Informe de Actividad Práctica
Taller de Automatización y Control

Manuel Hirsch Gabriel Vigo Abad

15 de septiembre de 2025

Introducción

El objetivo de esta práctica es la lectura de un sensor analógico y el accionamiento de un servo.

Limitaciones del sensor MPU-6000/6050

- **Retardo (latencia):** El tiempo desde que ocurre un movimiento real hasta que el dato está disponible depende de la tasa de muestreo (ODR) y del filtro digital pasa-bajos (DLPF). El giróscopo presenta un tiempo de arranque de aproximadamente 30 ms, una tasa máxima de muestreo de 8 kHz, y el acelerómetro de 1 kHz. Los filtros pasa-bajos programables (5–256 Hz en giroscopio, 5–260 Hz en acelerómetro) introducen retardo adicional.
- **Mínima unidad discernible:** Está limitada por el ruido del sensor y la resolución del ADC. Para el giróscopo, el ruido típico es de $0{,}05\,^\circ/\mathrm{s}_{\mathrm{RMS}}$ con el filtro a 100 Hz, mientras que para el acelerómetro la densidad de ruido es de aproximadamente $400\,\mu g/\sqrt{\mathrm{Hz}}$. En cuanto a resolución por cuantización:

$$1\,\mathrm{LSB}_{\mathrm{gyro}} \approx 0{,}0076\,^\circ/\mathrm{s} \quad (\pm 250\,^\circ/\mathrm{s})$$

$$1\,\mathrm{LSB}_{\mathrm{acc}} \approx 61\,\mu g \quad (\pm 2\,g)$$

- **Ancho de banda:** Definido por el filtro pasa-bajos digital. En el giróscopo se puede configurar entre 5 y 256 Hz, y en el acelerómetro entre 5 y 260 Hz. Una mayor banda permite registrar movimientos rápidos con menor retardo, mientras que una menor banda reduce el ruido a costa de mayor latencia.

Práctica

En esta práctica se leen los datos de la MPU6050 con el arduino y se los envía por el puerto serial para que matlab los lea y se puedan ver a través de simulink.

Código Arduino

A continuación, parte del código implementado en la práctica:

```
    #include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup() {
  Serial.begin(115200);

  // Inicializar I2C y el sensor
  if (!mpu.begin()) {
    Serial.println("No se encontro el MPU6050, verifique conexiones!")
       ;
    while (1) {
      delay(10);
    }
  }

  // Configurar el rango del acelerometro
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

  // Configurar el rango del giroscopio
  //mpu.setGyroRange(MPU6050_RANGE_500_DEG);

  // Configurar el filtro de paso bajo
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  // Encabezado de datos
  Serial.println("Ax,Ay,Az");
}

void loop() {
  // Obtener lecturas del sensor
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  float d1 = a.acceleration.x;
  float d2 = a.acceleration.y;
  float d3 = a.acceleration.z;

  // Enviar datos a Matlab
  matlab_send(d1, d2, d3);

  delay(1000);
}

void matlab_send(float dato1, float dato2, float dato3) {
  Serial.write("abcd");   // marcador de inicio
  byte *b;

  b = (byte *) &dato1;
  Serial.write(b, 4);

```

```
53    b = (byte *) &dato2;
54    Serial.write(b, 4);
55
56    b = (byte *) &dato3;
57    Serial.write(b, 4);
58  }
```

Listing 1: Código de control en Arduino

Simulink

Se utilizó un esquemático de simulink provisto por la catedra para leer la información del puerto serie. En la figura 1 se muestra lo obtenido.

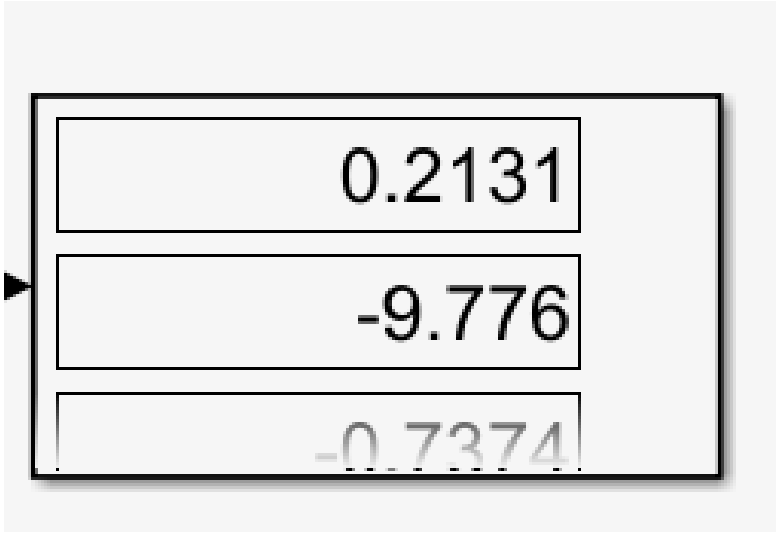

Figura 1: Captura de pantalla de simulink.

Otras experiencias

Además de lo previamente mencionado se realizaron calculos con los datos provistos por la IMU, por ejemplo calcular el angulo integrando la aceleración.

0.1. Conclusión

Realizando esta experiencia nos familiarizamos con la IMU que usaremos en el proyecto y manipulamos su información para obtener los datos que necesitabamos según el contexto.