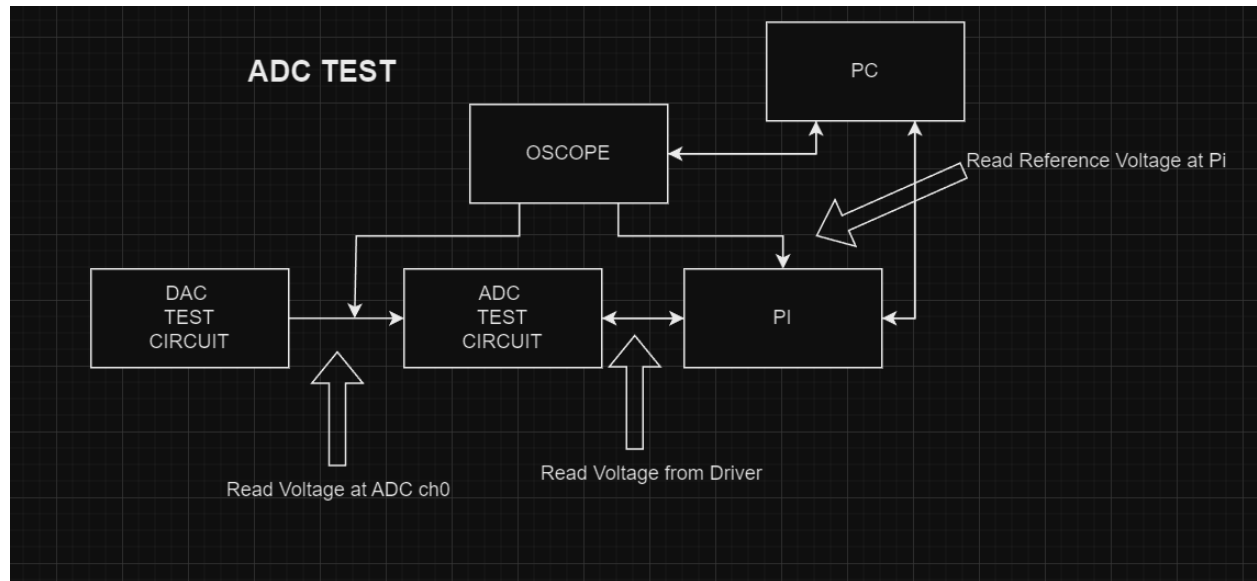


ADC Verification

Goal: Write ADC firmware, Interface ADC hardware with Pi to get < 1% error margin

- **Sub-Goal: Test and develop automated testing environment for onboard benchmark testing**

Test Setup High Level



Made Automated Testing Script that got the reference voltage, DAC Output/ADC Input, ADC/Pi Reading and ADC timing.

- Step 1
 - Run code on AVR microcontroller that has a steady voltage, sweep, and frequency sweep (4.1Vp)
 - Link to github
- Step 2
 - Write Python Script that interfaces with the Keysight Oscilloscope to get reliable data for data analysis in this test I am using the oscscope to use reference voltage for ADC and ADC input to compare the pi software
 - Link to github
- Step 3
 - Make the setup in an easy repeatable and easy debuggable manner (lol)
- Step 4
 - Write a data analysis script that easily shows the oscscope vs pi software and percent difference
 - Link to github
- Step 5
 - Write bash script that all these parts in the proper sequence and correctly file handling
- Step 6
 - Use Excel to find adjustment equation to best fit the input and read value

- Step 7
 - Troubleshoot hardware and software until goal is met

Updates made to test

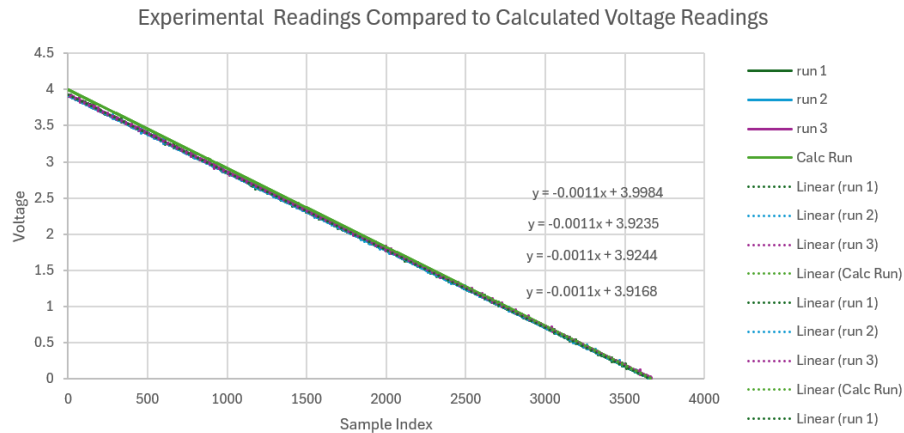
- Pi script for ADC time is 40ms -> 0.2ms with c++ file handling in bash
- Python script improved +20ms read to 0.12 ms over **166x** speed improvement read time with separate read and processing

Update made to firmware

- Made adjustment formula to get correct result
 - Formula to go from ADC read bit to voltage is

```
d1 = (((d1/4096))*5.126)
d1 = d1 + (d1*0.076833)/3.997349
```

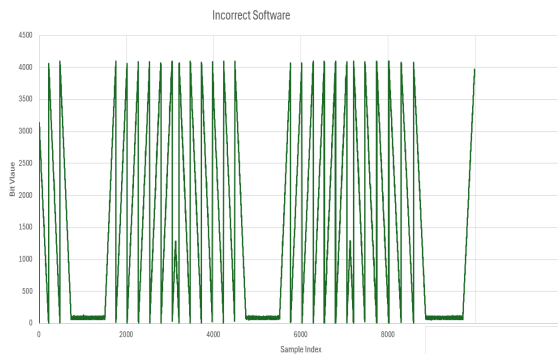
- I got this formula by sweeping from 0-3.99V and compared the lines and calculated the adjustments needed to make the reading match the actual value



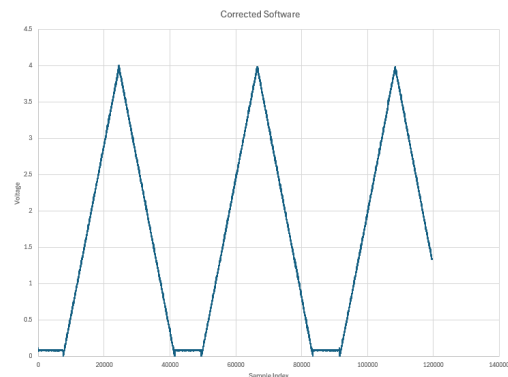
- - This shows 3 experimental runs compared to an ideal run and I noticed it was off by 0.076833 volts at 3.99 volts so I use that relationship to calculate how to adjust each reading.

- Fix control byte and second bit for tx buffer (MOSI) & Pre-define rx buffer (MISO)

Before



After



- left is in bit value and right is in voltage but they are proportional 0-4.1V sweep with a 1 second gap at 0 volts

- Move CS pin from main function to ADC read function

```
// Construct the control byte for single-ended mode, channel 0, MSB first
uint8_t control_byte = 0b00001101 | (channel << 6);

// Buffer to hold the data to send/receive

//uint8_t tx_buffer[3] = {0b00000001, 0b1010000, 0b00000000};

//uint8_t rx_buffer[3] = {0};

uint8_t tx_buffer[3] = {control_byte, 0xA0, 0x00};
uint8_t rx_buffer[3] = {0};
```

```
// Construct the control byte for single-ended mode, channel 0,
//uint8_t control_byte = 0b00001101 | (channel << 6);
uint8_t control_byte = 0b00000001 | (channel << 6);

// Buffer to hold the data to send/receive
//uint8_t tx_buffer[3] = {0b00000001, 0b1010000, 0b00000000};
//uint8_t rx_buffer[3] = {0};

volatile uint8_t tx_buffer[3] = {control_byte, 0xA0, 0x00};
volatile uint8_t rx_buffer[3] = {0x00, 0x00, 0x00};
```

Update to Hardware

- added a RC filter to minimize SPI from AVR uController (Arduino) and BCM2877 (Pi) interference that introduces noise
- Possibly adding buffer to minimize input impedance for the S/H circuitry of the ADC

Test Results

- This is data I got before adjusting

