Lab 5: Interfacing Analog to digital Converter (ADC) with Microcontroller.
Report Due: Oct 21st
Demo Due: Oct 23rd

Learning Objective: In this lab, you are going to

1. Learn about interfacing analog sensors with controller using A/D converter.
2. Use of an ADC chip to analyse analog signals.
3. Examine the characteristics of a signal using oscilloscope.

Components Needed:
Pi 3 Board, resistors, temperature sensor, ADC chip MCP3008 and light sensor.

Programming requirements:
Use C or Python to complete this Lab.

Lab Activities:

1. **Introduction to Raspberry Pi' SPI port:** SPI, or Serial Peripheral interface is a communication bus that is used to interface one or more slave IC (integrated circuits) to a single master device (Raspberry Pi is the master in this lab). In lab 4, though you have used the I2C port as output, the SPI port could have been used instead. Please note that SPI is a faster bus than I2C. On the other hand, I2C can connect many devices with only a 2 wire bus, while each slave device for SPI requires an additional wire bus. There are three SPI wires shared by all slave devices on the bus:
    a. Master in slave out (MISO), data from slave to master on this port
    b. Slave in master out (MOSI)
    c. Device clock (CLK)

    Each slave device will have one additional port connected to the master device. This port is for selecting a device to communicate with. For Raspberry Pi, there are two available ports for you to connect with your slave devices. Once you connect devices, data transmission can happen. During each clock cycle, the master sends data on the MOSI line and the slave reads it. At the same time, the slave sends data on the MISO line, and the master reads it. This behavior is maintained even if only one line has data to transmit.

    Question 1: Summarize the difference between SPI and I2C ports. Explain in what situation using the SPI ports is better than the I2C ports, and vice versa.

2. **Introduction to MCP3008 ADC:** Your Raspberry Pi has no built-in analog inputs. This means it can not read any data from analog sensors. The MCP3008 is a 10-bit, 8-channel, SPI-based ADC (analog to digital converter). It

communicates with the Pi using the SPI bus on the Pi's GPIO header.

The following table shows how you can connect your Pi to the chip:

```
VDD         3.3V
VREF        3.3V
AGND        Ground
CLK         GPIO11(SPI_SCLK)/Phys23
DOUT        GPIO9(SPI_MOSI)/Phys21
DIN         GPIO10(SPI_MOSO)/Phys19
CS          GPIO8(SPI_CE0)/Phys24
DGND        Ground
CH0-CH7    Analog Input(8 channel)
```

When connected to Raspberry Pi, the ADC chip will take analog input from Ch0-Ch7 and provide 1/0 digital signal to the Dout pin. The following communication happens when the Pi tries to read from the ADC chip,

1. The Raspberry Pi will first send a byte containing a digital with a value 1. The MCP3008 will send back its first byte, which is not important, as a response.
2. Then the Raspberry Pi will send a second byte to indicate which channel on the MCP3008 chip should receive the analog signal.
3. As the result will be a 10 bit data, which can not be held by a single byte. MCP3008 will send back the second byte, which contains two bits of the conversion result (which is the 8th and 9th bit).
4. The Raspberry Pi then sends a response byte to indicate that the previous byte was received, and then MCP3008 sends back the last byte containing the rest of the bits (bits 0 to 7) of the converted digital value of the analog signal.
5. Finally, Raspberry Pi merges bits 8 & 9 with bits 0 to 7 to create the 10 bit digital value from the conversion.

Question 2: What are the various types of ADCs in use? Which type of ADC is MCP3008 and what are its advantages/disadvantages?

3. **Reading Data from a light sensor:** We will use a photocell (CdS photoresistor) as an example to demonstrate how to read data from light sensor through MCP3008. Under the normal light condition, the resistance of the photocell is about 5-10KΩ, and in the dark it goes up to 200KΩ. We can use it just like we use a normal resistor.

   We can connect one side of a 10k resistor to the power of 3.3V. (To protect the chip, make sure you connect to the 3.3V pin) Then the other side of the 10K resistor to two wires, one connected to the photoresistor and then to the ground, the other connected to the input pin of MCP3008. With bright light in the room, the resistance will drop to a low value. This will cause higher current to flow through the photoresistor, resulting in a voltage drop across it. The input voltage will drop to almost 0V. In the dark,

the voltage will go up near 3.3V due to the high resistance in the photocell relative to the resistance in the resistor.

To read the data, following Python script can be used:

```
import spidev

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0) #1st param is channel

def readChannel(chan):
    adc = spi.xfer2([1,(8+chan)<<4,0]) #sends 3 bytes: 1,
(8+channel)<<4, 0
    #adc will contain 3 bytes (as many as sent)
    print(adc)
    data = ((adc[1]&3) << 8) + adc[2]
    print(data)
    return data
```

Activity 1: Use your Raspberry Pi to read data from a light sensor and a temperature sensor **(Two sensors should connect to different channels and then use software to read data one by one from the correct channel).** Display the current readings in the command window.

For the temperature sensor, the temperature range is approximately -50°F-280°F corresponding to 0-3.3V. For the lighting sensor, we will use a scale of 0 to 100 which corresponds to 0-3.3V. You can find the temperature sensor datasheet here.

4. **Building a simple raspberry Pi oscilloscope with a MCP3008 sensor:** In the previous activity, you have accomplished how to read data from MCP3008 and to process that input data with Raspberry Pi. In this activity, you are going to build a more complex system, a simple mini-oscilloscope using your Raspberry Pi with the MCP3008. The simple oscilloscope will have two functions:
    a. **Recognize a wave**: This is a very basic feature that all oscilloscopes should have. In this lab assignment, you can pick one of the two options to implement:
        i. **Visualize the waveform on a opening window using python GUI**: You will have to continuously sample the input wave, and project the data onto a plot just like the oscilloscope you used in the lab4. The system should keep sampling and plotting, and input should be allowed to switch between the square, sine and triangle waves while the system is running.

The plotting should be paused when system is processing the received data.

ii. **Display the name of the input wave on a command window:** You should implement algorithm to distinguish square, sine and triangle waves based on the data you sampled. Every time your algorithm detects a shape change, you print out the name of the shape to the command window. There will be noise when sampling the data.

b. **Characterizing waveform:** The mini-oscilloscope should be able to find out the Frequency for the input waveform and display it to a command window or with your visualized wave.

Activity 2: Build a simple oscilloscope that will meet the requirements as above. Use lab4's output as input in this activity. Two teams (no more than 4 people) are allowed to work with each other to provide at least one working prototype and demonstrate to the TA before the deadline.

Question 3: What is the sampling rate? What is the sampling rate for your mini-oscilloscope?

Question 4: Remember, never use the same raspberry Pi to do a waveform generation and a waveform recognition at the same time. Otherwise, you will generate a waveform that frequency keeps changing or get random readings from the MCP3008. Explain why this is the case.

Question 5: It is highly likely that your sampled data contains lots of noise. How you can filter the noise? Explain your method.