stats\_tgyr = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]  
stats\_agyr = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]  
n = 0  
**while** n < 200:  
 raw = ble.read\_gyro()  
 tgyr = raw[0]  
 agyr = raw[1]  
 **for** i **in** range(3):  
 **if** tgyr[i] < stats\_tgyr[0][i]:  
 stats\_tgyr[0][i] = tgyr[i]  
 **elif** tgyr[i] > stats\_tgyr[1][i]:  
 stats\_tgyr[1][i] = tgyr[i]  
 stats\_tgyr[2][i] += tgyr[i]  
 **if** agyr[i] < stats\_agyr[0][i]:  
 stats\_agyr[0][i] = agyr[i]  
 **elif** agyr[i] > stats\_agyr[1][i]:  
 stats\_agyr[1][i] = agyr[i]  
 stats\_agyr[2][i] += agyr[i]  
 time.sleep(0.05)  
 n += 1  
**for** i **in** range(3):  
 stats\_agyr[2][i] = int(stats\_agyr[2][i] / n)  
 stats\_tgyr[2][i] = int(stats\_tgyr[2][i] / n)  
self.val\_prof[mode] = [stats\_tgyr, stats\_agyr]  
print('Finished Calibrating')  
**return**

{'atheta':0,'aphi':0,'ttheta':0,'tphi':0}

self.gravity\_direction['aphi'] = math.atan(average\_acc[0][2]/math.sqrt(average\_acc[0][0]\*\*2+average\_acc[0][1]\*\*2))  
self.gravity\_direction['tphi'] = math.atan(average\_acc[1][2] / math.sqrt(average\_acc[1][0] \*\* 2 + average\_acc[1][1] \*\* 2))  
self.gravity\_direction['atheta'] = math.atan

(average\_acc[0][1] / average\_acc[0][2])  
self.gravity\_direction['ttheta'] = math.atan(average\_acc[1][1] / average\_acc[1][2])

start\_time = time.time()  
**while**(time.time()-start\_time<2000):  
 int\_time = time.time()  
 ble.empty\_buffer()  
 acc\_data = ble.read\_acc()  
 gyro\_data = ble.read\_gyro()  
 net\_acc = [0]\*6  
 **for** i **in** range(6):  
 net\_acc[i] = acc\_data[i]-self.gravity[i]  
 self.position[i] += (net\_acc[i]\* (time.time()-int\_time)\*\*2)

// read raw accel/gyro measurements from device

accelgyro.getMotion6(&a1x, &a1y, &a1z, &g1x, &g1y, &g1z);

accelgyro2.getMotion6(&a2x, &a2y, &a2z, &g2x, &g2y, &g2z);

// these methods (and a few others) are also available

//accelgyro.getAcceleration(&ax, &ay, &az);

//accelgyro.getRotation(&gx, &gy, &gz);

#ifdef OUTPUT\_READABLE\_ACCELGYRO

// display tab-separated accel/gyro x/y/z values

Serial.print("FIRST a/g:\t");

Serial.print(a1x); Serial.print("\t");

Serial.print(a1y); Serial.print("\t");

Serial.print(a1z); Serial.print("\t");

Serial.print(g1x); Serial.print("\t");

Serial.print(g1y); Serial.print("\t");

Serial.println(g1z);

Serial.print("SECOND a/g:\t");

Serial.print(a2x); Serial.print("\t");

Serial.print(a2y); Serial.print("\t");

Serial.print(a2z); Serial.print("\t");

Serial.print(g2x); Serial.print("\t");

Serial.print(g2y); Serial.print("\t");

Serial.println(g2z);

#endif

#ifdef OUTPUT\_BINARY\_ACCELGYRO

Serial.write((uint8\_t)(ax >> 8)); Serial.write((uint8\_t)(ax & 0xFF));

Serial.write((uint8\_t)(ay >> 8)); Serial.write((uint8\_t)(ay & 0xFF));

Serial.write((uint8\_t)(az >> 8)); Serial.write((uint8\_t)(az & 0xFF));

Serial.write((uint8\_t)(gx >> 8)); Serial.write((uint8\_t)(gx & 0xFF));

Serial.write((uint8\_t)(gy >> 8)); Serial.write((uint8\_t)(gy & 0xFF));

Serial.write((uint8\_t)(gz >> 8)); Serial.write((uint8\_t)(gz & 0xFF));

#endif

delay(1000);

// blink LED to indicate activity

blinkState = !blinkState;

digitalWrite(LED\_PIN, blinkState);

**def read\_data**(self):   
 s = ble.read\_received() # reads the shoe's response  
 **while not** ('start' **in** s **and** 'end' **in** s):  
 s.extend(ble.read\_received())  
 #print(s)  
 **if not** no\_input:  
 **return  
 if** "end" **in** s **and** "start" **not in** s:  
 s = []  
 gyro\_data = [0, 0, 0, 0, 0, 0]  
 accel\_data = [0, 0, 0, 0, 0, 0]  
 acc = **False** gyro = **True** # change this when adding acc data  
 start = s.index('start')  
 ind = start + 1  
 i = 0  
 **while not** s[ind] == 'end':  
 **if** s[ind] == "G":  
 acc = **False** gyro = **True** i = 0  
 **elif** s[ind] == "A":  
 acc = **True** gyro = **False** i = 0  
 **elif** "." **in** s[ind]:  
 **if** gyro:  
 gyro\_data[i] = float(s[ind])  
 **elif** acc:  
 accel\_data[i] = float(s[ind])  
 i += 1  
 ind += 1  
 print(gyro\_data)  
 self.write\_to\_file(gyro\_data)  
 **return** gyro\_data, accel\_data

'''file = open("time\_data.txt", "w")  
ble.dummy\_read(file, time.time())  
file.close()  
print("out")  
while True:  
 b = ble.read\_buffer()  
 if not b is "":  
 print(b)'''  
'''print("reading")  
read = ble.read\_received()  
start = time.time()  
while not 'START' in read: # glitchy - requires a well timed reset  
 ble.send\_message('SENDALL')  
 time.sleep(0.1)  
 if time.time() - start > 30:  
 print("took too long!!")  
 quit()  
 read = ble.read\_received() # somehow get this data into read data at the end of this method...  
print("MADE IT")  
outfile = open('raw\_data.txt','w')  
ble.dummy\_read(outfile,time.time())  
outfile.close()'''