CSCI 250 Python Computing: Building a Sensor System
TR 12:30-1:45 and 2:00-3:15 – MZ022 – Spring 2018
Lab 5: Magnetism
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 collect data from the Insulated Reed Switch that detects a magnetic field.
  - Reference hookup guide and information from Sparkfun's website.
  - Practice Python coding and concepts: object-oriented code design with classes and methods.

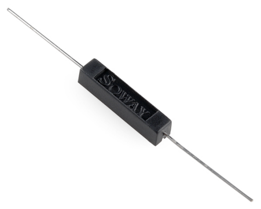
Let's begin:
1. Collect the equipment.
   The first thing to do is make sure you have the proper electronics ready … like following a recipe. Know what you are working with and the supplies you need. As you move forward into the Capstone, knowing how to find information about your various sensors is key.

   Reference your What's in the Box worksheet and SparkFun's website to gather the following items:
   Insulated Reed Switch - https://www.sparkfun.com/products/10601

   Magnet Square - https://www.sparkfun.com/products/8643

   LED

   OPTIONAL - Hall Effect Sensor (extension) -
   https://www.sparkfun.com/products/9312



2. Breadboard the basic circuit using an Insulated Reed Switch to detect magnetic fields.
   Reference the actual chip and the hookup guide, you will quickly learn how to wire-up the components on your own.

   Description: Our reed switch is insulated (unlike the glass one in the photo) … and is "normally open" … in other words; it starts off as open, disallowing flow. When the body of the reed switch is exposed to a magnetic field (our magnet), two ferrous materials inside pull together, the connection closes, and current can flow. In absence of a magnetic field, the reed switch (and the circuit) opens, restricting the flow.

   Connect one end of the reed switch to 3.3v and the other to a GPIO pin*.
   * Resistance – it is recommended to use a pull-up resistor for the reed switch to bias the switch to high and output a False or 0 – and when the switch closes, it will output True or 1 - we will accomplish this in code using the Raspberry Pi (see below #3).

Part of our information is directly from the hookup guide for the Reed Switch on Sparkfun's site:
https://learn.sparkfun.com/tutorials/reed-switch-hookup-guide?_ga=1.215061335.137750560.1486666660

3. **TASK 1**: Basic data collection and testing your setup.
    a. This initial circuit will output a simple True or False based on its detection of a magnetic field – and is all digital based, so we can just use one of the GPIO pins for reading input from the sensor.
    b. We will not need to install any additional external libraries for this lab.
    c. Resistance – as discussed in #2, it is recommended to use a pull-up resistor for the reed switch to bias the switch and output a False or 0 – and when the switch closes, it will output True or 1 - we will accomplish this in code using the Raspberry Pi - the GPIO pins all have "internal pull up/downs" that can be controlled in code by:

        Adding the following 3rd parameter to the **GPIO.setup()** command: **GPIO.PUD_DOWN**
        Then, when reading the pin using **GPIO.input()**, we get a True - 1 OR a False - 0.

    d. Before we implement a more "object oriented design" and reading our data using objects and a class, let's do some initial code. Write code to collect and print the collected data (you should get 0s or 1s) – download the template file: **MagneticLab5.py** from Canvas in Files->Labs - (you will add code to this file, turn it in, and then modify it with the object-oriented changes in #4 :-).
    e. Update MagneticLab5.py:
        i. Fill in the header information with your name, the date, and a better description of the lab.
        ii. Read from the circuit you just built and turn an LED on/off when you sense the presence of the magnetic cube.

4. **TASK 2**: Object-oriented code – detecting the speed of a magnet moving in a circle – the speedometer.

    Overview: one useful thing that magnetic sensors can be used for in projects is to be a speed detector. Though it can detect linear speed, they work very well when detecting speed of rotating object. One project idea we have seen is a "Speedometer for a bike," where the programmer essentially had a reed switch detect a magnet rotating around a bike wheel and used some basic physics and multi-threaded programming to calculate the speed the bike was moving at.

    For this task, we will make a very simple revolution detector – and just **manually rotate** our small magnet around our sensor (**YES** we are pretending a constant radius of rotation for today :~) … for an extension, you can find someway to create the rotation – this code will serve as a proof of concept.

    New idea!! For our speed detector we will have 2 "threads" (or a better word is processes). We will want to do something called "**multi-threading**" in this lab so we can do two things at once: the speed calculation and updating our timers for elapsed time. Multi-threading means you can go through multiple pieces of code SIMULTANESOULY. Thus far, all of our applications have been "single-threaded". Usually, that is enough; sometimes multi-threaded programming will be useful – so we will practice it for this lab.

    We use the basic data collection code from #3 to use the concepts we have been learning in class:
    a. Save the file from #3 to another name – you will be turning in both (e.g., **MagneticLab5_OO.py)**
    b. Make a class for your Speedometer to store things such as the elapsedTime, startTime, speedMPS, totalDistance, pulseCount, etc.

        YOU can code this however you want – be creative:

        HINT: To capture the current time, you can use time.time()

c. Your class should contain methods for (atleast):
   i. **__init__(self):**
   ii. **__call__(self,channel):** method – this is similar to the __init__ method that is called when our signal falls from 1 to 0; we need this method to create a second process or thread which updates the pulse count and other attributes you determine need updating for each revolution.
   iii. **calculateSpeed(self, radius_cm):** takes in the radius in centimeters, performs the calculation, and updates the speed in meters per second: speedMPS, which is the speed of the rotating magnet (**Note**: incoming radius is currently a made up value until you extend the program with a known rotating object and size). This method should be called during each iteration of our while loop to re-calculate the speed in real time.

      Suggested Formula: V = 2 **π** r / elapsedTime

   iv. **printData(self):** method that prints the current values for: speedMPS, totalDistance, and pulseCount (or whatever information you feel is valuable).

d. Instead of getting data using the **GPIO.input()**, we will implement something new … it is called a callback function: **GPIO.add_event_detect()** – summarized below and here is an online resource example: https://sourceforge.net/p/raspberry-gpio-python/wiki/Inputs/

   GPIO.add_event_detect(pin#, GPIO.FALLING, callback = SpeedometerObj, bouncetime = 25)
   - Pin# - pin number of your sensor (reed switch)
   - GPIO.FALLING – determines if you want to detect RISING 0-1 or FALLING = 1-0
   - SpeedometerObj – the name of the object you created – the callback will call (envoke your __call__ method automatically).
   - bounceTime – optional parameter – time in ms, good to use to give a little to provide the circuit "time" to detect the RISE or FALL. You can experiment with different values – we tried 15, 25, etc.

e. When done with the creation of your class, use it! For example, from the main area:
   i. Create a **Speedometer** object.
   ii. Setup a callback function to collect the data and invoke the __call__ function using a separate thread or process (Wait, what? This is new … make sure you read c. above :~).
   iii. Have a LOOP that…
      1. Calculates and prints the speed for a given radius using your speedometer object and the calculateSpeed() method.
      2. Prints the current values of the data using your printData() method.
      3. Sleeps for a bit to slow things down.

Submission Details – upload the following on Canvas:
(20 pts) Picture of your completed circuit.
(40 pts) Python Code for basic lab (MagneticLab5.py file).
(40 pts) Python Code with modifications for object oriented (MagneticLab5_OO.py file).

Creative Extensions … 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!

(10 pts each) code/photo for:

1. Use this program on something that spins, measure it's radius, and run it.
2. Use a Hall effect sensor in the same way we used the reed switch (maybe?).
3. Display your understanding of multi-threading by creating your own program. It can be as simple as you want (maybe just multi-thread between two buttons).