1 ACD - Analog to Digital Conversion

An ACD converter is needed to translate analog data (temperature) from the thermistor to digital data (voltage) that the RaspberryPi can understand. If I were using an Arduino I would not need such a converter as an Arduino has an ADC converter built in where as the RaspberryPi is a digital only microcontroller. Using the MCP3008 for ADC, its pin out:

- VDD - power
- DGND - Digital ground

used to power the converter

- DOUT - data out of the coverter (the covnerted data we desire)
- CLK - clock pin
- DIN - data in from raspberry pi
- CS - Chip select

These 4 pins are used for the SPI (Serial Peripheral Interface) of the raspberry pi. Section 2 gives more on this.

- AGND - analog ground (used for precision which is optional and so can merely be connected to GND)
- Vref - analog reference voltage (used as a scale for the ADC)

Everything above comes from Analog Inputs for Raspberry Pi Using the MCP3008. Discussed below is how we can use this digital data from the ADC converter to translate the data into what we expect, see Analog to Digital Conversion by Adapfruit for more.

Consider a microcontroller is powered with 5v, it understands 0v as a binary 0 and 5v as a binary 1. Now we have a problem when the voltage is anything between 0 and 5 volts (what will be binary 0 and what is binary 1) - this is analog data and is why micocontrollers have difficult time translating what analog data means, this is why we need an ADC converter, to understand analog data in a form it understands (digital data). The MCP3008 ADC converter is a 10-bit converter, meaning it can detect $2^{10} = 1024$ discrete analog levels. The ADC is a ratio value given as $\frac{Resolution of the ADC converter}{System Voltage} = \frac{ADC Reading}{Analog Voltage Measured}$. This means the converter will use the value 1023 to depict the system voltage and value between 0 and 1024 will be a ratio between the 5V and the ADC value 1023. The system voltage of the raspberry pi is either 3.3V or 5V dependending on the voltage pin used to power the converter. Thus we can get the *analog voltage* from this equation that uses ADC values. As an example, imagine we are measuring a voltage with our Pi and the system voltage we are using is 5V, lets assume the ADC value measured on the Pi is 434 and we are using a 10-bit ADC converter. Now to get the analog voltage from this we do the following

$$\begin{aligned} \frac{1023}{5} &= \frac{434}{Analog Voltage Measured} \\ Analog Voltage Measured &= 2.12V \end{aligned} \tag{1}$$

2 SPI - Serial Peripheral Interface

SPI is a communication proocol. SPI is used for synchronous communucation between devices (a master and a slave). The 4 connections of SPI:

- SCLK - clock
- MOSI - Master Out Slave In which is a data line from the master device to the slave device
- MISO - Master In Slave Out which is another data line for sending data from a slave to a master
- SS - Slave Select used to tell the slave that it must be on the 'look out' for data (either sending or receiving)

As the pins suggested, devices oeprating with SPI can act either as a slave or as a master. The SS pin also hints at the ablity to connect multiple slaves to a master device. We can have 3 slaves and we can select which one we want to communicate with (either to receive data from them or to send data to) by applying or not applying voltage to the slave's SS connectionof interest. To start communucation with a slave device we set the SS pin to low.

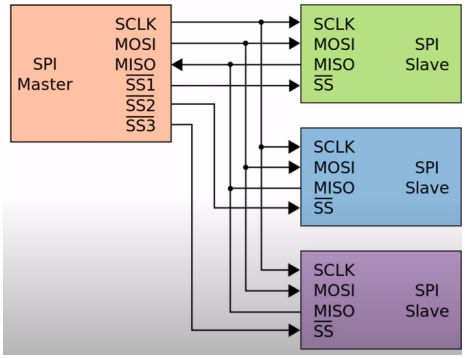

Figure 1: Image from nanland

SPI is full duplex meaining you can send and receive data at the same time. It is not suited for long distance communication (nanland goes as far as saying that even more than a meter is not suited for SPI).

Now when it comes to programming the microcontroller for SPI communication, there are 4 modes of SPI we can set that correspond to when data must be read based on the clock signal send from master to slave via the CLK pin. These 4 modes can be

1. Read/write when clock is low and on rising edge
2. Read/write when clock is high and on rising edge

3. Read/write when clock is low and on falling edge

4. Read/write when clock is high and on rising edge

Great Scott explains the modes rather well. The mode to use depends on the device being used, see the datasheet of the device for more.

SPI transfers data at a CRAZY rate and is part in parsle the reason for its existance

There are 2 ways we can use SPI communication, software SPI or hardware SPI. Hardware SPi is if the devei has pins built specifically for SPI communication. Software SPI is used for devices that can be programmed to use the SPI protocol.

3 Thermistor - Reading Temperature From Voltage

An article titled Thermistors/Temperature Measurement with NTC Thermistors was referenced for this section of the research. As well as an adafruit tutorial on the matter thant can be found here.

An NTC (Negative Temperature Coefficient) thermistor will decrease its resitance with an increase in temperature.

To measure temperature with a thermistor we need to measure the resitance of the thermistor. To do this we can measure the voltage and an additional resitor along with the voltage divider and Ohm's law to calculate the resistance - we can get a better reading by averaging values and by using voltage lines that dont have noise on them. From the resitance we can convert it to a temperature reading using the Steinhart Equation (not as accuate as a temperature table that is provided by the manufacturer of the thermistor). It is given as

$$\frac{1}{T} = A + Bln(R) + C(ln(R))^3 \tag{2}$$

This equation can be used if the manufacturer of the thermsitor provides the values for A, B and C. An alternative is the simplified B parameter equation that only depends on the constant B which most manufacturers do provide.

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}ln(\frac{R}{R_0}) \tag{3}$$

Where T_0 and R_0 are kelvin room temperature and resitance at room temperature. This is essentially the steinhat equation with values for A and C substituted for specific values (see Wikipedia for more).

4 MCP3002

The pip gpiozero package was used to read the ADC value obtained by the MCP3002 API - SPI Devices. There simply was not code online to run the

java version using pi4j and so I had to revert to this option. A number of ways exist to read and execute a python script as listed by this Baeldung tutorial. The ProcessBuilder class was the option that was opted for to execute the mcp3002_adc_value.py script which was also saved to the resources folder of the pump-controller project for history purposes. To read the code the following script was generated

```
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from gpiozero import MCP3002
>>> reading = MCP3002(0)
>>> reading = MCP3002(1)
>>> reading
<gpiozero.MCP3002 object using <weakproxy at 0xb645f180 to LocalPiHardwareSPIShared at 0xb64d9930>>
>>> reading.value()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'float' object is not callable
>>> reading.value
0.0014655593551538004
>>> reading0 = MCP3002(0)
>>> reading0.value
0.3287738153395212
>>> reading0.value
0.23009281875915977
>>> reading0.value
0.2437713727405959
>>> reading0.value
0.2437713727405959
>>> reading0.value
0.24181729360039084
>>> reading0.value
0.23790913531998048
>>> reading0.value
0.23497801660967266
>>> reading0.value
0.22813873961895448
>>> reading0.value
0.22716170004885194
>>> reading0.value
0.23009281875915977
>>> client loop: send disconnect: Broken pipe
```

Figure 2: Terminal Generated Code :)

5 MCP3002 and Thermistor Results

There exists a (roughly) 2.5 degrees celsius error between the converted MCP3002 ADC obtained temperature and the true reading of the temperature.

6 Ground

The RaspberryPi grounds may have some noise on them. To check this simply connect your multimeter to the various ground points of the RaspberryPi. The voltage between the grounds must be 0. If this is not the case then its time to play the illimunation game to determine which ground gives a voltage other than 0. The culprit will be the one prone to noise. Simply use another ground on the RaspberryPi.