CSCI 250 Python Computing: Building a Sensor System
TR 12:30-1:45 and 2:00-3:15 – MZ022 – Spring 2018
Lab 2: Light with Extensions … Dictionaries, Graphing, and More
Wendy Fisher

Corresponding Learning Outcome:

- Develop and run basic Python functions and programs in the Linux environment to collect data from sensors using the Raspberry Pi Hardware (e.g., optical, acoustic, acceleration, magnetic field).

  During this lesson, students will learn how to:
  - Connect and use the ADC and continue forming an understand the basics of using the breadboard
  - Interface with the General-Purpose Input/Output (GPIO) pin to control ADC and Sensors
  - Create python script and use secondary file as a custom library for reading the ADC

  Remember: This lab will have less instructions than the previous and the next several labs will become progressively more difficult and require more independent programming and resourcefulness of looking at specification sheets for the sensors, etc.

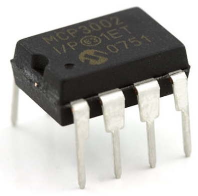
Purpose:
The purpose of this lab is to set up the Analog to Digital Converter (ADC) Chip and an introduction how to use additional sensors (e.g., Mini Photocell) on your Raspberry Pi to collect data.

Hardware: Let's begin:
1. In addition to the equipment you need to get your Raspberry Pi up and running and the items used in the previous lab, collect the following equipment from your box. This is all you will need for the main part of this lesson (if you do the extra credit, you determine what is needed).

   Note: it is good practice to discharge yourself before handing electronics; you can easily do this by touching a grounded object.

   You can use the links to get additional information on these items – features and documents (datasheets and hookup guides, and some have tutorials):
   a. Analog to Digital Converter Chip: https://www.sparkfun.com/products/8636

   b. Mini Photocell: https://www.sparkfun.com/products/9088



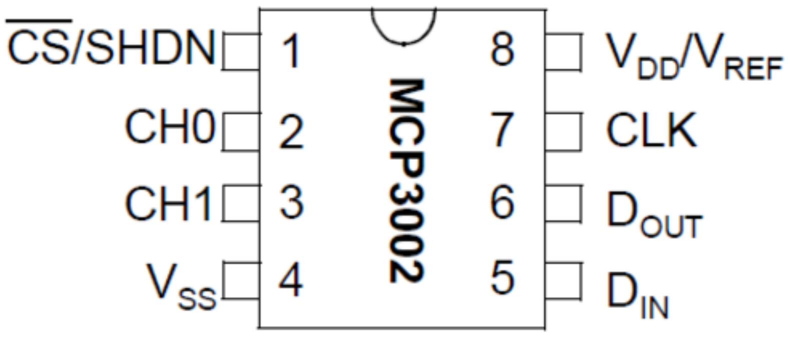
2. **Connect and Wire up the Analog to Digital Converter Chip** – a few things to keep in mind:
An ADC allows a microcontroller (which is usually all digital signal based) to connect to analog sensor that reads in an analog voltage (an analog voltage is sinusoidal). Analog voltage is good for sensors since it's continuous therefore the sensor will continually update with time at any frequency we want.

   This lab will walk you through hooking up the ADC and testing a couple of our sensors to get used to how to make these circuits. We provide most the ADC related code for now and you will include it as a library function (see software section).

   a. The way to install the component onto the breadboard is to straddle the chip over the middle rail of the breadboard such that four legs are on one side and the other four are on the opposite (think about why this is needed). Ensure the chip is seated well (pushed all the way in).
   b. To follow the chip numbering, you need to locate the semi-circle on the chip, which designates the front of the chip (see the table and image below).

ADC Pin	Description	RPi Wedge Connection
1	**CS = Chip select** We choose what channel the ADC communicates with. The RPi gives 2 choices, labeled CE0 and CE1 on the wedge.	**CE0 OR CE1 (use CE0 for this lab)**
2	**CH0 = ANALOG TO DIGITAL INPUT** Where you would place your pin you want to convert from analog to digital (i.e., where you hook up your sensor too).	**ANALOG INPUT 0 (use this one for this lab)**
3	**CH1 = ANALOG TO DIGITAL INPUT** See pin 2 description.	**ANALOG INPUT 1 (leave empty for this lab)**
4	**V_{SS}** We need to connect GND here to complete the circuit to not blow up the chip!	**GND**
5	**D_{IN} = Serial Data In** The SPI port serial data input pin is used to clock in input channel configuration data*.	**MOSI**
6	**D_{OUT} = Serial Data Out** The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place*.	**MISO**
7	**CLK = CLOCK PULSE PIN** Hookup this pin to SCK (stands for Serial Clock) of the wedge. In order to convert analog to digital signals, the pi and the ADC chip need to be "synchronized" through a clock signal.	**SCK**
8	**V_{DD}/V_{REF} = REFERENCE VOLTAGE** Hook up voltage here. 3.3V for this lab.	**VOLTAGE (3.3V)**

** Reference datasheet for full info: http://ww1.microchip.com/downloads/en/DeviceDoc/21294C.pdf*



3. **Hook up the Mini Photocell sensor** – this is a simple connection with:
   a. One leg to the power rail.
   b. The other to the ADC analog input pin for reading the values collected from the photocell (you can use either CH0 or CH1) – we will use CH0 for this lab.
   c. **Resistance**: you could run without one or simply add a resistor between the power rail and the photocell that you have in your kit.

Software: Time to code it up:
Again, we want to make sure the entire class is on the same page and has a solid foundation, so we will walk through this together. The next four labs will become progressively more difficult and require more independent programming. The following template should be used for your new python file:

1. Open a terminal and **Install the SpiDev library** – type these commands (remember Linux Day!?)

```
sudo apt-get update
sudo apt-get -y install git python3-dev python-dev
git clone https://github.com/doceme/py-spidev.git
cd py-spidev/
sudo python3 setup.py install
```

2. Configure the RPi to allow us to use the ADC – **Enable the SPI and I2C**. These are interfaces that allow us to read values from the sensor using the ADC.
   a. GUI: Menu, preferences, raspberry pi configuration (interfaces) OR
   b. Terminal: sudo raspi-config (advanced).

3. Test your library is loaded correctly before continuing – use the terminal or interactive mode to simply try to import the library (e.g., **import spidev**) … if no error, you are ready to continue.

4. Download the **reader.py** file from Canvas (you will include this as a library and call the read function). Save this file in the same directory that your new python file will be located in (see #4 below).

   We provide this code since the SpiDev uses some concepts that are ahead what we have learned so far but needed for our ADC; below is some basic information - we will expand on this in later classes.

```
#we will just be calling the defined read function with a zero:  (e.g., read(0)) for our lab
spi = spidev.SpiDev()                   #initiate spi object
spi.open(adc_channel, spi_channel)      #adc channel = 0 (can use 1) and spi channel = 0.
spi.max_speed_hz = 1000000              #is 1.0 MHz can change the value but this worked best.
```

5. Open your IDE, create a new python file, the below is a template if you need it, and save it as light.py (or something you deem worthy).

   a. Include the proper libraries (see the template below if needed).

   b. Using a for loop, call the "read" function from the provided reader.py file and collect 500 values from the Mini Photocell and print the to the screen.

You should practice incremental coding and print a few simple output to see if you are getting any values, then you can ramp up to all 500, etc. They should be between 0-1 depending on the lighting in the classroom – my experiments at home were all around 0.62 – 0.94.

c. Experiment with covering and/or shining a flashlight on your sensor while your program is executing, to see how the values fluctuate.

d. After verifying your collecting the appropriate data, store the values into a numpy 1D array.

e. To slow your program down a bit and allow plenty of time to get light on/off oscillations, you can either: leave in the print statements or add in a sleep statement in between reads.

```
#Author:
#Date:
#Description:

#import libraries:
        #numpy for arrays
        #time library to use the sleep command
        #reader (our custom library for the read function to get values from the ADC)

#Looping – this is where you will create an array and loop through collecting 500 values using
reader.read(0)
```

*** Notice, we are not using the GPIO commands for this lab – we are going through the ADC*

When completed, if you are still in class, let us know if you want to show us a demo of your circuit!

<u>Submission Details (100 total points) – upload the following on Canvas:</u>
(50 pts) Picture of your completed circuit, *with ADC and photocell.*
(50 pts) Python Code with for loop control and storage into array.

<u>**Creative Extensions**</u> … it is up to you to figure out how to add the following for possible extra credit, you can reference sparfun.com or raspberrypi.org and/or search on the internet… have fun and be creative! **Remember, these have little instructions … allowing for creativity and are purposely vague:**

1. (10 pts) Every element in the array should be multiplied by 100 and rounded down to 2 decimal places and converted to a string

   EX) code: x = “%.2f” % 1.1234 WOULD ASSIGN STRING X to 1.12.
   Place these values in a dictionary to read like following:

Key	Value
Reading 1	89%
Reading 2	75%
Reading 3	80%

   And so on to reading 500 or whatever

Display dictionary nicely by importing json library and using the “dumps” function.

2. (10 pts) Provide a graph of the data.

3. (10 pts) “Interface” the program to not read sensor values unless the user presses a button (and has a green light while program is running and a red light while the program is not running).

4. (10 pts) Crazy extra extension: Add a temperature Sensor (or any other of our sensors to the circuit). Interface the circuit so you can choose to run one sensor, the other sensor, or both (with 3 buttons and three lights to show which mode is on (one sensor, the other sensor, or both).