MODULE 3: Using an Analog to Digital Converter

The Raspberry Pi does not come with an A/D converter like the Arduino does, but one can be added fairly easily. This lab will show to connect and use the A/D converter with a variety of sensors.

GOALS:

- Connect and use the MCP3008 A/D with the Pi
- Briefly discuss communication Protocols, SPI, I2C
- Review the use of the A/D converter along with the circuits required
- Use various sensors
  - Photoresistor
  - Temperature Sensor
- More python features: writing functions

A/D Converter

The Raspberry Pi although very powerful, lacks an onboard A/D converter. In order to gather data that is not binary in nature, such as a temperature, a device must translate the analog signal over to some binary format. Analog signals may assume an infinite number of values over a continuous range. These signals must be approximated to some binary number that the Pi can understand. A/D converters are designed with a specific precision in mind, the precision is given in terms of bits. Common values are 8, 10 or 12 bit converters. The number of bits defines the number of quantized levels the A/D converter will represent over a given range, so a 10 bit converter will translate values over 1024 different values from 0 to 1023 (2^{10} -1).

An example of a circuit used with an A/D converter is to the right. A simple voltage divider is set up with a resistor and a sensor with variable resistance such as a thermistor or photoresistor whose value changes with some property such as temperature or light. The connection of the two resistors is used as the input to the A/D converter as that voltage level will change as the resistance of the sensor changes. When the resistance of the sensor is low, the A/D converter will output a low binary number and when it is high, it will output a much higher value to the Pi.

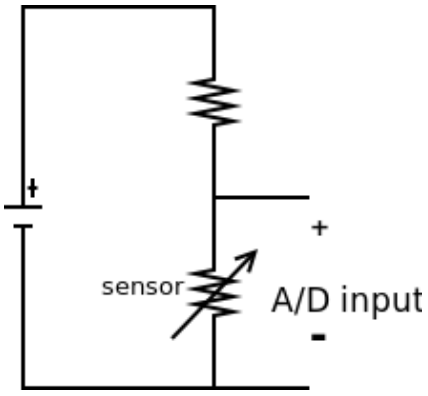

Illustration 1: Voltage Divider

MCP3008 Setup

The MCP3008 is a 10 bit A/D converter with 8 analog input channels that comes on a 16 pin chip. With 8 input channels, several different sensors can be used at one time. The chip connects to the Pi through the SPI interface (see note below). The following guide, although no longer supported, works fine for connecting the MCP3008 [1]. It is recommended that with the Pi 3 B+ that the hardware SPI be used with the following pin combinations [2].

MCP3008	Raspberry Pi
CLK (13)	SCLK (23)
DATA OUT (12)	MISO (21)
DATA IN (11)	MOSI (19)
CS/SHDN (10)	CEO (24)

Comment out the software SPI code and uncomment the hardware portion of the simpletest.py program before running the code. For testing, connect a few of the input pins of the A/D converter to high, low, a potentiometer

and let some "float". Notice that those connected high will be very close to 1023, those low to 0. Those that were allowed to "float" may have seemingly random answers, as their input is not fixed. The specification for the chip on its datasheet lists the maximum sampling rate for the converter at 75 thousand samples per second at the 3.3V supply voltage [3]. For many sensors, this will not matter as most physical systems do not change state that fast, however it is something to be aware of for systems that might need faster sampling.

NOTE The SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit) protocols are both well established methods of transmitting data from one device to another [4]. Both use serial (one bit at a time) connections and both can work with multiple devices over the same lines. As both are quite popular, the Pi has pins that are specifically allocated for those uses.

Photoresistor

A photoresistor is a resistor in which the resistance changes with exposure to light. With the photoresistor set up in a voltage divider circuit seen in Illustration 1, the voltage can be read by the A/D converter which can then be used to determine the amount of light the around the sensor. In general, the more light the less the resistance and the less light the higher the resistance. Beyond just telling if a room is dark or not, these types of sensors can be used for other applications, such as recognizing when the sensor passes a black object as opposed to a lighter colored object as more light will be reflected by the lighter colored object. The photo sensor can be used in a number of novel ways, a group in a previous project used it as a means of detecting the liquid level in a container as the amount of light passing through air is different than the liquid.

TMP36

The TMP36 is a solid state temperature sensor. The output of the sensor given by the following formula.

Degrees C = (Millivolts out – 500)/10

Instructions for connecting the sensor may be found online [5] as well as the datasheet for the device [6]. Note that on page 3 of the datasheet, the sensor does not follow the formula exactly. A sensor has a typical accuracy of +-1 degree C, and some of the devices can vary as much as +-3 degrees C.

Python Functions and Objects

It is often advantageous to make complex problems easier by breaking them up into smaller less complex problems. This concept is part of the motivation behind using functions when programming. As programs become longer and tackle more complex issues, it can be very helpful to use functions to help simplify the code development. An example of a function that accepts an integer reading from the A/D converter and returns the temperature in Fahrenheit is given below.

```
def convert_int2f (input):
   # Assuming, 3.3 supply voltage and reference voltage
   c = (input*3.3*1000.0/1023 - 500)/10
   f = 9.0/5.0*c+32
   return f
```

Then later in the program, this function may be called to print out the value.

```
>>> print ('Degrees F: %s' % convert_int2f(mcp.read_adc(analog_pin))
Degrees F: 66.83870967741936
```

Note that the `%s` stands for string output and is a placeholder for the data value that comes along after the closing quote in the print statement. In this case the data is coming from the call to the `convert_int2f` function, which is accepting data from yet another function call from `read_adc` which is a method attached to an object of type `Adafruit_MCP3008` which was imported. Review the simpletest.py program from link [1] for context. Objects are variables of a particular type, and in some cases, they have methods (functions) that can be called using the dot (.) operator. In this course it may be necessary to use objects, but it will not be required that students write their own object oriented code.

The following format prints a much nicer value for the temperature.

```
>>> print ('Degrees F: %f4.1' % convert_int2f(mcp.read_adc(analog_pin))
Degrees F: 68.0
```

In this case, the output is specified to be of type float instead of string with one digit after the decimal. This works fine if the output is expected to take four characters. It could easily be modified if it was expected to take more, as in the next example. Note the extra space since in this case the temperature did not require all five spaces.

```
>>> print ('Degrees F: %f5.1' % convert_int2f(mcp.read_adc(analog_pin))
Degrees F:  68.0
```

Conventions - OK, so I should have put this into module 1, but it is too late for that :)

Lines that start with:

`$` - something that is run from the command line
`>>>` - run from after invoking the python interpreter
`Number>` - indicates the line number for a program or section of code for later reference

File Transfer Tips

Git has been used to save code. This code can always be pulled down later for access on other devices. Other ways exist to transfer files from one system to another without using Git. Secure copy (scp) and secure file transfer (sftp) provide command line solutions for Mac or Unix/Linux users to transfer files to and from the Pi assuming the Pi has a network connection. WinSCP provides a convenient graphical interface for transferring files on a Windows system to the Pi. Other solutions such as using Google Drive, Drobox, etc. can also be used as ways to move files from one system to another. Become comfortable using one of the above methods to send and retrieve files from the Pi to other remote systems, which will make it much easier to continue to make progress when the Pi is not readily available.

MODULE 3: Using an Analog to Digital Converter

Homework

Prelab – Have this completed prior to showing up in the lab

1. Review links [1] & [5].
2. Download the simpletest.py program from link [1].
3. If the range of values of resistance of the photoresistor is from 1K Ohms to 10K Ohms for the readings that are of most interest, translate that over to 10 bit values with the following series resistors in the voltage divider circuit in illustration 1.
   a. 20K Ohms
   b. 10K Ohms
4. Write a function that will accept an integer value of 0-1023 and will return a value of bright, normal, dim or dark depending upon the value from the photoresistor input. Assume that the photoresistor is put in series with a 10K Ohm resistor as in illustration 1. The photoresistor has greater resistance for darker values of luminosity and less resistance for brighter values. For the purposes of this function, you may divide the digital values up proportionately.

Lab

- Measure the resistance of the photoresistor in various conditions of light using the multimeter. Why would it be incorrect to measure the resistance with the sensor connected to a circuit?
- Connect the A/D converter with a photoresistor and a 10K Ohm resistor as in Illustration 1. Verify that the circuit works using the simpletest.py program.
- Now use the TMP36 with another input. Use the function written from the prelab in the program. Make sure the program displays both temperature as well whether the lighting is bright, normal, dim or dark. You may want to adjust the values that trigger those luminosity outputs after some testing. If you modify simpletest.py to do this, make sure to note at the top of the program where the original code was obtained and which portion of the original was used.
- Write a schematic for the circuit from the previous exercise and also upload the program developed to Git as module3a.py.
- Use the photoresistor to detect a sheet paper with a 4 inch black stripe on it being slid under it at a rate of approximately 1 foot per second. The paper may have some random black markings and the black stripe might have some random white markings on it.
  - First determine values for the photoresistor over black and then over other surfaces such as the table and white.
  - Write a program that will turn on an LED when the black stripe is detected.
  - Write a schematic for the circuit required.
  - Save this program to Git as module3b.py.

Bibliography

[1] MCP3008, https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008, Accessed 7/15/19

[2] Raspberry gPIo, https://learn.sparkfun.com/tutorials/raspberry-gpio/gpio-pinout, Accessed 7/15/19

[3] Datasheet for MCP3004/3008, https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf, Accessed 7/16/19

[4] Introduction to I2C and SPI Protocols, https://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/, Accessed 7/15/19

[5] Overview of TMP36, https://learn.adafruit.com/tmp36-temperature-sensor/overview, Accessed 7/16/19

[6] Datasheet for TMP35/36/37, https://cdn-learn.adafruit.com/assets/assets/000/010/131/original/TMP35_36_37.pdf, Accessed 7/16/19