Task 1:

Output Serial Monitor ×

Not connected. Select a board and a port to connect automatically.


```
}
{
  "ACC_X": 0.362721562,
  "ACC_Y": -2.711433411,
  "ACC_Z": 9.518148422,
  "GYR_X": -0.610351563,
  "GYR_Y": -0.305175781,
  "GYR_Z": -0.122070313
}
{
  "ACC_X": 0.355538964,
```

Task 2:

File Edit Sketch Tools Help

Arduino NANO 33 IoT

nano33_wifiscan.ino

```
// Initialize WiFi module
if (WiFi.status() == WL_NO_MODULE) {
  Serial.println("Communication with WiFi module failed!");
  while (true);
}

// Connect to WiFi network
char ssid[] = "Jacob IPhone";      // Change this to your WiFi network SSID
char password[] = "wifi1234";  // Change this to your WiFi network password

int status = WiFi.begin(ssid, password);
if (status != WL_CONNECTED) {
  Serial.println("Failed to connect to WiFi network!");
  while (true);
}

// Wait for WiFi to be ready
while (WiFi.status() != WL_CONNECTED) {
```

Output Serial Monitor

Message (Enter to send message to 'Arduino NANO 33 IoT' on 'COM4')

```
}
Failed to connect to WiFi network!

Available WiFi Networks:
Network 1: Jacob IPhone
Network 2: UCLA_WIFI
Network 3: UCLA_WEB
Network 4: eduroam
Network 5: ssysarch
Network 6: Linksys02316
Network 7: ncel-wifi3
Network 8: DRL63109
Network 9: GL-A1300-649
Network 10: NETGEAR59
```

Task 3:

Lab4 > nono33_MQTT.py > ...

```python
            "ACC_Z": data["ACC_Z"],
            "GYR_X": data["GYR_X"],
            "GYR_Y": data["GYR_Y"],
            "GYR_Z": data["GYR_Z"]
        }
        print("Data stored:", imu_data)
    except Exception as e:
        print("Error processing message:", e)


client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect_async('mqtt.eclipseprojects.io')
client.loop_forever()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SERIAL MONITOR GITLENS PORTS

```
Data stored: {'ACC_X': 0.397437483, 'ACC_Y': -2.444479942, 'ACC_Z': 9.59835434, 'GYR_X': -0.305175781, 'GYR_Y': -0.366210938, 'GYR_Z': -0.061035156}
Data stored: {'ACC_X': 0.397437483, 'ACC_Y': -2.442085743, 'ACC_Z': 9.593565941, 'GYR_X': -0.427246094, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.405817181, 'ACC_Y': -2.437297344, 'ACC_Z': 9.601945877, 'GYR_X': -0.183105469, 'GYR_Y': -0.366210938, 'GYR_Z': -0.061035156}
Data stored: {'ACC_X': 0.411802679, 'ACC_Y': -2.428917646, 'ACC_Z': 9.610325813, 'GYR_X': -0.366210938, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.41060558, 'ACC_Y': -2.425326347, 'ACC_Z': 9.605537415, 'GYR_X': -0.48828125, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.416591108, 'ACC_Y': -2.432508945, 'ACC_Z': 9.603142738, 'GYR_X': -0.549316406, 'GYR_Y': -0.305175781, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.422576606, 'ACC_Y': -2.431311846, 'ACC_Z': 9.594763756, 'GYR_X': -0.366210938, 'GYR_Y': -0.366210938, 'GYR_Z': -0.061035156}
Data stored: {'ACC_X': 0.422576606, 'ACC_Y': -2.432508945, 'ACC_Z': 9.603142738, 'GYR_X': -0.366210938, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.424970806, 'ACC_Y': -2.433706045, 'ACC_Z': 9.604340553, 'GYR_X': -0.366210938, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.423773706, 'ACC_Y': -2.430114746, 'ACC_Z': 9.597157478, 'GYR_X': -0.427246094, 'GYR_Y': -0.305175781, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.41060558, 'ACC_Y': -2.436100245, 'ACC_Z': 9.605537415, 'GYR_X': -0.305175781, 'GYR_Y': -0.366210938, 'GYR_Z': -0.061035156}
Data stored: {'ACC_X': 0.411802679, 'ACC_Y': -2.438494444, 'ACC_Z': 9.609128952, 'GYR_X': -0.122070313, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.417788208, 'ACC_Y': -2.440888643, 'ACC_Z': 9.605537415, 'GYR_X': -0.061035156, 'GYR_Y': -0.366210938, 'GYR_Z': -0.122070313}
Data stored: {'ACC_X': 0.405817181, 'ACC_Y': -2.7521348, 'ACC_Z': 9.594763756, 'GYR_X': -0.122070313, 'GYR_Y': -0.366210938, 'GYR_Z': -0.061035156}
Data stored: {'ACC_X': 0.403422982, 'ACC_Y': -2.45285964, 'ACC_Z': 9.600749016, 'GYR_X': -0.305175781, 'GYR_Y': -0.366210938, 'GYR_Z': -0.061035156}
```

There is not much lag, but to reduce it we could switch MQTT servers to a closer one or maybe even self-host it nearby.

To work around this lag, we could make it so that we expect it, so when analyzing for timing, we give a small delay buffer knowing that we will receive the data from MQTT a few hundred milliseconds late.

Task 4:

When idle, I get about 9.8 m/s^2 in the direction that is down, which is expected.

I set up a rule that if the m/s^2 in any dimension differs by more than 0.1, it is not idle, else it is idle. This works very well with almost 0% error rate.

```
scan.ino
T_idleornot.py U
T_simplegest... U
T.py U

71
72  client.connect_async('mqtt.eclip
73  client.loop_forever()
74
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SERIA

```
Forward push detected
Upward lift detected
Upward lift detected
Upward lift detected
Upward lift detected
Forward push detected
Forward push detected
Forward push detected
Forward push detected
Forward push detected
Forward push detected
Forward push detected
Traceback (most recent call last):
  File "c:\Users\jacob\OneDrive\2023-2024\2023-20
    client.loop_forever()
```

To track circular motion, I simply said that if the magnitude of the x-y acceleration is above a threshold of 5m/s^2 for over 25 samples, then we must be in circular motion because if we are not walking around, we would have needed to return to our starting point by then.