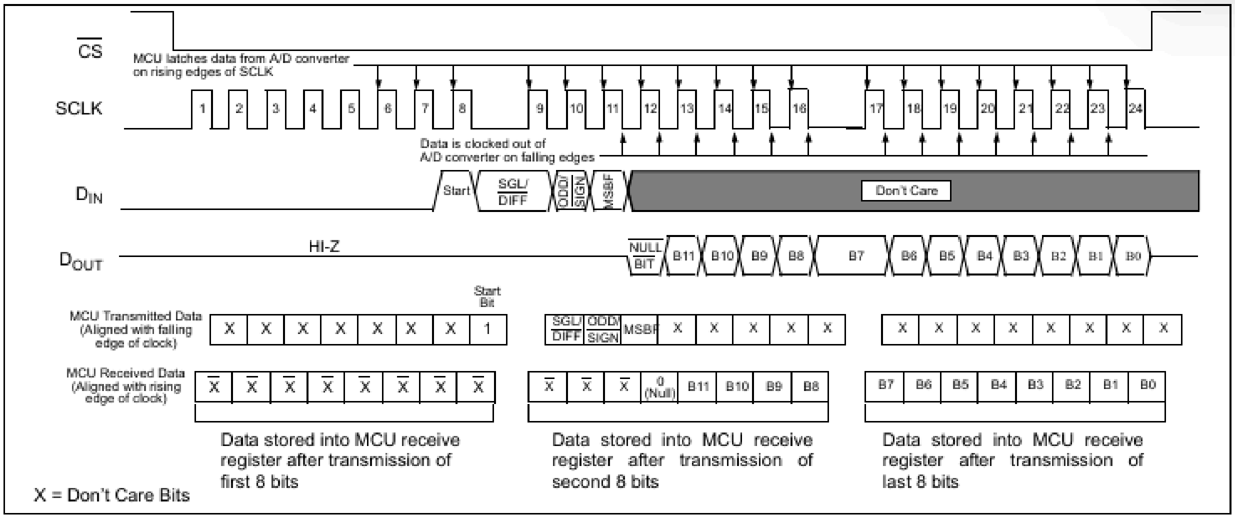

CS
MCU latches data from A/D converter on rising edges of SCLK
SCLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Data is clocked out of A/D converter on falling edges
DIN
Start
SGL/DIFF
ODD/SIGN
MSBF
Don't Care
DOUT
HI-Z
NULL BIT
B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
Start Bit
MCU Transmitted Data (Aligned with falling edge of clock)
X X X X X X X 1
SGL/DIFF ODD/SIGN MSBF X X X X X
X X X X X X X X
MCU Received Data (Aligned with rising edge of clock)
X X X X X X X X
X X X 0 (Null) B11 B10 B9 B8
B7 B6 B5 B4 B3 B2 B1 B0
Data stored into MCU receive register after transmission of first 8 bits
Data stored into MCU receive register after transmission of second 8 bits
Data stored into MCU receive register after transmission of last 8 bits
X = Don't Care Bits

FIGURE 6-1: *SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).*

TABLE 5-1: CONFIGURATION BITS FOR THE MCP3202

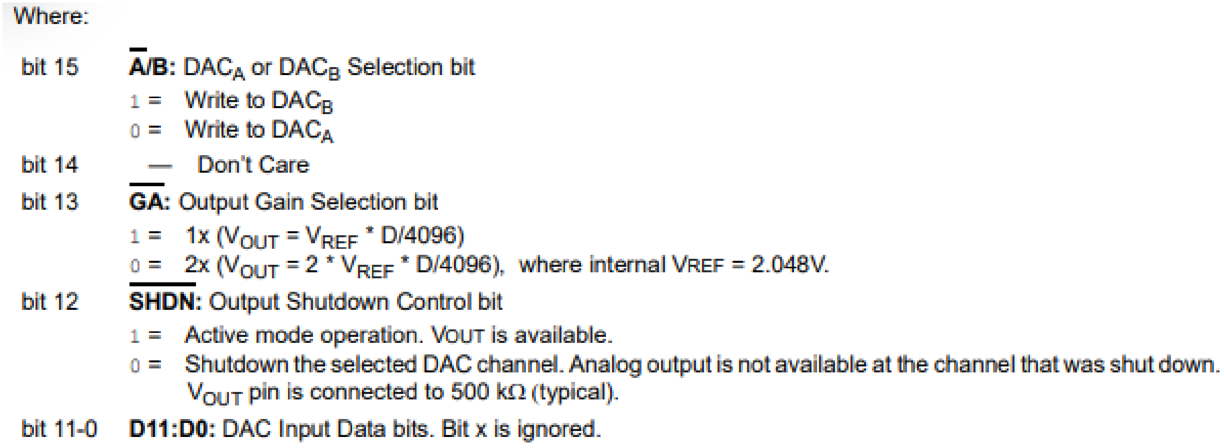
	Config Bits		Channel Selection		GND
	SGL/ DIFF	**ODD/ SIGN**	**0**	**1**	
Single-Ended Mode	1	0	+	—	-
	1	1	—	+	-
Pseudo-Differential Mode	0	0	IN+	IN-	
	0	1	IN-	IN+	

DAC: MCP4822

Back-End Driver:

https://github.com/SteveGdvs/MCP48xx/blob/master/src/MCP48xx.h#L90

look into customizing my own using my own gpio drivers too making it come full circle



Where:

bit 15 **$\overline{A}$/B:** DAC$_A$ or DAC$_B$ Selection bit

1 = Write to DAC$_B$

0 = Write to DAC$_A$

bit 14 — Don't Care

bit 13 **$\overline{GA}$:** Output Gain Selection bit

1 = 1x ($V_{OUT} = V_{REF} * D/4096$)

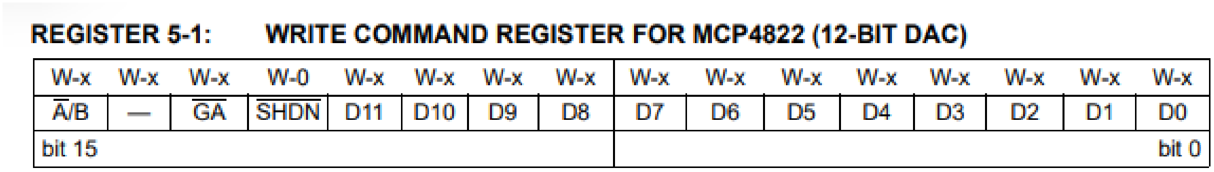
0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$), where internal VREF = 2.048V.

bit 12 **$\overline{SHDN}$:** Output Shutdown Control bit

1 = Active mode operation. VOUT is available.

0 = Shutdown the selected DAC channel. Analog output is not available at the channel that was shut down. V_{OUT} pin is connected to 500 kΩ (typical).

bit 11-0 **D11:D0:** DAC Input Data bits. Bit x is ignored.



REGISTER 5-1: WRITE COMMAND REGISTER FOR MCP4822 (12-BIT DAC)

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
$\overline{A}$/B	—	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
bit 15															bit 0

Bench Mark Testing

- This is the main framework I want to use to test the overall performance of the system for setting standards and general and data-driven ways to push improvements.
- I want to make a standardized way to test the system as a whole
- Things to be measured and compared
  - Power draw
    - Improve power efficiency
  - Wi-fi speed
    - faster ,stable,reliable wifi connection
  - tx/rx speed
    - Might be synonymous with wifi
  - Sensor accuracy
    - Improve signal of crucial sensor data especially with emi
  - Abort Logic
    - Proper safety measures are working properly
  - Memory and CPU Usage

        - Smaller mem and cpu footprint
    - Temperature
        - Temperature varying performance and limits
    - Security
        - how to measure security performance
    - Filtering
        - How well filtering algos work

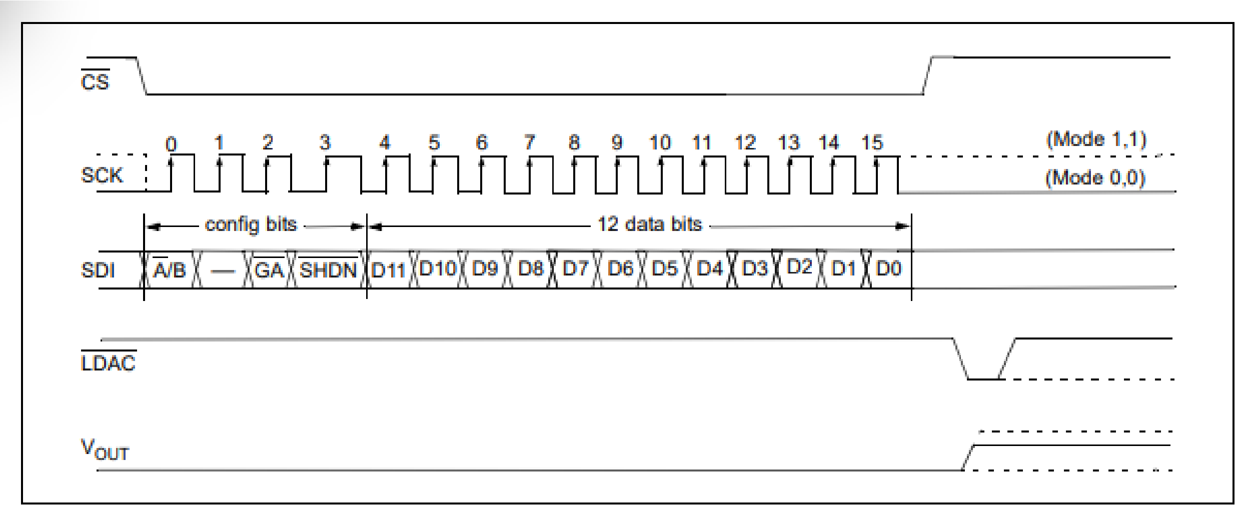


FIGURE 5-1: *Write Command for MCP4822 (12-bit DAC).*