Alexander Akomer (Partner: Shaheen Ebrahimi)

CSCE 462 – 500 Liu

3/4/2022

Lab 4 Report

1) **How frequently can the sensor be polled? Is this fast enough for this lab? Why or why not?** The MPU6050 allows a sample rate of 8kHz for the gyrometer, though only 1kHz for the accelerometer readings. However, this is more than fast enough for this lab. In this lab, calculations are made to determine if a step is detected or not, and humans only step at highest about 2-3 times per second. Therefore, 1kHz provides well over the required margin to accurately determine changes in acceleration related to steps.

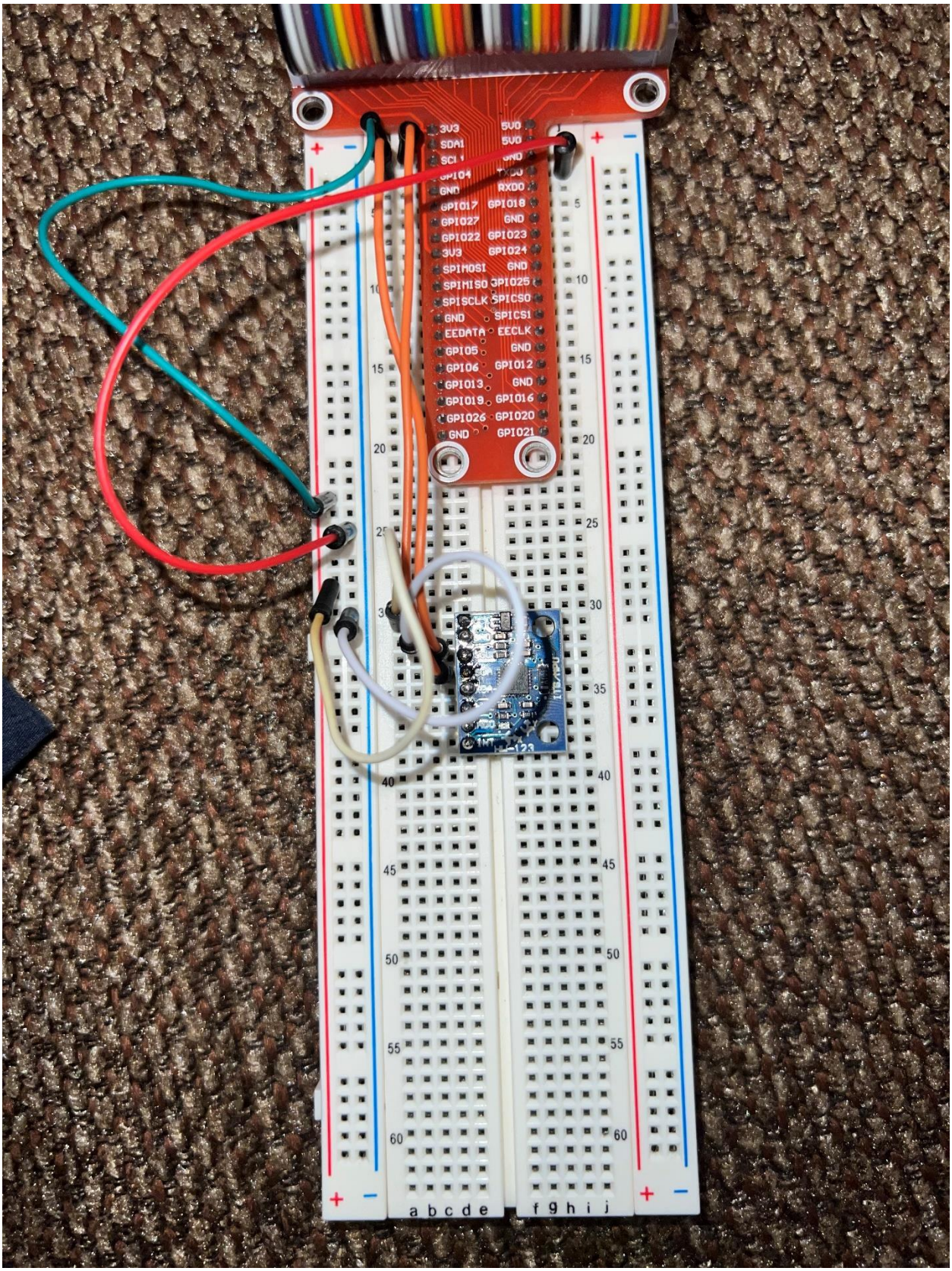
2) **Did you apply filter(s) to sensor readings? If yes, how did you use the filter(s)? If no, why?** Originally, we thought we would have to apply a filter to the sensor readings. However, upon initial testing, a plot of the acceleration magnitude over time showed very clear peaks indicating each step. There was noise that could have been reduced but, maybe due to the fact that I am quite tall and therefore take longer strides, noise became negligible. Essentially any magnitude greater than 130 (which is an arbitrary number determined from several tests), was well above the point of being noise and well below the peaks of the magnitude generated by my steps. Therefore, we learned that no filters were required to accurately deduce steps as I took them.

3) **Explain your step counting algorithm and its accuracy.** The counting algorithm was simply, utilizing an infinite loop, determine if magnitude (Pythagorean theorem of all three vectors of the accelerometer: x, y, and z) reaches a peak of 130 or greater. If so, increment the number of steps and print it; if not, continue as normal. This algorithm worked perfectly without the need of overcomplicating with filters or normalization – again, maybe due to the fact that I am tall and therefore my peak accelerations were well above the noise. Each iteration of this loop implemented a sleep based on the sampling rate of 0.1 seconds to avoid the problem of overcounting because there are many data points within each peak. Using this simple system, we showed 100% accuracy during the lab demo and about 95% accuracy on average (which lowered to 90% as less linear paths were taken or the step counter was shaken too much).

Code:

```
import board
import busio
import adafruit_mpu6050
import plotly.express as px

# perf_counter is more precise than time() for dt calculation
from time import sleep, perf_counter

i2c = busio.I2C(board.SCL, board.SDA)
mpu = adafruit_mpu6050.MPU6050(i2c)

t_vals = [] # time variables used solely for plotting
mag_vals = []
sampling_rate = 0.1
tolerance = 130

def getSteps():
    t = 0
    steps = 0
    while True:
        (x,y,z) = mpu.acceleration
        magnitude = x**2 + y**2 + z**2 # calculation of magnitude
        mag_vals.append(magnitude)
        if (magnitude > tolerance):
            steps += 1
            print(steps)
            sleep(sampling_rate)
        t_vals.append(t)
        t += sampling_rate
        sleep(sampling_rate)
    return steps

def plot(t_vals, y_vals): # a secondary function used to plot the magnitudes over time
    fig = px.line(x=t_vals, y=y_vals, title='Acceleration Plot')
    fig.show()

print("Total Calculated Steps: ", getSteps())
plot(t_vals, mag_vals) # plot function only runs when not running infinite loop
```

Schematic:


3V3
SDA1
GND
GPIO17
GPIO27
GPIO22
3V3
SPIMOSI
SPIMISO
SPISCLK
GND
EEDATA
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
5V0
5V0
GND
RXD0
GPIO18
GND
GPIO23
GPIO24
GND
GPIO25
SPICS0
SPICS1
EECLK
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21
a b c d e
f g h i j