Darkness Sensor (2.0! Now with more LEDs)

Final Report

Aashima Mehta, Anh Phan and Taylor Matthew

Due: December 14, 2018

Summary:

Our project is about sensing the darkness in a room and turning a number of LEDs on according to it. Our intent is that if the room is bright then no LEDs will light up. If the room is a little dark then two LEDs will turn on, and if the room is very dark then all four LEDs will turn on. Our project is basically a light sensor and will light up the circuit, depending upon the readings from the photoresistor.

From this project we learned how to combine several previously known circuits together, and had to work to utilize space on the board appropriately. We also integrated loops, assigned variables and boolean statements, as well as the `return` function, to create a working circuit.

Table of Contents:

Introduction & Theory . . . . Page 3

Design & Experimental Setup . . . . Page 4 - 7

Code & Comments . . . . Page 7 - 11

Scoping . . . . Page 12

Resources . . . . Page 13

Peer Reviews . . . . Page 14

Introduction:

We all have seen light sensors in our everyday life, and this project is also about creating a light sensor at a circuit python programming level. By using the microcontrollers we can upload any python based code directing the circuit to do things for us..This project is an attempt to create a light sensor that can detect the brightness level in the room and can turn a specified number of LEDs on depending upon how dark the room really is. We can setup the threshold level by setting the values that would determine which LED to light and how many.

Theory:

The key components of our project are:

- The Photoresistor
- LEDs
- Usage of a `while` loop
- Variable assignment
- The `return` function

The photoresistor is used to receive input that dictates the number of LEDs lit on the board. This is essential to our project as it is a “darkness sensor”. The LEDs are important as the indicate the readings received from the photoresistor. We use the `while` loop to run our program continuously, to keep reading the photoresistors input and adjust accordingly. Variable assignment is especially important in our code as it allows us to set the thresholds for when each LED needs to be lit, and also assigns the different ports and pins on our board and gadgets for integration in the program. Finally, the `return` function is useful because it ‘saves’ the value given from the photoresistor at a given time and allows us to call on it in our `while` loop, to keep our code continuous and constant.

Design & Experimental Setup:

Circuit Python Microcontroller Wiring:

Wire your breadboard & M0 Express just like the diagram below:

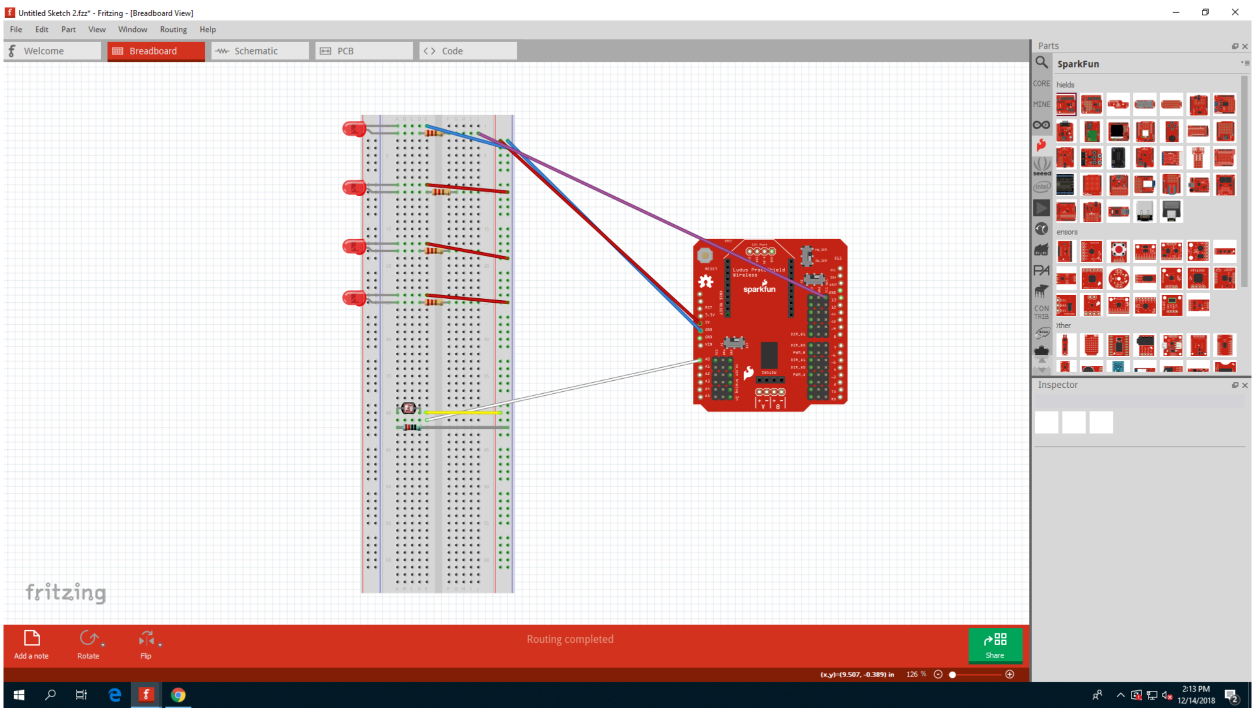

Figure 1: Fritz Diagram
Be really careful when it comes to using the right ports, it often helps to read the code as you wire, paying extreme attention to details helps minimize frustration and confusion later.
Note: the board shown in the picture above is "Redboard" whereas we will be using Metro M0 board for our experiment. Both of these boards are physically very similar to one another so the wiring is going to be exactly like as shown above.

Parts Needed:

- Adafruit Metro M0 Express Board
- Breadboard
- Photoresistor
    - *The readings produced from this will determine how many LEDs will be lit at a time.*
    - *In our code we can indicate the threshold for the sensor, which in turn will directed towards how many LEDs should be turned on at that time.*

  - *In our code, we suggested the time.sleep to be for two seconds. Meaning that our sensor will check for how bright the room is after every two seconds.*
- 13 cables
- 4 330Ω Resistors
  - *The resistors will help proving the resistance for our LED's, so they won't burst off because of the direct flow of the current towards them.*
  - *Resistor helps determine and control the flow of the current for each Led.*
  - *We will be using four LED's in our experiment therefore we will need four resistors.*

- 1 10K Ω Resistor
- 4 LEDs of various colors

Hookup Table:

Component	***M0 Metro Board***	***Breadboard***	***Breadboard***
Jumper Wire	***5V***	***5V Rail (+)***	
Jumper Wire	***GND***	***GND Rail (-)***	
LED		***A1 LED (-)***	***A2 LED (+)***
330Ω Resistor ***(orange, orange, brown)***		***E2***	***F2***
Jumper Wire		***E1***	***GND Rail (-)***

Jumper Wire	Digital Pin 13	J2	
Photoresistor		A26	B25
10kΩ Resistor (brown, black, orange)		C26	D27
Jumper Wire	Analog Pin 0 (A0)	E26	
Jumper Wire		E25	5V Rail (+)
Jumper Wire		E27	GND Rail (-)
330Ω Resistor (orange, orange, brown)		E4	F4
330Ω Resistor (orange, orange, brown)		E6	F6
330Ω Resistor (orange, orange, brown)		E8	F8
Jumper Wire		E3	GND Rail (-)
Jumper Wire		E5	GND Rail (-)

Jumper Wire		***E7***	***GND Rail (-)***
Jumper Wire	***Digital Pin 12***	***J4***	
Jumper Wire	***Digital Pin 11***	***J6***	
Jumper Wire	***Digital Pin 10-***	***J8***	
LED		***A3 LED (-)***	***A4 LED (+)***
LED		***A5LED (-)***	***A6 LED (+)***
LED		***A7 LED (-)***	***A8 LED (+)***

Code & Comments:

Python Usage:

Below we're going to go through the code utilized step by step, for clarity.

First, we'll need to initialize the program. To do this, **`import board`** at the top of your code, and to define all of our connecting ports and pins, **`import digitalio`** . Our other required libraries are **`analogio`** and **`time`** , which will allow us to use a while loop effectively, so we need to import these as well.

```
import board
import digitalio
import analogio
import time
```

Next, we'll set the values for which each step of the program will be activated.

```
threshold_1 = 1
threshold_2 = 2.3
threshold_3 = 3
```

These variable values are based on the reading received from the photoreceptor.

*For our pins and ports***,** we need to assign more variables, that connect the board to our code, so that we can call upon them as needed.

```
led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT
##This connects an LED to the board's pin D13##

led_1 = digitalio.DigitalInOut(board.D12)
led_1.direction = digitalio.Direction.OUTPUT

led_2 = digitalio.DigitalInOut(board.D11)
led_2.direction = digitalio.Direction.OUTPUT

led_3 = digitalio.DigitalInOut(board.D10)
led_3.direction = digitalio.Direction.OUTPUT

volts = analog_voltage(photoresistor)
photoresistor = analogio.AnalogIn(board.A0)
```

To define our function, where the real action happens, we need to use the **`def`** function, and give our function a name like **`analog_voltage`**. This takes in the values from the **`photoresistor`**, as a parameter. Then, we'll return the value from the `photoresistor`, but edited so that the program can read it with ease, as we did in the first photoresistor project.

We used `65535` value because it represents the maximum value of the voltage. We are dividing it with the `photoresistor.value` and multiplying with the `reference voltage` because it will give us the value of the resistance of the current flowing through our circuit. Also the reference volatage for our board was 3.3 but it was better to write it as reference voltage as it just makes it more general, our boards can only understands the resistance values.

```
def analog_voltage(photoresistor):
 return (photoresistor.value / 65535 *
photoresistor.reference_voltage)
```

To keep our program continuous, we use a **`while`** loop for our different threshold values and our number of LEDs used.

```
while True:
```

Using **`if`** statements, we will set the different levels of darkness, with the different threshold variable titles:

```
 if  (volts/6) > threshold_3:
  (led.value = True)
```

Using an **`elif`** statement is best here, because it does not ‘stop’ the value received, but moves it along through the other conditionals until a condition is met.

```
elif (volts/3)> threshold_2:
  (led.value = True) and (led_2.value = True)
elif (volts/2) > threshold_1:
  led.value = 1
  led.value_1 = 1
  led.value_2 = 1
  led.value_3 = 1
```

Finally, an else statement is used as the last resort, a default for what happens when none of the other conditions are met. In this case, the else statement is for when the room is bright, and so none of the LEDs are turned on.

```
else:
  led.value = False
  led.value_1 = False
  led.value_2 = False
  led.value_3 = False
```

And as a final note, a `print` statement is used to show the coder what values are being received. Also, the `sleep` function is used here to reinitialize the code every 2 seconds.

```
print(volts)
time.sleep(2)
```

Just the Code:

```
import board
import digitalio
import analogio
import time
#we establish different-different values for our threshold. These
will detect how bright the room is and will determine voltage our
circuit has to light up the LEDs.

threshold_1 = 1
threshold_2 = 2.3
threshld_3 = 3

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

led_1 = digitalio.DigitalInOut(board.D12)
led_1.direction = digitalio.Direction.OUTPUT

led_2 = digitalio.DigitalInOut(board.D11)
```

```
led_2.direction = digitalio.Direction.OUTPUT

led_3 = digitalio.DigitalInOut(board.D10)
led_3.direction = digitalio.Direction.OUTPUT

volts = analog_voltage(photoresistor)
photoresistor = analogio.AnalogIn(board.A0)

def analog_voltage(photoresistor):
 return (photoresistor.value / 65535 *
photoresistor.reference_voltage)


while True:
 if  (volts/6) > threshold_3:
   (led.value = True)
 elif (volts/3)> threshold_2:
   (led.value = True) and (led_2.value = True)
 elif (volts/2) > threshold_1:
   led.value = 1
   led.value_1 = 1
   led.value_2 = 1
   led.value_3 = 1
 else:
   led.value = False
   led.value_1 = False
   led.value_2 = False
   led.value_3 = False

print(volts)
time.sleep(2)
```

Scoping:

For scoping, we chose to add features from two other chapters to a two star problem from the back of any chapter.

We chose problem 4.4 from the end of Chapter 4.

(Game: learn addition) Write a program that generates two integers under 100 and prompts the user to enter the sum of these two integers. The program then reports true if the answer is correct, false otherwise. The program is similar to Listing 4.1.

To complicate this problem even further we chose to add a loop into the problem, and integrate a list of responses for correct answers, that we can pull from randomly.

```
import random
comments = ["Nice job, you did it!", "Great!", "Woohoo! Next!",
"You're super smart.", “Wow, whiz kid!”, “Gee, are you Einstein or
something?”]

def learn_addition():
    print("Welcome to the learning addition game! /n Get ready!")

    a = random.randint(1,101)
    b = random.randint(1,101)
    answer = input("What is " + str(a) + " + " + str(b) + " ?")

    if int(answer) == (int(a) + int(b)):
            print(random.choice(comments))
    else:
            print("I'm sorry, that's not correct!")

learn_addition()
```

Bibliography:

Resources:

- ***JOEL_E_B (n.d.). SparkFun Inventor's Kit Experiment Guide - v4.0: Circuit 1C: Photoresistor. Retrieved from https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40/circuit-1c-photoresistor***
- ***http://fritzing.org/learning/tutorials***
- ***JOEL_E_B (n.d.). SparkFun Inventor's Kit Experiment Guide - v4.0: Project 1: Light. Retrieved from https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40/project-1-light***
- ***Liang, Y. D. (2012). Introduction to programming using Python***

Peer Reviews:

Taylor	I think Aashima and I worked equally on the project, on top of editing each other's work we worked really well together. I got stuck on the sensor part as the design wasn't correct. But Aashima and Anh figured it out. We both worked on the poster in both format and content.

Aashima	I also think that we both worked equally hard on this project. I personally think that working with a partner was a better idea for final or in general for all projects, as most of the times if someone would have been stuck at a point then the other would have some solution and visa versa.

Anh	Taylor & Aashima both worked on the project the same amount. They collaborated on many parts of the project too to work together. Both of them also helped me figure out the code for my motion project.