MPU6050

IMU Controlling Program

Introduction

- This lesson will demonstrate how to use the IMU to control a graphic cursor in a program. The graphic cursor will be a ball.
- The angle of the MPU6050 will be used to control the cursor. To move the cursor in the X direction, the Z-axis rotation of the sensor will be used. For the Y direction, the Y-axis rotation will be used. To simulate a mouse click, the rotation around the X-axis will be used.
- Control will be made wireless using WiFi.

Program Operation

- The code on the SLATE will sample the MPU6050 sensor and send the angle data to the python program continuously. It is a simple loop.
- The python program will start up, open a 900x900 pixel window and start receiving the MPU6050 angle data. The data will seem to be starting at some random value. A calibration needs to be done. The user will point their hand with the sensor toward the screen. The user will then press a key and the program will take the sensor and use it as the center position of the window. After that, the program will interpret the sensor data and calculate the cursor position.

Arduino Program

- The program from the MPU6050 program will be used again.
- Shown to the right is the initialization part of the Arduino program.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <MPU6050_tockn.h>
#include <Wire.h>

WiFiUDP client;
MPU6050 mpu6050(Wire);

long timer = 0;
char buf[64];

void setup() {
  Serial.begin(115200);
  Wire.begin(4,5);
  mpu6050.begin(ACCEL_2G,GYRO_500);
  mpu6050.calcGyroOffsets(true);
  WiFi.mode(WIFI_AP);
  WiFi.softAP("UDPIMU");
}
```

Arduino Program

- The loop() function also does not change.
- The X,Y and Z axis angles are sent to the python program.

```
void loop() {
  if(millis() - timer > 100) {
    mpu6050.update();
   timer = millis();
    snprintf(buf,64,"%f,%f,%f\n",mpu6050.getAngleX(),mpu6050.getAngleY(),mpu6050.getAngleZ())
    Serial.print(buf);
    client.beginPacket("192.168.4.2",3000); destination IP and port
    client.write(buf);
    client.endPacket();
  }
}
```

Python Program

- The highlighted code is used to find the computers WiFi IP address. This is needed so the program knows where to listen for UDP packets. A computer will have at least one IP address and it is always 127.0.0.1. This is used for internal network communications. The highlighted code tries to contact the IP address 8.8.8.8. This gets the computer to use the WiFi and the WiFi IP address becomes the active network.

```
import socket
import pygame

s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.connect(('8.8.8.8',80))
localip = s.getsockname()[0]
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind((localip,3000))

px = 0
py = 0
first = 1
ix = 0
iy = 0
iz = 0
x = 0
y = 0
z = 0
range = 80 #motion range of 80 degrees
```

Python Program

- The next two lines restart the network connection. **s.bind()** is how the program is set up to listen to the computers network port for incoming UDP packets.
- The function **socket.socket()** needs to be called again so **s.bind()** will work.

```
import socket
import pygame

s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.connect(('8.8.8.8',80))
localip = s.getsockname()[0]
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind((localip,3000))

px = 0
py = 0
first = 1
ix = 0
iy = 0
iz = 0
x = 0
y = 0
z = 0
range = 80 #motion range of 80 degrees
```

Python Program

- Variables px and py are created and set to zero. These variables will be the cursor position in the window.
- Variables ix,iy,iz will be the calibration variables. The variables will be set when a key is pressed.
- Variables x,y,z will be used to hold the angle data. They need to be declared and set here so they can be used in all parts of the code.
- range specifies the sensor angle to the screen width.
- All these variables are global.

```
import socket
import pygame

s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.connect(('8.8.8.8',80))
localip = s.getsockname()[0]
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind((localip,3000))

px = 0
py = 0
first = 1
ix = 0
iy = 0
iz = 0
x = 0
y = 0
z = 0
range = 80 #motion range of 80 degrees
```

Python Program

- This part of the code initializes and opens the window.
- In the while loop, all events are processed. If the close button on the window is clicked, the program will quit. If a key is pressed on the keyboard, the calibration variables will be loaded with the current sensor data.

```
pygame.init()
scn = pygame.display.set_mode((900,900))

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        if event.type == pygame.KEYDOWN:
          ix = x
          iy = y
          iz = z
```

Python Program

- **s.recvfrom()** waits and receives the UDP packet. Variable a will hold the packet content and variable b will contain the IP address of who sent the UDP packet.
- The data packet received is in a byte text format. **c.decode()** converts the byte text to a String type value which is unicode.

```
a,b = s.recvfrom(256)
c = a.decode('utf-8')
d = c.split(',')
if len(d) == 3:
    x = float(d[0])
    y = float(d[1])
    z = float(d[2])
    if first == 1:
        ix = x
        iy = y
        iz = z
        first =0
    print(x,y,z)
    px = 900/range * (iz - z) + 450
    py = 900/range * (iy - y) + 450

scn.fill((0,0,0))
pygame.draw.circle(scn,(200,0,0),(int(px),int(py)),20)
pygame.display.update()
```

Python Program

- Since the data is sent as text with a coma separating each value, the **c.split()** function separates the three values in the string and stores them in variable **d** which becomes a list type variable.
- The values are converted from String to floating point values and printed to the python IDLE console window.

```
a,b = s.recvfrom(256)
c = a.decode('utf-8')
d = c.split(',')
if len(d) == 3:
    x = float(d[0])
    y = float(d[1])
    z = float(d[2])
    print(x,y,z)
    px = 900/range * (iz - z) + 450
    py = 900/range * (iy - y) + 450

scn.fill((0,0,0))
pygame.draw.circle(scn,(200,0,0),(int(px),int(py)),20)
pygame.display.update()
```

Python Program

- The range of the movement and center position needs to be mapped to the window coordinates. A line slope equation is used.
- The slope is the range of the pixels over the range of the input. The center of the screen added to the equation since that is the intercept point.

```
a,b = s.recvfrom(256)
c = a.decode('utf-8')
d = c.split(',')
if len(d) == 3:
    x = float(d[0])
    y = float(d[1])
    z = float(d[2])
    print(x,y,z)
    px = 900/range * (iz - z) + 450
    py = 900/range * (iy - y) + 450

scn.fill((0,0,0))
pygame.draw.circle(scn,(200,0,0),(int(px),int(py)),20)
pygame.display.update()
```

Python Program

- The last part of the program updates the display. The x-axis movement is flipped using the **900 – px**.
- The circle function requires integers only which is why the **int()** function is included.

```
    a,b = s.recvfrom(256)
    c = a.decode('utf-8')
    d = c.split(',')
    if len(d) == 3:
        x = float(d[0])
        y = float(d[1])
        z = float(d[2])
        print(x,y,z)
        px = 900/range * (iz - z) + 450
        py = 900/range * (iy - y) + 450

    scn.fill((0,0,0))
    pygame.draw.circle(scn,(200,0,0),(int(px),int(py)),20)
    pygame.display.update()
```

Testing

- Load the SLATE code.
- Connect the computer to the SLATE WiFi SSID.
- Run the python program.
- Point the sensor toward the screen and press a key.
- The circle should appear in the center of the screen and as the sensor is tilted, the circle should move around the screen.

Detecting a Click

- Up to this point, the Y and Z axis of the MPU6050 are used for positioning the circle cursor in the python program window.
- The x axis could be used to detect a click. Detecting the rotation of the sensor could be used to detect a click. Rotate counter clockwise for a left click and clockwise for a right click.
- The SLATE program does not change.

Click Detection

- One more variable needs to be added to the python program.
- Variable **rotate** will be used to detect the rotation of the sensor.
- Variable threshold is set to 20 degrees.

```
import socket
import pygame

s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.connect(('8.8.8.8',80))
localip = s.getsockname()[0]
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind((localip,3000))

px = 0
py = 0
first = 1
ix = 0
iy = 0
iz = 0
x = 0
y = 0
z = 0
range = 80 # motion range of 80 degrees
rotate = 0
threshold = 20
```

Click Detection

- In the section of code where the cursor position is calculated, the click detection code is added.
- First the initial position is subtracted from the current sensor value to remove any offset.
- Then the resulting value is compared with the threshold to determine left or right clicks. The left click is a negative angle.

```
a,b = s.recvfrom(256)
c = a.decode('utf-8')
d = c.split(',')
if len(d) == 3:
    x = float(d[0])
    y = float(d[1])
    z = float(d[2])
    print(x,y,z)
    px = 900/range * (iz - z) + 450
    py = 900/range * (iy - y) + 450
    rotate = ix - x
    if rotate < -threshold:
      print("Left click detected")
    if rotate > threshold:
      print("Right click detected")

scn.fill((0,0,0))
pygame.draw.circle(scn,(200,0,0),(int(px),int(py)),20)
pygame.display.update()
```