SensorEducation Workshop

Version 0.1
Jacob Ulasevich
Sawyer Coleman
Erin Harding
Hans Mok

Introduction

The goals of this lab involve getting a basic understanding of circuitry with the use of a Raspberry Pi as well as a breadboard. Students will also gain experience working as programmers through the use of Python. Those going through this lab require no prior knowledge of wiring or programming and all instructions will be simple enough to follow for those with no past experience.

Python

Experiment 0

Python is a powerful programming language that has an easy syntax making it a fantastic language for beginner programmers. In this exercise you will learn how to download Python and use the software IDLE to code your programs.

1. Go to https://www.python.org/downloads/
2. Select “Download Python 2.7”
3. Choose where you want Python to be saved on your computer.
4. While the installation is going on it may be best to go ahead and read over the rest of this document to get an idea of what you will be working on.
5. Once the install is finish in your search bar look for the program “IDLE” and open it.
6. To start creating a new program simple go to File, then “New File”
7. Congratulations you are ready to begin coding!

Variables

Experiment 1

1) Variables are one of the most important items in programming, they allow you to store and manipulate data. While there are many different types of variables, we will go over four of them:
    a) Integer
    b) Double
    c) String
    d) Boolean
2) In the top left corner of your Raspberry Pi Desktop select the dropdown menu > programming > Idle 2.7. Idle is the software that allows you to develop your code and run your program. You will be using that for a majority of this workshop.
3) Under File select “Open” and navigate to SensorEducation where you will select “Variables.py”
4) To assign a variable, you put the name on the left, an equals sign, and whatever data you want on the right.
5) The below pieces of code assign 4 different variables with 4 different types of data.

```
#The following is an INTEGER variable.
a = 5
#The following is a DOUBLE variable.
b = 3.14
#The following is a STRING variable.
c = "Hello World"
#The following is a BOOLEAN variable.
d = False
```

6) What are some differences you notice?
    a) Integers are whole numbers.
    b) Doubles can contain decimals.
    c) Strings are collections off characters (including numbers) and must be put between quotations.
    d) Booleans are either true or false.
7) Now that you have some knowledge of variables assign your own and create a print statement then run your program.
8) You can add some variables to assign data to a new variable, however it is important to note you cannot add a string with a number, but what happens when you add an integer with a double?
9) Assign a “double” number to x, an “integer” number to y, and sum them on the right of the z declaration

```
#Make x a double variable
x =
#Make y an integer variable
y =
z =
print("Result: " + str(z))
```

10) Run your program and analyze the results.
11) Our last variable exercise will deal with adding two strings together. Assign a string data type to the two variables below then run your program, what is the result?

```
#Assign stringOne a word or sentence
stringOne =
#Assigne stringTwo a word or sentence
stringTwo =
print("Result: " + stringOne + stringTwo)
```

Variables are the foundation of computer programming. Without them it would be very difficult to get the computer to understand what you are trying to accomplish. Be sure to keep an eye out as we manipulate more variables throughout this workshop.

Loop

Experiment 2

1. A loop in programming is a certain area of code that gets ran over and over again until something tells it to stop. Think of it like Newton's law. A loop in execution will stay in execution until another bit of code tells it to stop.
2. As we discussed about variables, there are different types of loops but this workshop will only introduce WHILE loops. For this loop to run one condition must be met.
3. While in Idle under File select “Open” and after Navigating to SensorEducation open Loop.py
4. Remove the “”” surrounding the first loop and run your program, what happens?

```
"""
while(True):
    print("*")
"""
```

5. Because the condition inside the parentheses never changes the program will run forever.

6. Put the “”” back and do the same with the second loop, what is the result of running your program?

```
"""
while(False):
    print("*")
"""
```

7. In order for a while loop to run, the boolean value inside the parentheses must be True, and rather than just writing True or False there are many ways to change what’s inside.
8. Before you run the following while loop what do you think this bit of code does? After you put some thought in run your code to see what happens, how many times did the star print?

```
#Assign counter an integer digit
counter = 0
#Assign runCount an integer digit greater than runCount
runCount = 10
while(counter < runCount):
    print("*")
    counter +=1
```

9. What we did here was create a counter, and then a limit for the counter to go up to. By adding 1 to the variable every time the loop ran we were able to affect the condition inside the loop.
10. While counter is less the runCount(true) the loop will run, while counter is greater than (this making counter < runCount false) the loop will not run.
11. Try making your own while loop with your own variables, how will you change the condition to make your code run?

Functions

<u>Experiment 3</u>

1. Functions allow you to write bits of code that will only be used when you call the function, if you don’t call the function the code won’t be ran!.
2. Think of a programming function just like a math function, there’s an input, called parameters, and an output which is given in a return statement.
3. While in Idle under File select “Open” and after Navigating to SensorEducation open Functions.py
4. The below bit of code is a basic function with no parameters or return statements. It is defined, has a name, and has a body.

```
def printHelloWorld():
    print("Hello World")

#Call the function simply by typing the name
printHelloWorld()
```

5. The function can be called simply by typing what had came after “def”.
6. The following function has both a set of parameters and a return statement (meaning we could set a variable equal to the result of this function).

```
def sumNumbers(x, y):
    result = x + y
    return result
```

7. Look at the execution of sumOffVariables then run your code.
8. A function can have any number of parameters and they do not need to all be the same data type.
9. The following function has 3 variables, 2 different data types, and runs a while loop! Run your program to see how this function works.

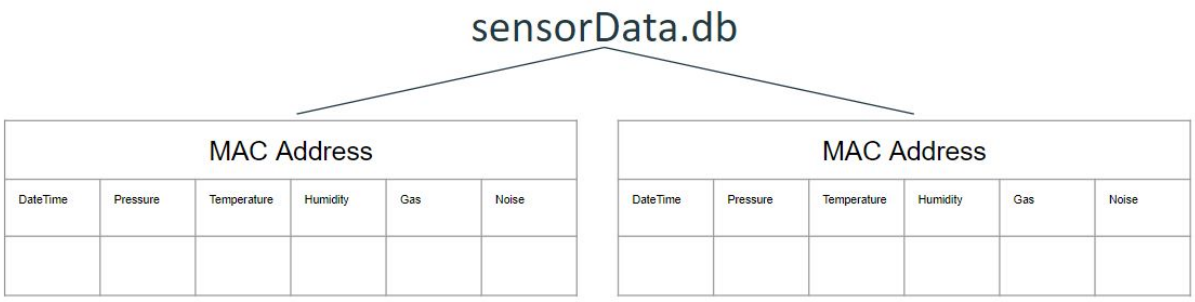
```
def printMultipleTimes(counter, countUpTo, word):
    while(counter < countUpTo):
        print(word)
        counter += 1

printMultipleTimes(0,10,"Hello")
```

10. The possibilities of functions are endless! Try making your own function below then run your program to see your work in action!

Database

There are many different ways to structure a database. In this section we will quickly go over an SQL architecture as that is how we will store the data our sensor reads. An SQL database is essentially a bunch of different table in one file. The following is a diagram of how our database is set up:



Here are some terms you will see when dealing with an SQL database:

- PRIMARY KEY - The primary variable you will use to get data from a table.
- TEXT - This is a variable type for SQL, comparable to pythons string.
- FLOAT - This is how double/float numbers are identified in SQL.
- INSERT INTO - Add data to a specified table and specified column.
- FROM - From a specific table.
- SELECT - Choose what variable you want from a table.
- WHERE - From the data you selected only show data where your condition is met.

SQL is a powerful format for databases and can get way more in depth than the knowledge shared above. This basic introduction will be enough for you to understand how the sensor data is saved.

Circuit Lessons

Experiment 4

Have you ever had to jump start your car or see a movie where a character has? In either of these situations you've probably seen a black and red wire. When it comes to circuits these two colors are the bread and butter of it all. Let's find out what each of them mean!

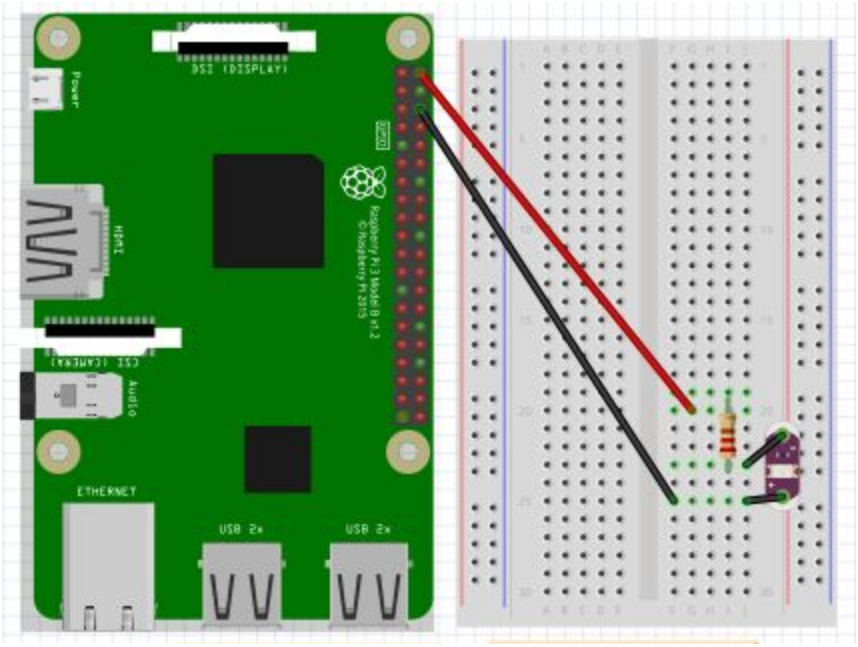
The Red Wire - In circuits, red is the color of power. In order to get the power from the voltage source to your apparatus you must connect the two items with a red wire.

The Black Wire - A black wire allows the power to get grounded, without it you can fry a sensor and render it useless. It's important to connect the black wire to the opposite end the red wire is connect to allowing the current to exit your apparatus.

Lets get started with making a light shine!

Setting Up The Hardware

1) Disconnect the pi from its power source. When working with circuits it's important to not be connected to power to protect your board and yourself.
2) Take a look at your LED light, notice anything different about the two ends?
   One end is longer than the other! This is where the positive voltage enters and thus should be connected to the red wire. The shorter side is the negative lead and thus where the voltage leaves meaning it should be connected to the black wire.
3) Place your two LED leads in the same alphabetical column.
4) Next we need to add a resistor to help control the flow of the current through the LED to prevent burning it. Connect one end of the resistor in the column next to the positive lead of your LED, and the other end in the same column but a different row.
5) Connect your red wire to the part of your resistor not parallel to the LED and connect your black wire to the negative end of your LED.
6) Now we will connect the breadboard to your Raspberry Pi. You can look at the image below to figure out where to connect both the black and red wire.



Making The Program

1. While in Idle under File select "Open" and after Navigating to SensorEducation open Circuits.py
2. GPIO is used to set up the pins on your Raspberry Pi to get them ready for data collection.
3. Time is used to control the board by giving specific periods of waiting time.

4. The following bit prepares the board for collecting data using the import mentioned above.

```
#How many times do you want the LED light to blink?
repeat = 60

#The following sets up the entire board.
GPIO.setmode(GPIO.BOARD)

#Sets up pin 16 on the board for reaiding data.
GPIO.setup(16, GPIO.OUT)
```

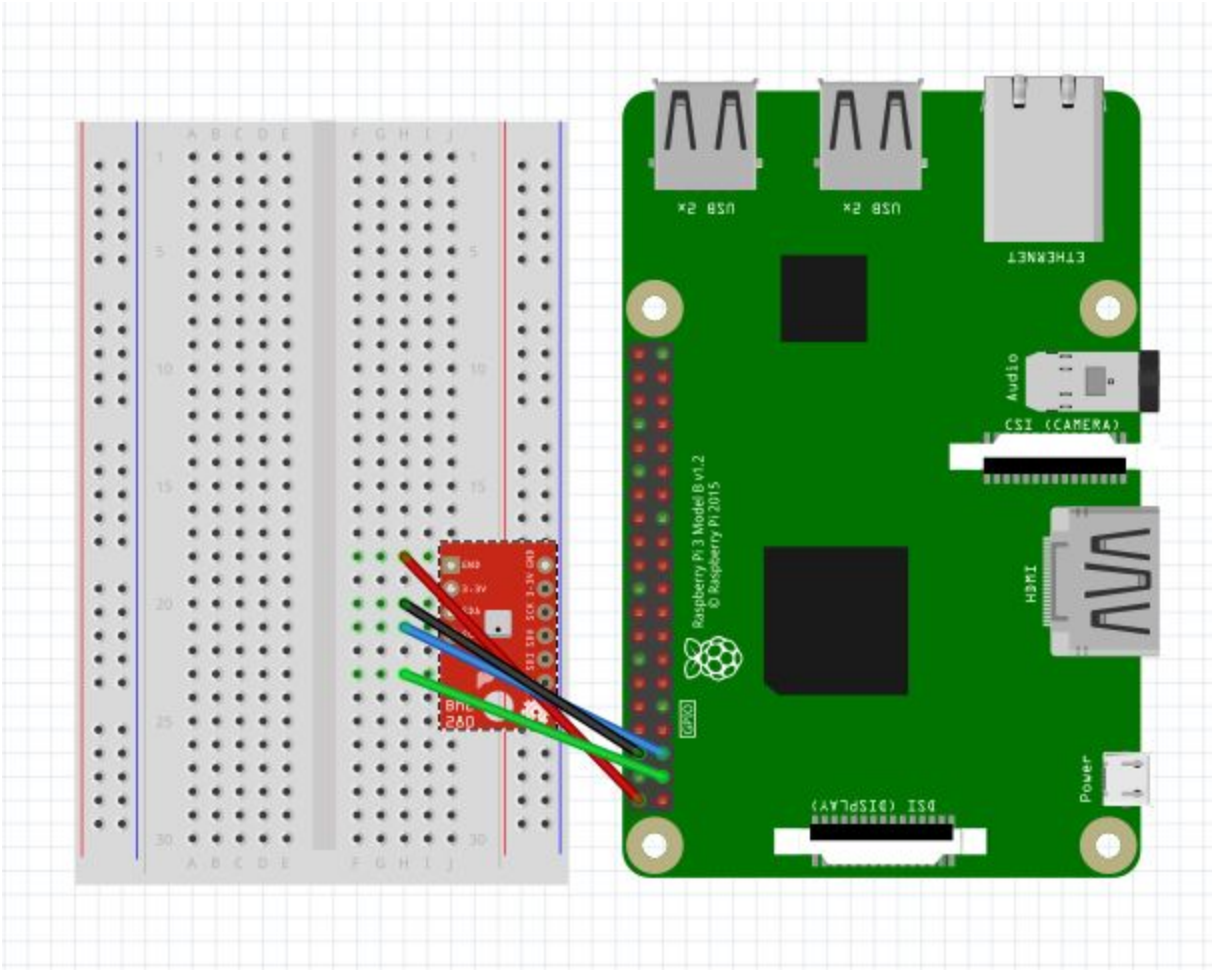
5. Make your own while loop to make the LED light blink, three functions you must use include:
    a. GPIO.output(PIN_NUMBER, GPIO.HIGH)
    b. GPIO.output(PIN_NUMBER, GPIO.LOW)
    c. time.sleep(WAIT_PERIOD)
6. Once you got your while loop set up run your program to see if the light blinks.

Sensor Code

Sensors can perform a variety of different tasks and each individual sensor can perform differently depending on how you code it. This workshop works with Adafruits BME680 sensor to collect data on pressure, temperature, humidity, and gas resistance.

Setting Up The Hardware

Using your knowledge of wiring from the LED experiment, try to set up your board given the diagram below.



Setting Up The Software

1. While in Idle under File select “Open” and after Navigating to SensorEducation open SensorProgram.py
2. Let's look at the imports first. You’ve seen most before but the bme680 library is the library for our sensor and datetime allows you to set up code to get, believe it or not, the date and time.

```
#Imports
import bme680
import time
import datetime
import sqlite3
import os
from sqlite3 import Error
```

3. A good practice when working in a program that does a lot is having a main function that you can execute at the end.

4. Lets assign the variables at the top that will help us control our while loop later on.

```
#Number of times you want while loop to repeat
repeat =
#Seconds you want to wait between each reading
wait_period =
#Keep 0, incriment each time you take a reading
count =

#Create Sensor Variable
sensor = bme680.BME680(i2c_addr=0x77)
```

5. In order to manipulate our sensor we must first assign it to a variable like so.
6. As we mentioned early this sensor will store its data in an SQL database. To do so we must declare a variable for the database and create a cursor to access its data.

```
#Database
#Creates/open a file called sensorData.db by defining the path
db = sqlite3.connect("/home/pi/Pimoroni/bme680/examples/testDB.db")
cursor = db.cursor()

#In order to get the MAC Adress we need to run another program
try:
    cursor.execute('''
        CREATE TABLE <MAC_ADDRESS>(dateTime TEXT PRIMARY KEY,  temperature FLOAT,
                        pressure FLOAT, humidity FLOAT, gas FLOAST)
    ''')
except Error as e:
    print("Error: " + str(e))
```

7. For this sensor, we set up a while loop to gather the data. Within the while loop is an “if else statement”. If the condition is true it will perform the block of code after it, if it is false it will move to the else or do nothing at all.
8. We want to know when a reading takes place, so we find the date and find the time and combine them to one variable.
9. Before we go over getting the readings float("{0:.2f}".format() allows us to round a number to x amount of decimal places, where x is the number before f.

10. To get a certain reading from the board we can use sensor.data.TERM, where term is whatever data we are looking for (assuming the sensor can record that information).
11. The following code allows us to save all the data recorded, try converting the temperature data from celsius to fahrenheit.

```
temperatureCelcius = float("{0:.2f}".format(sensor.data.temperature))
#Convert the above variable to fahrenheit and assign the result to temperature
temperature =
pressure = float("{0:.2f}".format(sensor.data.pressure))
humidity = float("{0:.2f}".format(sensor.data.humidity))
gas = float("{0:.2f}".format(sensor.data.gas_resistance))
```

13. Print the data in whatever manner you like so we can see what the sensor is recording.
14. Don't forget to increase the incrementer so our loop eventually ends.
15. Run your code to see if everything has been set up properly.

Reading

Experiment 7

1. Gathering data can be useless if your not able to look at it again, this experiment will take you through printing
2. While in Idle under File select "Open" and after Navigating to SensorEducation open readData.py.
3. At the beginning of this workshop we learned about while loops, this program uses a for loop to go over all the data in a database.
4. To format a print statement you can put a {#} with # being the order it comes after the initial print statement, for example,
    a. print("Hello {0}, my name is {1}".format("Student", "Jacob"))
5. If row[0] displays the date/time what do you think displays the other information? Write a print statement that will display the data within the database
6. Run your program when you are ready.

Conclusion

By the conclusion of this lab you should have a basic understanding of circuits and breadboards. In addition with completion of this workshop students should have learned introduction to Python ranging from variables, all the way to functions. The Raspberry Pi board is now equipped to gather environmental data that the city or yourself can inspect.