Intro

Welcome to Lab 2! In this lab you will begin connecting your SmartPlug platform to a variety of sensors. We will be exploring the differences between analog and digital, as well as how to account for those differences in your project.

Your lab objectives for this session are:

- Integrating passive sensors to the Raspberry Pi
- Using Python Libraries
- Working with the differences between digital and analog for the Raspberry Pi

MCP3008

The MCP3008 is an Analog to Digital Converter (ADC) which is able to read analog signals (for example, signals received from sensors) and output a digital signal. This is very useful since the RPi does not have an onboard ADC. Other prototyping platforms such as Arduino have this functionality integrated, but since the Raspberry Pi's GPIO only supports digital signals, we must rely on an external ADC to read any sort of analog inputs.

We will be using Adafruit's MCP3008 Library for this project. To install, navigate to your SmartPlug repo on your RPi and run:

```
sudo apt-get update
sudo apt-get install git build-essential python-dev
git clone https://github.com/adafruit/Adafruit_Python_MCP3008.git
cd Adafruit_Python_MCP3008
sudo python setup.py install
```

This will clone Adafruit's MCP3008 library from GitHub and into your own repository.

Next, you must enable SPI, the MCP3008's communication protocol of choice. To do this, simply run:

```
sudo raspi-config
```

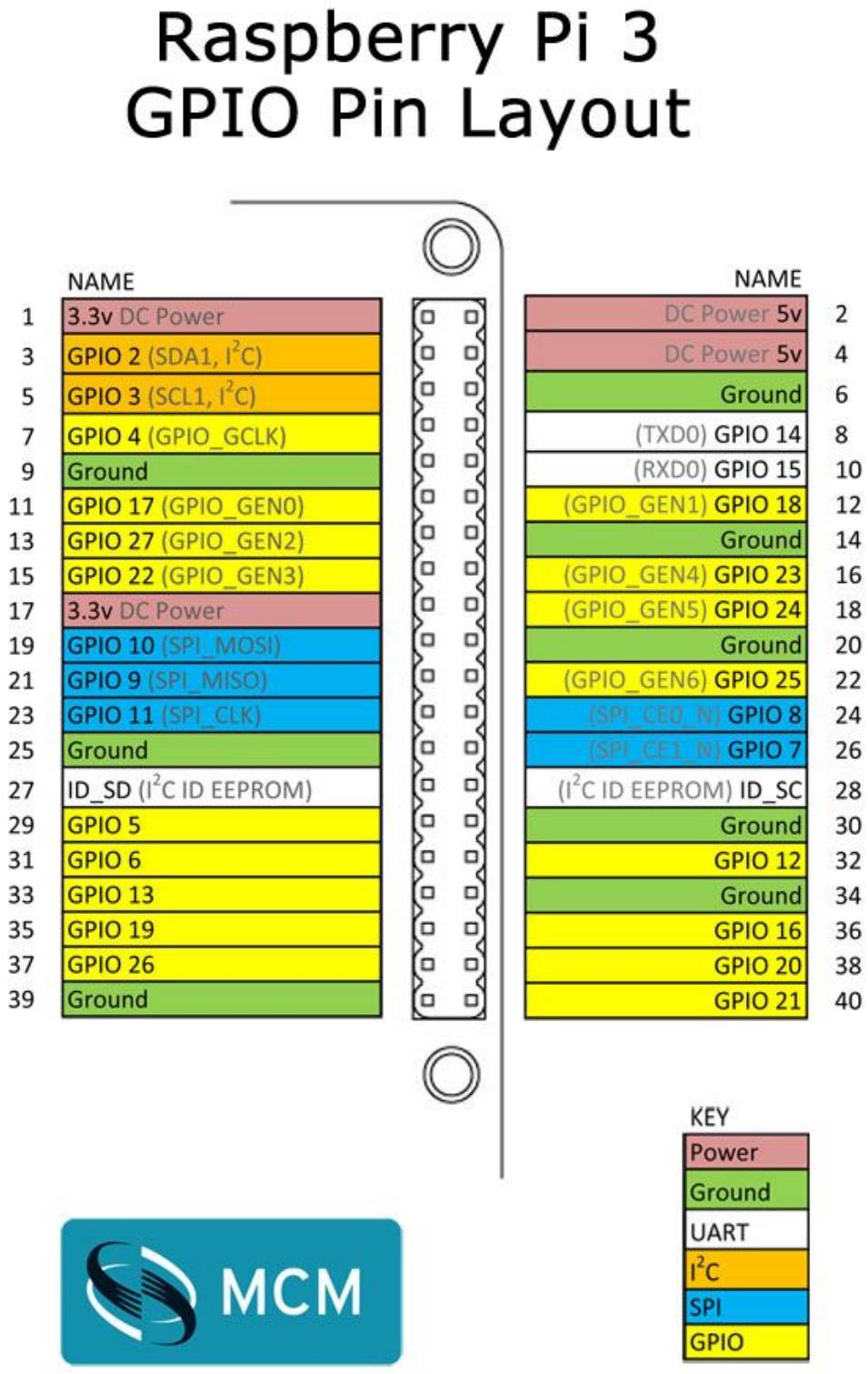
Once prompted, choose 'Advanced options' and then select SPI and enable it.

Next, navigate to the Adafruit Library's folder and edit examples/simpletest.py so that you comment out the 'Software SPI Configuration' and uncomment the 'Hardware SPI configuration'. It should look like this:

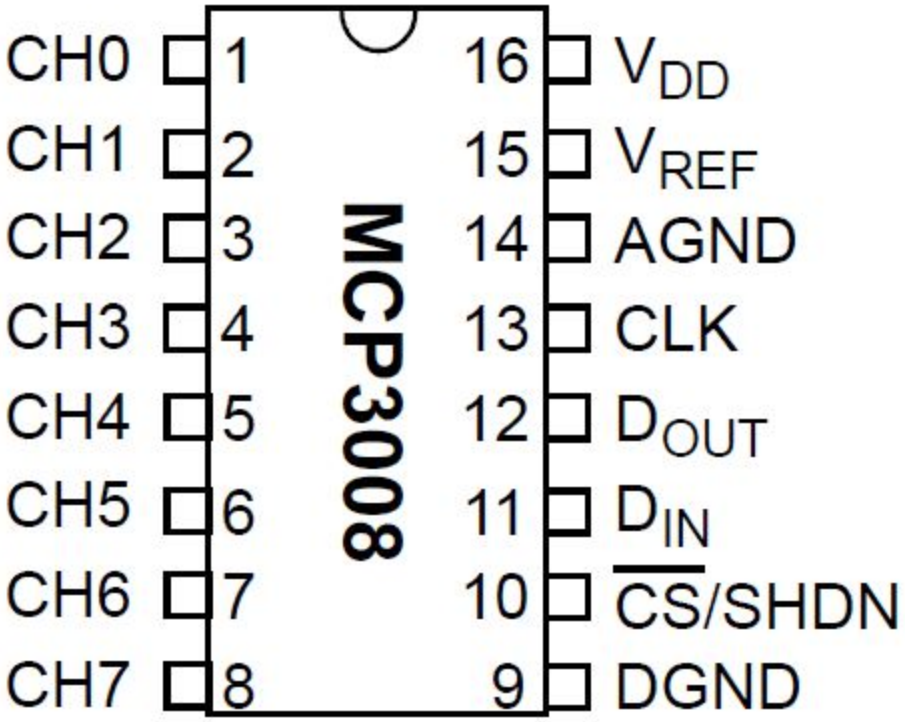
```
# Software SPI configuration:
# CLK  = 18
# MISO = 23
# MOSI = 24
# CS   = 25
# mcp = Adafruit_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)

# Hardware SPI configuration:
SPI_PORT   = 0
SPI_DEVICE = 0
mcp = Adafruit_MCP3008.MCP3008(spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE))
```

Now comes the fun part! We will be wiring the RPi's GPIO pins to the MCP3008. The RPi's pins are detailed in the following diagram:



The MCP3008 is as follows:



Using your breadboard and jumper wires, connect the two components as such:

MCP3008	Raspberry Pi
VDD	3.3 V
VREF	3.3 V
AGND	GND
DGND	GND
CLK	SPI_CLK (Pin 23)
DOUT	SPI_MISO (Pin 21)
DIN	SPI_MOSI (Pin 19)
CS/SHDN	SPI_CE0_N (Pin 24)

This configurations connects the MCP3008 to the RPi's SPI pins. SPI is a four wire communication protocol that is widely used everywhere, from hobby electronics to industry. SPI allows the MCP3008 to send its processed data back to the RPi, working as a middleman between the RPi and sensors.

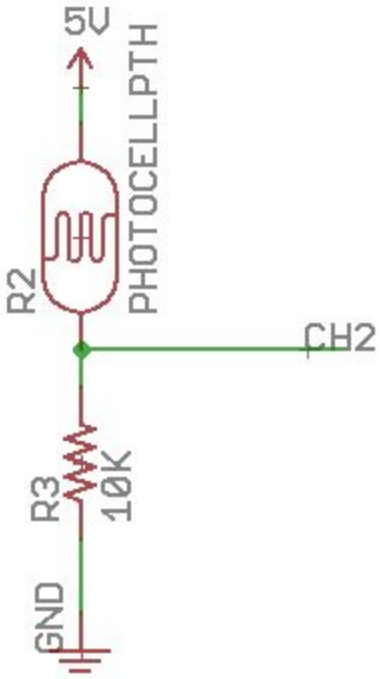
You can read more about SPI at:
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
And MCP3008 at:
http://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf

We are connecting the MCP3008 to a 3.3V power supply since it is the RPi's native logic level. **Dealing with logic levels is incredibly important!** Sending in one logic level but reading in a different level will cause the signal to be misinterpreted. You can read more about logic levels here:
https://learn.sparkfun.com/tutorials/logic-levels

The remaining CH0-7 pins on the MCP3008 are used to connect the analog input signals.

Connecting one sensor

We will connect our first sensor, a photocell resistor, to channel 2 on the MCP3008. Connect the components as follows:



The circuit provided above is a voltage divider, where the input is 5V and the output is the CH2 of the MCP3008

The datasheet for the photocell can be found here:
https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/SEN-09088.pdf

Based on that information, we know that the resistance range of the photocell is 8~20K ohms.

1) Is the datasheet correct for your current environment? Measure the actual resistance using an ohmmeter. Find the range of values in your readings.

2) Given our input voltage and our resistor values, what is the voltage range for the output of the voltage divider circuit?

Based on the MCP3008's datasheet (provided in the prior Milestone section) we know that it is a 10-bit ADC. This means that it will output values from 0 to 2^{10}-1, or 0 to 1023.

3) Given the voltage range of our photocell circuit, and the range of our ADC, what are the possible values that the MCP might output from channel 2's input?

Once completed, run the simpletest.py file inn the Adafruit Library and observe the results!

4) Were they within your calculations? Why or why not?

Spoiler alert: The provided photocell circuit is wrong.

5) What is the problem? Find the problem with the circuit, fix it, calculate the voltage ranges again, and check them against the MCP output.

We will use the MCP3008 for other analog signals, such as the Sound Sensor. You are more than welcome (read: encouraged) to add any other sensor you want! Just make sure you don't have the same problem we solved in this section.

DHT22

We just connected an analog sensor to the Raspberry Pi using an Analog to Digital Converter. Thankfully, not all sensors require an ADC to work with the Pi, some are able to communicate directly via the digital pins. The DHT22 sensor is a great example of these types of sensors.

The DHT22 is a Digital Humidity and Temperature sensor that communicates via a single wire, using a specific communication protocol. Therefore, a library is needed to use this sensor. We choose Adafruit's library.

The specsheet for the sensor can be found at: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

First, let us get the Adafruit code:

```
sudo apt-get install build-essential python-dev python-openssl
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_MCP3008
sudo python setup.py install
```

Note: It might be the case that you already have one or more of these installed (and up to date) already. Just ignore any message that might suggest such.

Time to wire the circuit! Connect the pins as follows:

DHT22	Raspberry Pi
VCC	3.3V or 5V
DATA	GPIO 4
NC	NULL
GND	GND

IMPORTANT: Additionally, you need a 10k resistor between VCC and DATA pins.

Navigate to the examples directory in the Adafruit repo to test the connection. AdafruitDHT.py takes in the sensor number and the GPIO pin of the sensor and returns the temperature and humidity values. Run the following command:

```
sudo python AdafruitDHT.py 22 4
```

(Since we are using DHT**22** and GPIO pin **4**)

Your output should be something like

```
Temp=27.6*    Humidity=40%
```

Congratulations! You have finished wiring the digital passive sensor.

Libraries

You will be writing two classes for this section. They will be called SmartDHT22 and SmartMCP3008. The purpose of these classes are to provide a simple interface that abstracts the complexity of using the Adafruit libraries. **You will not be re-writing the provided Adafruit libraries**, you will be making a simple wrapper for them.

You must use Python OOP standards for this assignment. If you forgot how this works, refer to the lab 1 tutorials.

These are the specifications for the Libraries:

SmartDHT22

Filename: SmartDHT22.py
Functions:

- get_temp_celsius()
    - Gets the temperature in Celsius. Returns just an int.
- get_temp_fahrenheit()
    - Gets the temperature in Fahrenheit. Returns just an int.
- get_humidity()
    - Gets the humidity. Returns just an int.

Additionally, the constructor must accept the GPIO pin number.
Any additional functions are allowed, though they will not be tested or corrected.

SmartMCP3008

Filename: SmartMCP3008.py
Functions:

- read(pin_num)
    - Reads data from a specific channel on the MCP3008. pin_num is the channel number.

Any additional functions are allowed, though they will not be tested or corrected.

Next, you will write a main.py script that uses all of the functions you wrote, in that order. Assuming the photocell is connected to channel 0, you will just run `SmartMCP3008.read(0)` for the SmartMCP3008 section.

This script will poll the sensor data once per minute, for an hour. This means you will have a total of 60 readings for each sensor. You must store the sensor reading as well as the time of the reading. Graph the data with your tool of choice, readings vs time, and describe the

environment it gathered data from. You can be cool and proactive and use one of the many graphing libraries in the python world (such as http://matplotlib.org/) or you can be boring and use Excel.

Make your repository from Lab 1 private. This can be done for free using your UCSD email address and following this address:
https://education.github.com/pack

The GitHub pack contains a large amount of fantastic resources for you to use, free!

Turnin Instructions

Write a lab report answering the problems highlighted in Green.

- Include all calculations
- Include all the data
- Graph the data.

Submit your code to your (now private) GitHub repo, and add your TAs as collaborators.

Be sure to include all of the team member's names and PIDs on top of the document.