Diseño y Construcción de un Nivelador y Receptor Láser

Pau Zaragoza Gallardo
Brian Fernando Arias

FIB UPC
Ingeniería Informática - Q1 2024/25

23/01/2025

Resumen

Este documento presenta el diseño y desarrollo de un nivelador y receptor láser, un dispositivo destinado a aplicaciones de medición y alineación con alta precisión. El proyecto incluye la selección y adquisición de materiales y componentes, el diseño de un circuito electrónico y su respectiva placa PCB, la creación de un *casing* utilizando software de modelado 3D, y la integración de todos los componentes en un sistema funcional.

Marco Teórico

Niveladores Láser

Un nivelador láser es una herramienta que utiliza un rayo láser para nivelar superficies, alinear estructuras o realizar mediciones precisas.

Existen tres tipos principales de niveladores láser: de punto, de línea y rotativo. Los niveladores láser de punto proyectan uno o varios puntos de luz sobre una superficie, lo que los hace ideales para transferir puntos de referencia desde el suelo al techo o viceversa. Suelen ser usados en tareas como la instalación de luminarias o conductos eléctricos. Los niveles láser de línea proyectan una o varias líneas de luz. Son útiles para trabajos como la instalación de estantes, azulejos, puertas y ventanas. Por último, los niveladores láser rotativos giran rápidamente 360º para proyectar una línea nivelada alrededor de una habitación o espacio. Son muy utilizados en grandes obras de construcción, como la nivelación de terrenos, instalación de techos y trabajos de excavación.

En cuanto al color del láser, encontramos principalmente dos opciones: el láser verde y el rojo. El primero ofrece un mayor alcance y mayor visibilidad para el ojo humano; sin embargo, resulta más caro y tiene un mayor consumo energético. El segundo tiene un alcance menor y en espacios con mucha luz resulta difícil de ver para el ojo humano; pero en contraparte, resulta más barato tanto económicamente como energéticamente.

Receptor Láser

Un receptor láser es un dispositivo diseñado para detectar y recibir el rayo láser emitido por un nivelador láser. Su principal función es ampliar el rango de uso del láser y permitir la detección del haz incluso en condiciones de alta luminosidad o cuando no es visible a simple vista.

Objetivos

Las funcionalidades que nos marcamos para nuestro proyecto fueron:

Nivelador Láser

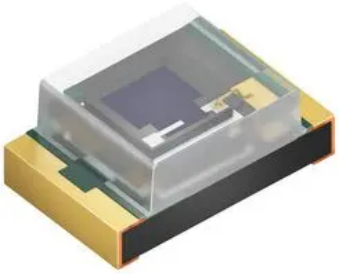
- Láser rotativo 360º
- Medidor de desnivel
- Capacidad de nivelarse autónomamente

Receptor Láser

- Capacidad de detectar si el nivelador está por encima, por debajo o centrado
- Pantalla y zumbador para

Selección de Componentes

Receptor Láser

Fotodiodo
SFH 2716 • Distribuidor: Farnell • Precio: 0,469€ (25+) | 0,403€ (50+) • Encapsulado: 0805 • Nº Pines: 2 • Longitud de Onda Pico de Sensibilidad: 620nm • Anchura: 1.25mm 

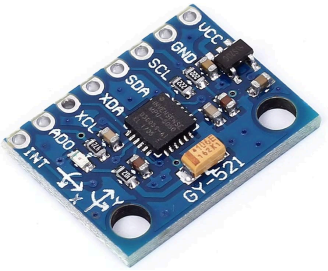
ADC
MCP3008-I/SL • Distribuidor: Farnell • Precio: 2,71€ (1+) | 2,25€ (25+) • Encapsulado: SOIC • Nº Pines: 16 • Nº Canales: 8 • Resolución: 10 bits • Protocolo de comunicación: SPI 

Zumbador activo
ABI-009-RC • Distribuidor: Farnell • Precio: 2,24€ (1+) | 2,11€ (25+) • Encapsulado: Pines • Nº Pines: 2 • Nivel de Sonido: 85 dB 

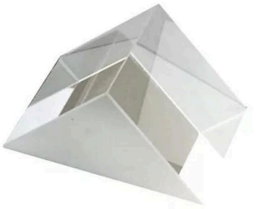
Display
0.91" OLED Display • Distribuidor: Aliexpress • Precio: 1,01€ (1+) | 6,19€ (5+) • Encapsulado: Pines • Nº Pines: 4 • Resolución: 128x32 • Controlador: SSD1306 • Protocolo de comunicación: I2C   

Nivelador Láser

Láser
CHT1230 • Distribuidor: Aliexpress • Precio: 3,32€ (1+) • Encapsulado: Cables • Nº Cables: 2 • Longitud de Onda: 650nm • Clase: 3a   

IMU
MPU6050 • Distribuidor: Aliexpress • Precio: 2,24€ (1) | 1,746€ (5) • Encapsulado: Board • Nº Pines: 8 • Protocolo de comunicación: I2C   

Pan & Tilt
Pan&Tilt • Distribuidor: Aliexpress • Precio: 14,51€ (1+) • Tamaño: 52.81 x 33.43 mm • Nº Servos: 2 • Ejes de movimiento: 2 

Prisma
Prisma • Distribuidor: Aliexpress • Precio: 3,49€ (1+) • Tamaño: 15 x 15 x 15 mm • Ángulo: 90º 

Como microcontrolador de ambos usamos un Arduino Uno R3.

Componentes Casing

Barra roscada acero inoxidable 100 mm 

Imanes de neodimio 10x3mm 

Como material para el casing optamos por usar PLA.

Diseño del Circuito

Para el diseño tanto de esquemáticos como de layouts de PCBs utilizamos la herramienta gratuita KiCad 8.0.

Receptor Láser

Para el diseño del circuito del receptor láser utilizamos 48 fotodiodos. Estos están repartidos en dos columnas, alternando las alturas para obtener así una mayor precisión. La otra opción era espaciar más los fotodiodos para cubrir una mayor área pero optamos por priorizar la precisión.

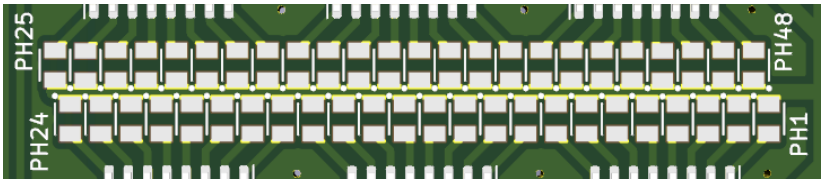


Figura 1: Disposición en dos columnas de los 48 fotodiodos alternados

Los fotodiodos están conectados en modo fotoconductivo, de tal forma que el cátodo está conectado a 5V y leemos la variación de corriente del ánodo.

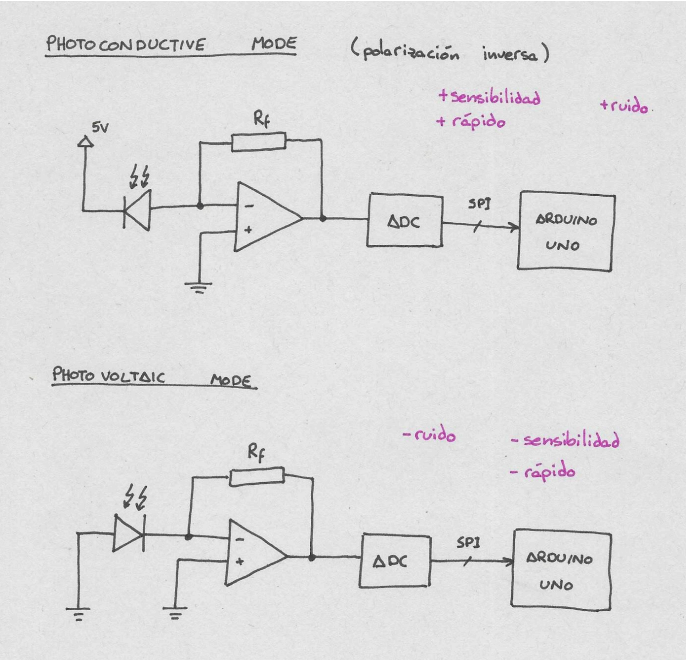


Figura 2: Modos de conexión de un fotodiodo PIN: fotoconductivo y fotovoltaico

Tenemos además 6 ADCs con 8 canales cada uno los cuales se encargan de leer la corriente de cada fotodiodo, transformar tal corriente de analógica a digital y proporcionar esos datos al Arduino. Los ADCs se comunican con el Arduino mediante SPI.

También tenemos un display OLED de 0.91" conectado al Arduino a través de I2C, el cual se encarga de mostrar visualmente donde se encuentra el nivel láser.

Por último tenemos un zumbador activo el cual, dependiendo de la distancia a la que se encuentre el láser del centro pita más o menos.

Esta tabla muestra las conexiones de los diferentes componentes con el Arduino Uno:

Componente	Pin	Pin Arduino
ADCs	CLK	D13
	MISO	D12
	MOSI	D11
	CS1 .. CS6	D10 .. D5
Zumbador	BUZZ	D3
Display	DISP_SDA	A4/SDA
	DISP_SCL	A5/SCL

Véase Anexo 1, Anexo 2, Anexo 3 y Anexo 4.

Nivelador Láser

Para el diseño del nivelador láser utilizamos los servos Pan&Tilt para controlar y ajustar la inclinación del láser , los cuales son controlados desde una placa Arduino Uno. Para obtener los parámetros de inclinación actual a tiempo real utilizaremos un módulo MPU6050 que obtendrá los datos de los diferentes ejes a partir de sus sensores.

El código del arduino se encargará de recolectar los datos de la IMU y según los resultados obtenidos, modificar los parámetros de los servos para que ajusten su posición y así compensar la inclinación, obteniendo un nivelador que se mantiene recto sin importar la inclinación de su base.

En la figura 3 se puede observar las conexiones realizadas y como el arduino es el nucleo que conecta los diferentes componentes.

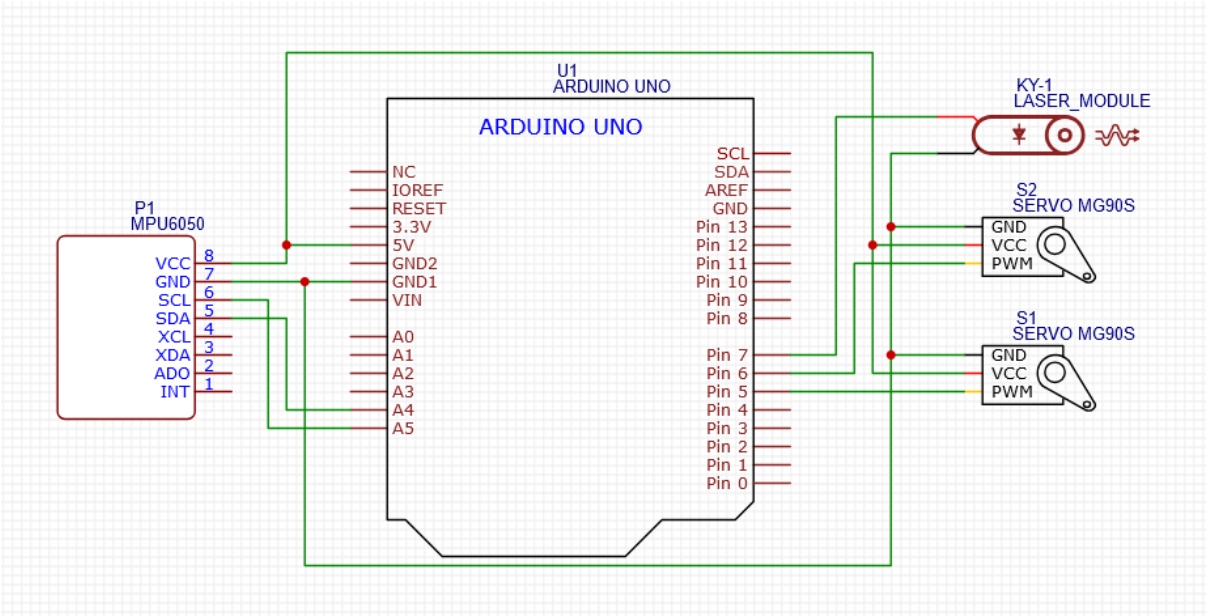


Figura 3: Diagrama de conexiones del nivelador láser.

Esta tabla muestra las conexiones de los diferentes componentes con el Arduino Uno:

Componente	Pin	Pin Arduino
Láser	LSR	D7
Servos	SV1 , SV2	D5 , D6
IMU	IMU_SDA	A4/SDA
	IMU_SCL	A5/SCL

Véase Anexo 5, Anexo 6, Anexo 7 y Anexo 8.

Diseño del Casing

Receptor Láser

En cuanto al *casing* del receptor láser decidimos dividirlo en dos: una pinza roscada para sujetar el receptor y una carcasa para proteger el circuito del receptor.

Para hacer la pinza roscada utilizamos un eje rotativo, parecido al que usan las impresoras 3D, de unos 100 mm. Cuando lo hacemos girar, este mueve un brazo de la pinza horizontalmente, el cual podemos controlar para ajustarlo a la vara a la que deseemos sujetar el receptor.

Para la carcasa del receptor no nos complicamos mucho y optamos por una caja simple con agujeros para los fotodiodos, la pantalla, el buzzer y los cables para enchufar el Arduino. Las piezas de esta carcasa se unen y sujetan mediante tornillos de 2.9 mm de ancho.

Las dos piezas se unen mediante unos imanes de neodimio.

Véase Anexo 9, Anexo 10 y Anexo 11.

Nivelador Láser

El diseño del *casing* del nivelador láser consiste en 2 partes , la primera que hará de base y se encargará de contener el módulo arduino junto con la pcb que conecta los diversos componentes además de ofrecer soporte a los servos, la segunda parte consistirá de pequeños módulos e irá conectada a los servos y se encargará de contener el láser junto al motor (no se pudo implementar debido a problemas con el primer motor). El casing no pudo imprimirse debido a falta de tiempo y varios cambios que se tuvieron que ir actualizando debido a cambios de componentes.

Véase Anexo 12 y Anexo 13.

Programación

Para programar utilizamos la IDE de Arduino.

Receptor Láser

El código del receptor láser está dividido en 3 partes.

La primera consiste en leer y almacenar los valores que nos proporcionan los ADCs de los fotodiodos. Para ello utilizamos la librería <Adafruit_MCP3008.h> la cual nos permite leer los ADCs de manera sencilla.

```C/C++
void readADCs() {
        int ph_index = 0;
        Adafruit_MCP3008* adcs[6] = {&adc1, &adc2, &adc3, &adc4, &adc5, &adc6};

        for (int i = 0; i < 6; i++) {
                for (int j = 0; j < 8; j++) {
                        photodiode_values[ph_index++] = adcs[i]->readADC(j);
                }
        }
}
```

Estos valores los guardamos en un array de enteros global para poder acceder a ellos desde otras partes del código.

```C/C++
#define TOTAL_PHOTODIODES 48
int photodiode_values[TOTAL_PHOTODIODES] = {0};
```

La segunda parte se encarga de interpretar los datos guardados y comprobar que fotodiodo apunta el láser. Para evitar posibles errores se establece un THRESHOLD para no tener en cuenta la luz normal a la hora de decidir si un fotodiodo está expuesto al láser o no.

```C/C++
const int THRESHOLD = 650;

int max_value = 0;
int max_photodiode = -1;
```

```
for (int ph = 0; ph < TOTAL_PHOTODIODES; ph++) {
        if (photodiode_values[ph] > THRESHOLD && photodiode_values[ph] > max_value) {
                max_value = photodiode_values[ph];
                max_photodiode = ph;
        }
}
```

Por último, y una vez hemos decidido a que fotodiodo apunta el láser, si es que apunta a alguno, entramos en la tercera parte: mostrar al usuario la altura mediante el display y el buzzer.

Véase Anexo X.

Nivelador Láser

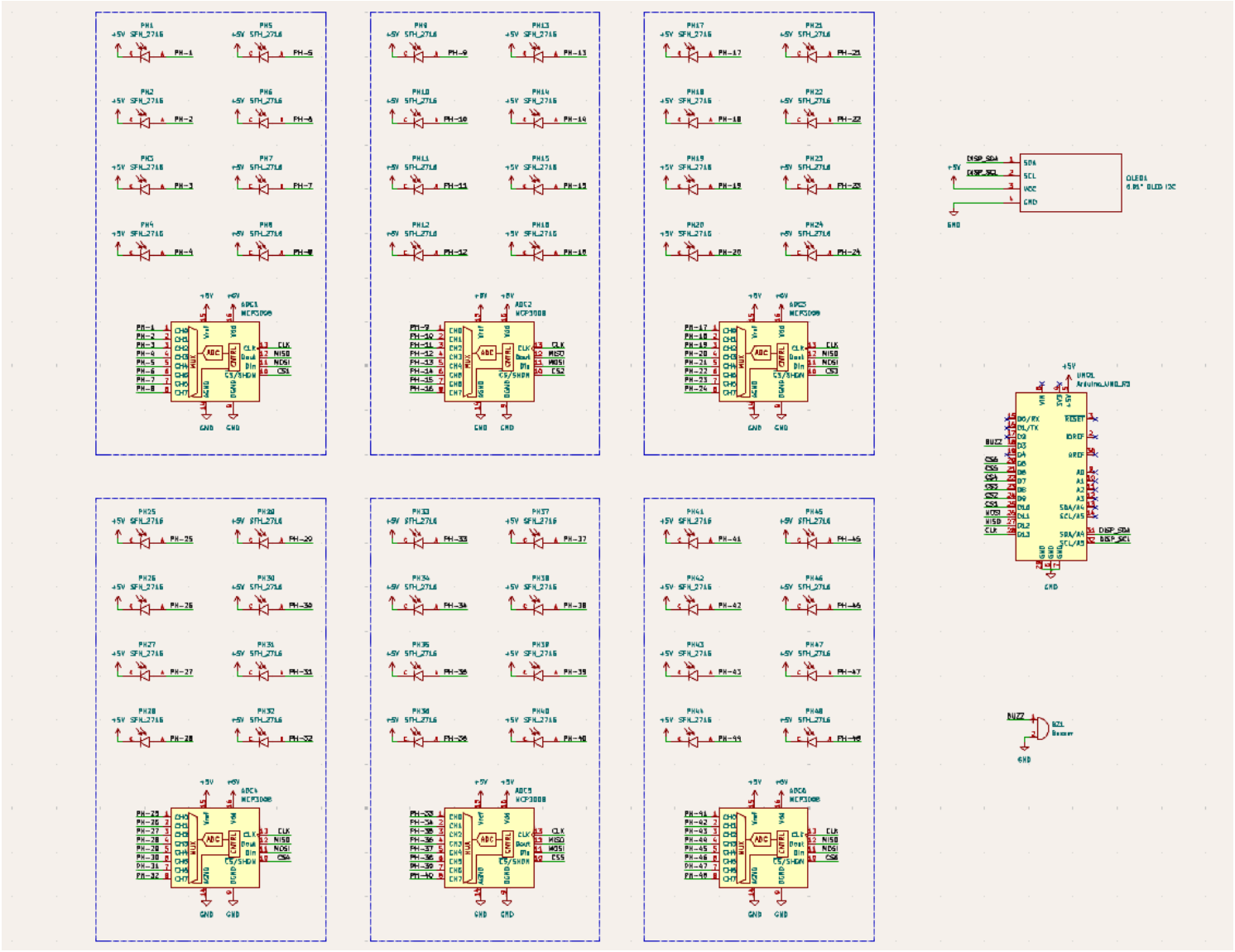
El código del nivelador primero se encarga de inicializar la MPU6050 junto con las variables para los servos. En la parte loop, se encarga de leer la MPU, guardar los datos leídos y según los datos obtener el ángulo de inclinación adecuado para los servos, los cuales serán actualizados con cada lectura. Para disminuir los movimientos bruscos y la falta de precisión, los datos son guardados en un buffer los cuales se utilizan para calcular una media y que la fluctuación de los datos sea más suave.

Resultados y Conclusiones

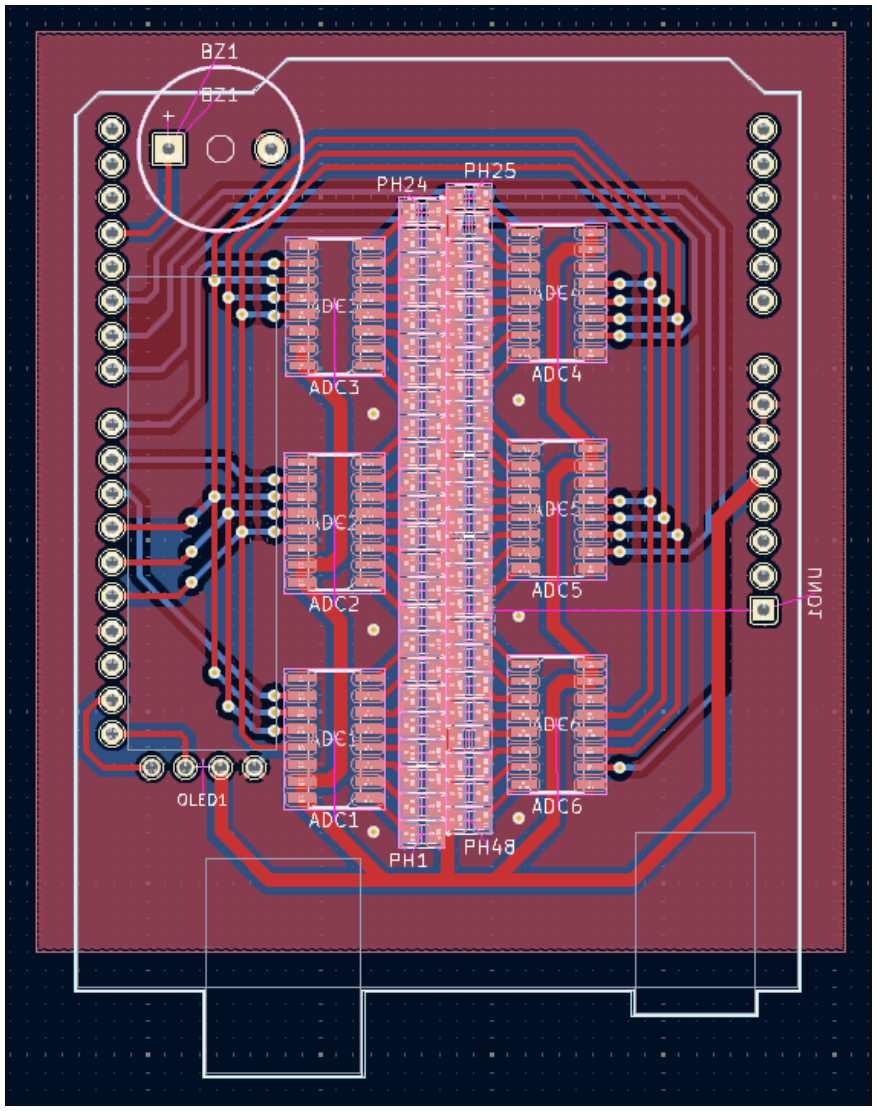
???

Anexos

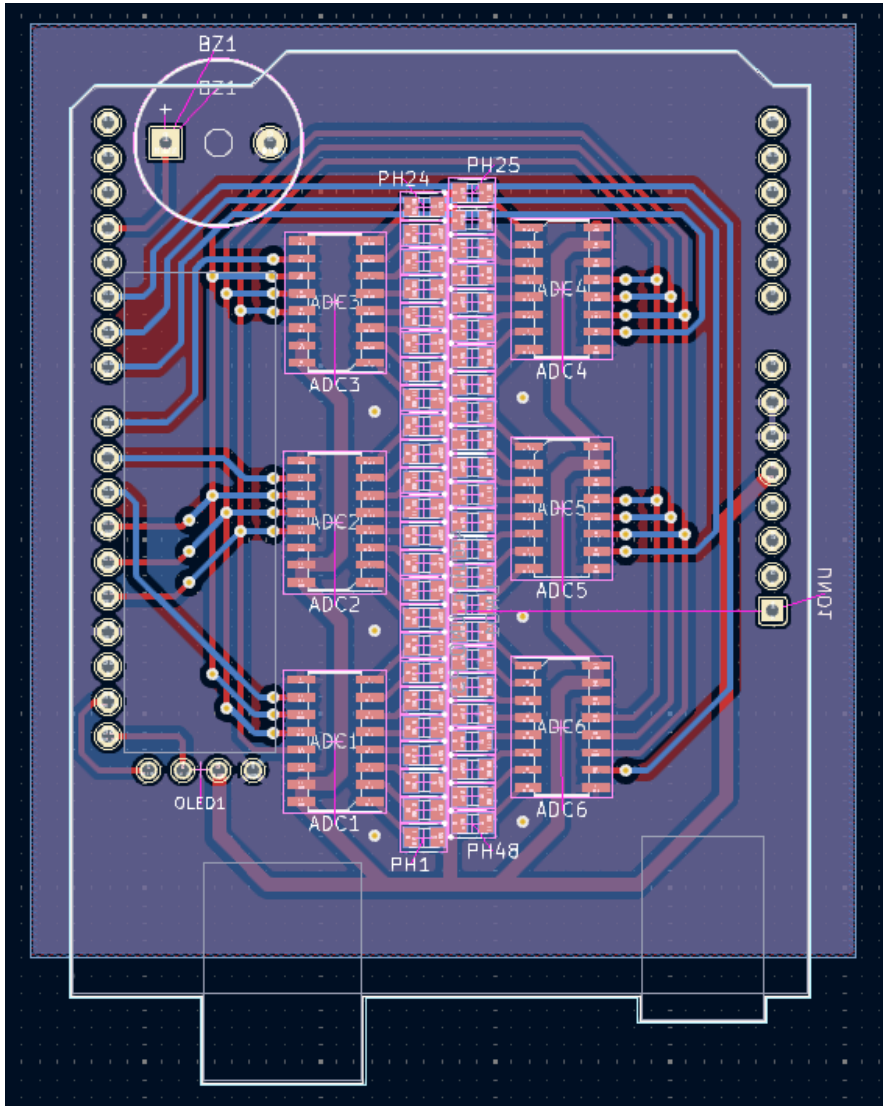
Aquí se encuentran los esquemáticos y layouts completos de los circuitos, los planos del diseño 3D y el código utilizado tanto en el receptor láser como en el nivelador.



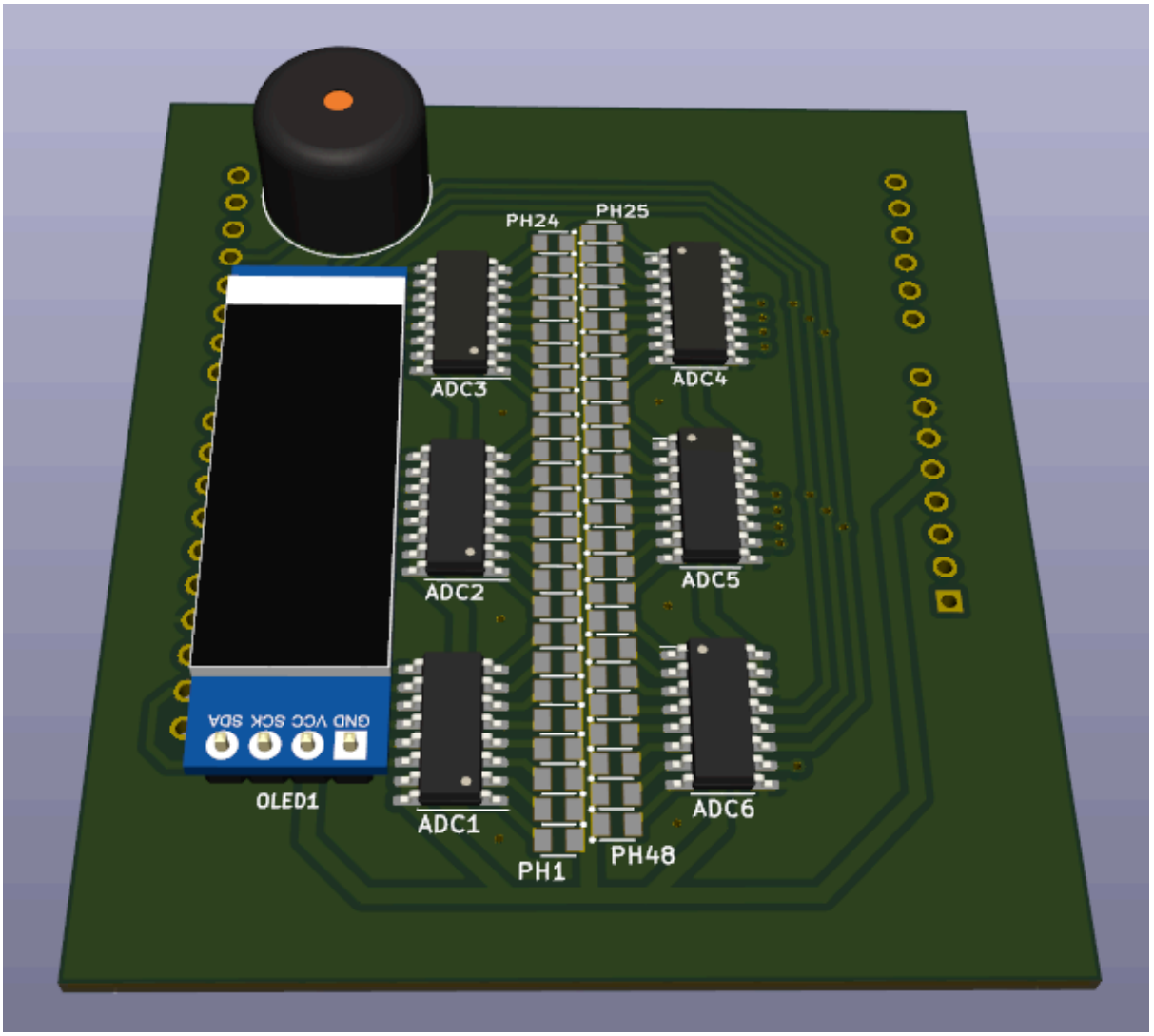
Anexo 1: Esquemático del circuito del receptor láser



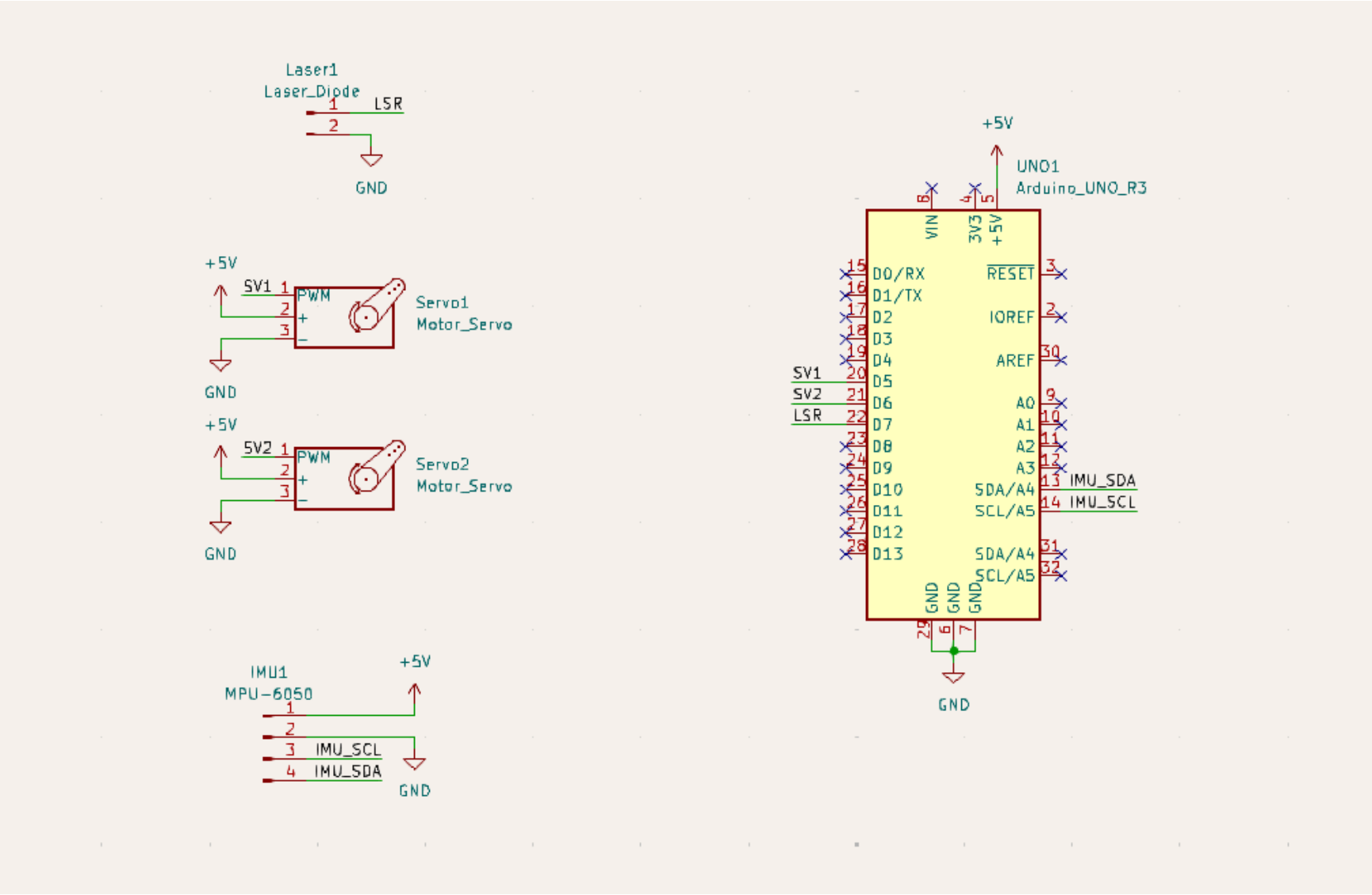
Anexo 2: Cara frontal de la PCB del receptor láser



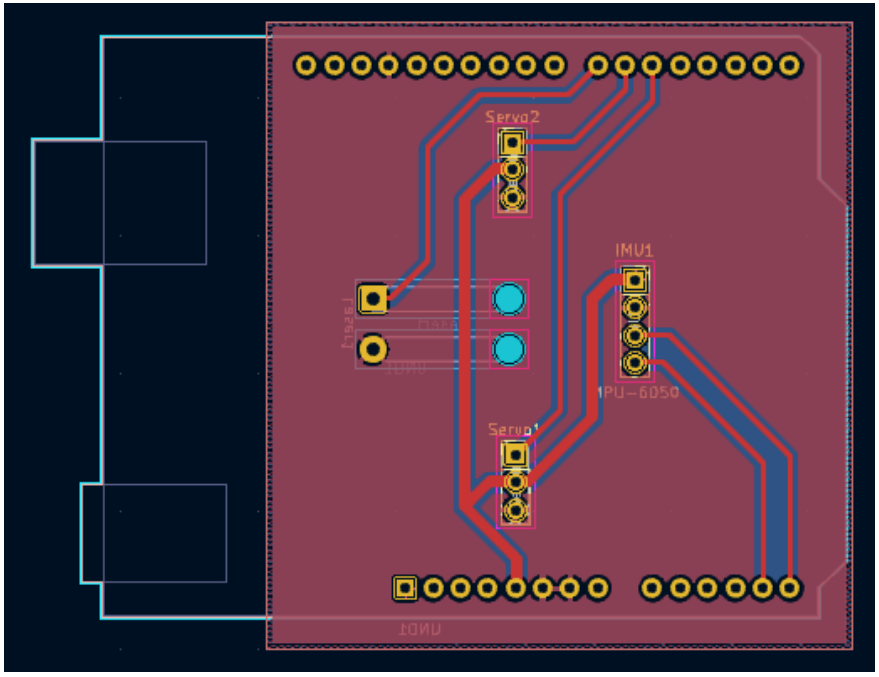
Anexo 3: Cara trasera de la PCB del receptor láser



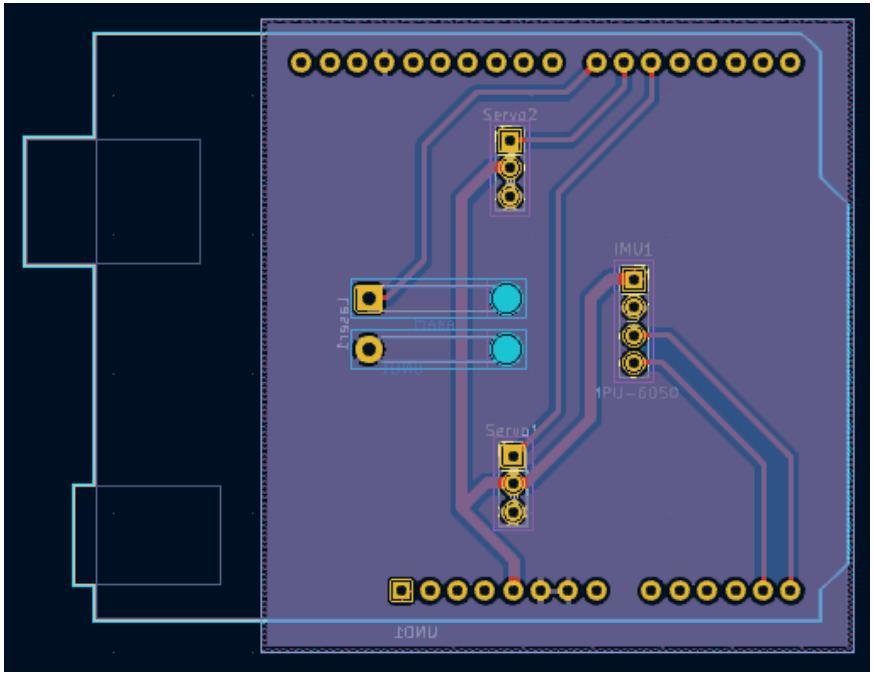
Anexo 4: Previsualización 3D de la PCB del receptor láser



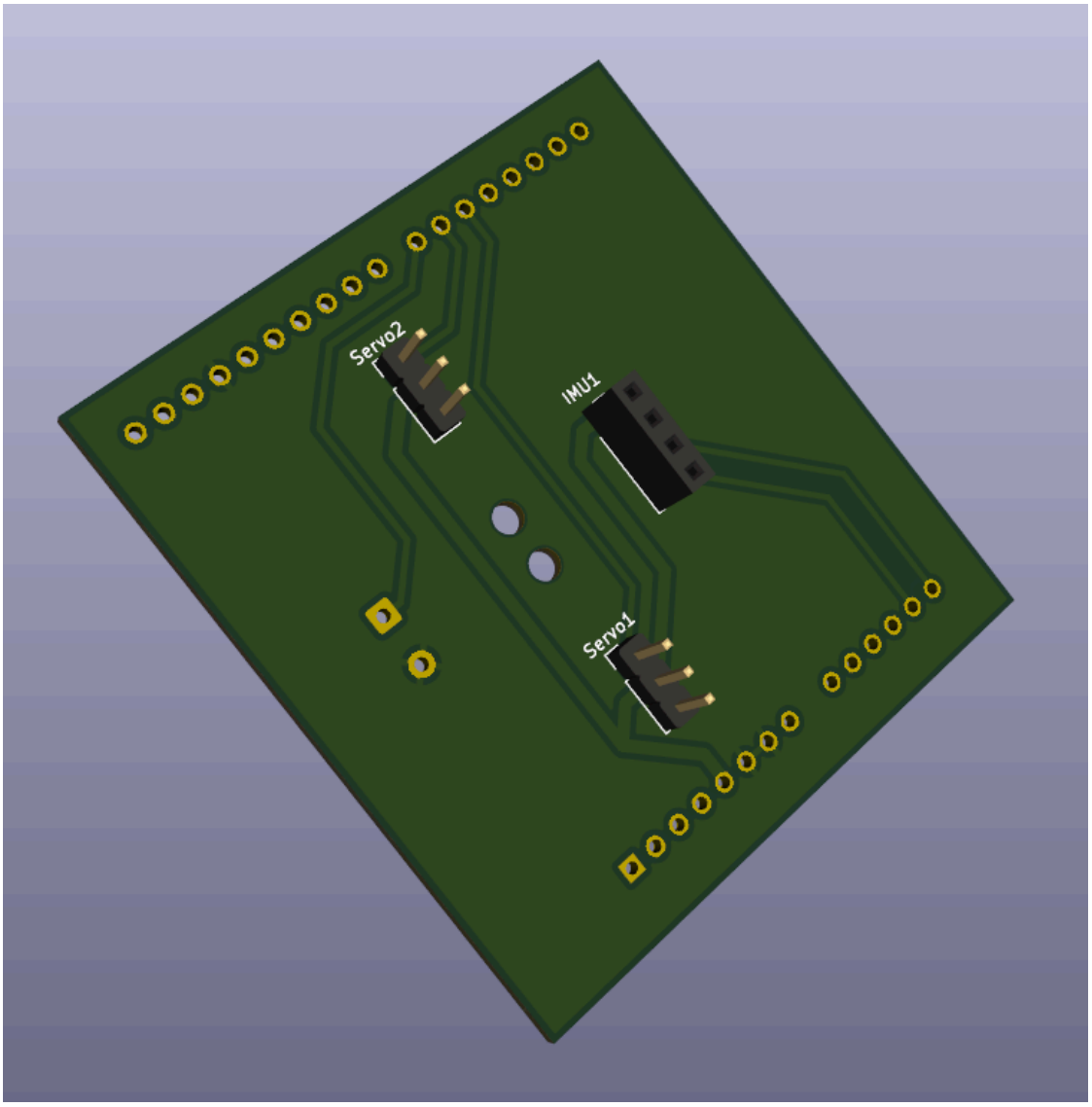
Anexo 5: Esquemático del circuito del nivelador láser



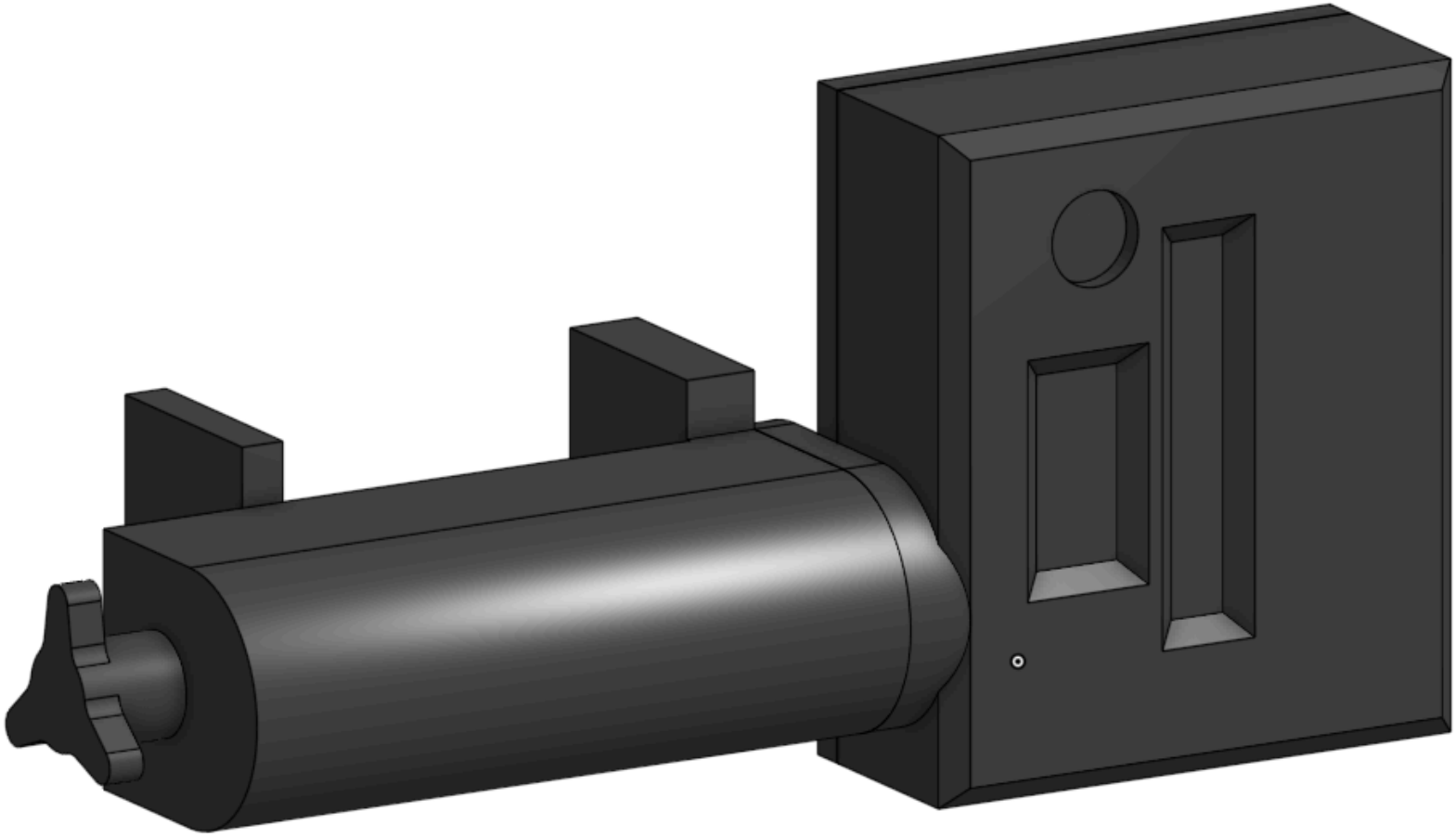
Anexo 6: Cara frontal de la PCB del nivelador láser



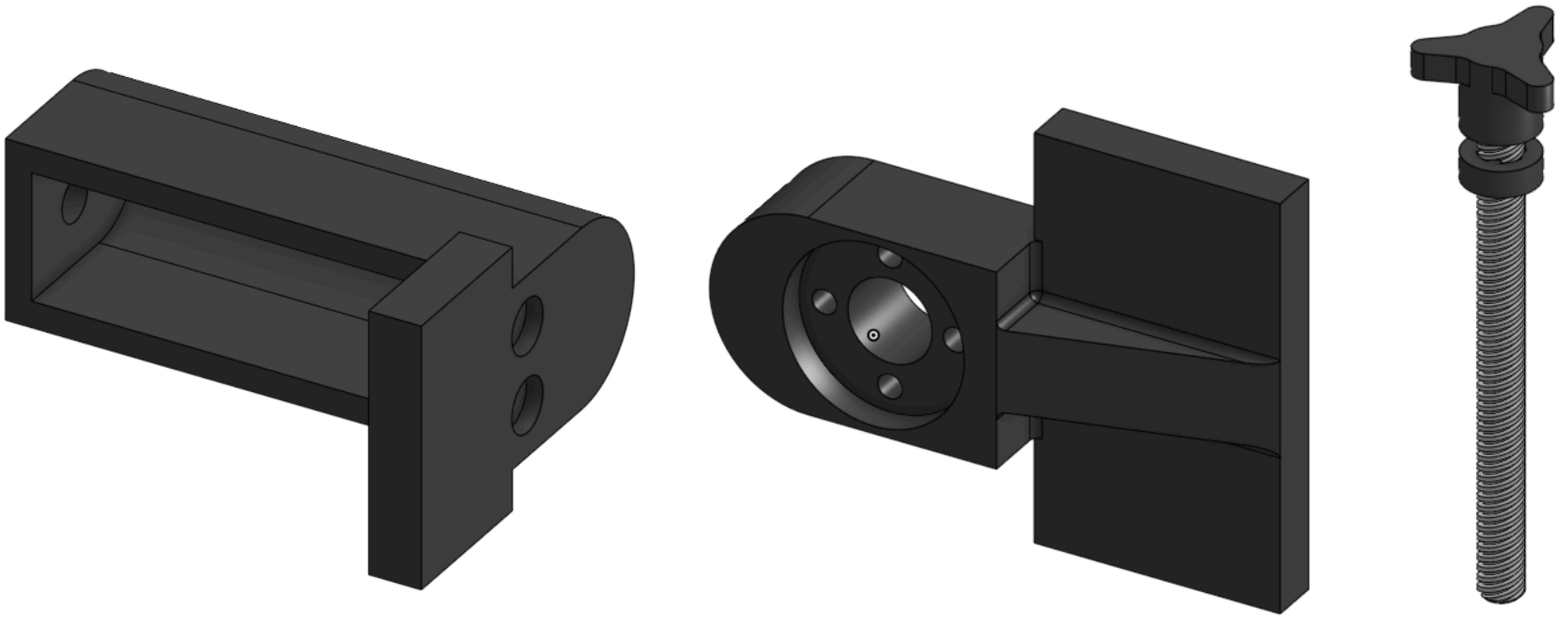
Anexo 7: Cara trasera de la PCB del nivelador láser



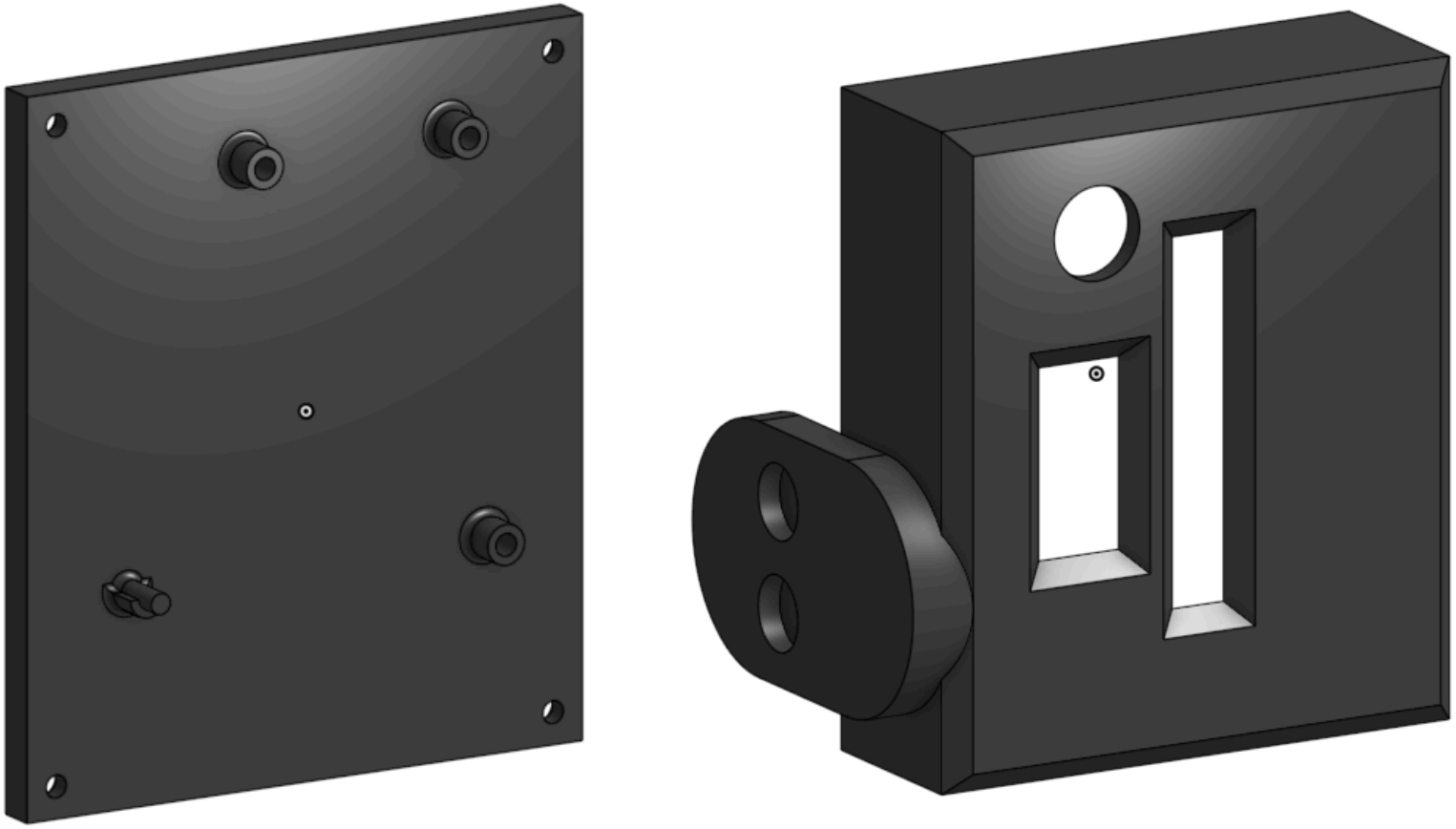
Anexo 8: Previsualización 3D de la PCB del nivelador láser



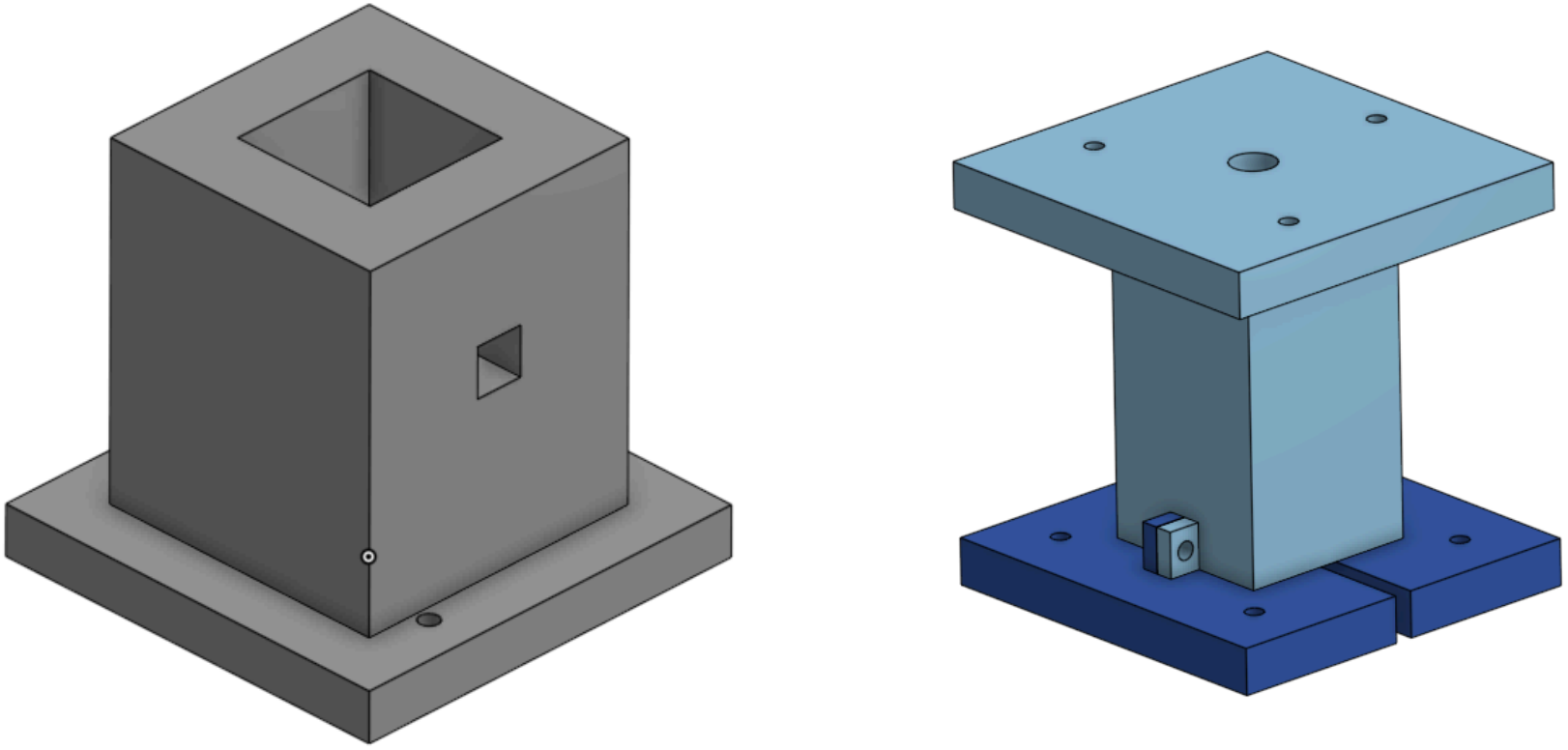
Anexo 9: Modelo 3D del *casing* completo del receptor láser



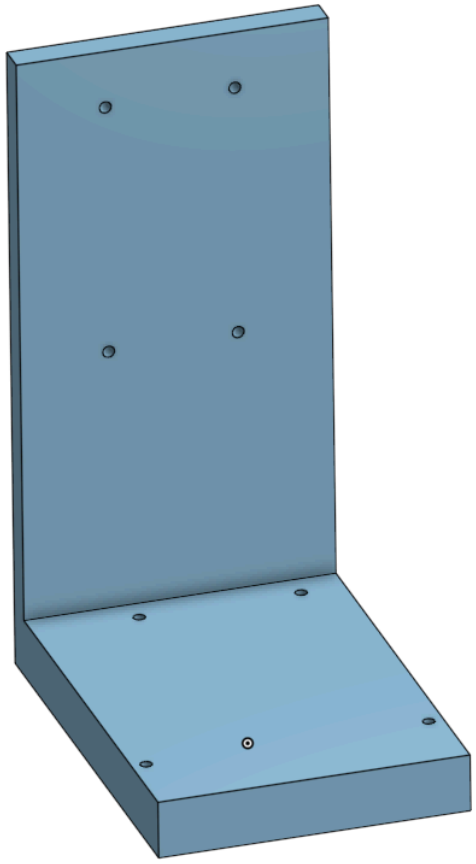
Anexo 10: Partes del modelo 3D de la pinza roscada del receptor láser



Anexo 11: Partes del modelo 3D de la carcasa del receptor láser



Anexo 12: Partes del modelo 3D del prisma y el casing del laser



Anexo 13: Partes del modelo 3D de la base del nivelador laser

C/C++

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

// Definir los servos
Servo panServo;   // Servo para el eje de giro (pan)
Servo tiltServo;  // Servo para el eje de inclinación (tilt)

// Crear objeto MPU6050
Adafruit_MPU6050 mpu;

// Variables para almacenar los valores de aceleración
float xAccel, yAccel, zAccel;
float pitchAngle, rollAngle;

// Buffers para promedio móvil
const int bufferSize = 10; // Tamaño del buffer para el promedio
float pitchBuffer[bufferSize] = {0};
float rollBuffer[bufferSize] = {0};
int bufferIndex = 0;

// Valores de mapeo para los servos
int panServoMin = 0;
```

```
int panServoMax = 180;
int tiltServoMin = 0;
int tiltServoMax = 180;

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

  // Inicializar el MPU6050
  if (!mpu.begin()) {
  Serial.println("No se encontró el MPU6050. Verifique la conexión.");
  while (1);
  }
  Serial.println("MPU6050 inicializado correctamente.");

  // Configurar el rango de los acelerómetros
  mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  // Conectar los servos a los pines
  panServo.attach(5); // Servo para el pan (giro horizontal)
  tiltServo.attach(6);   // Servo para el tilt (inclinación)

  // Inicializar los servos en sus posiciones centrales
```

```
  panServo.write(90); // Posición central para el pan
  tiltServo.write(90);   // Posición central para el tilt
}

void loop() {
  // Leer los datos del acelerómetro
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // Obtener las aceleraciones de los tres ejes
  xAccel = a.acceleration.x;
  yAccel = a.acceleration.y;
  zAccel = a.acceleration.z;

  // Calcular el ángulo de inclinación (pitch) y de giro (roll) con las aceleraciones
  float newPitch = atan2(yAccel, zAccel) * 180.0 / PI;  // Ángulo de inclinación
  float newRoll = atan2(xAccel, zAccel) * 180.0 / PI;   // Ángulo de giro

  // Actualizar los buffers
  pitchBuffer[bufferIndex] = newPitch;
  rollBuffer[bufferIndex] = newRoll;
  bufferIndex = (bufferIndex + 1) % bufferSize; // Incrementar índice circular

  // Calcular el promedio de los buffers
```

```
  pitchAngle = calculateAverage(pitchBuffer, bufferSize);
  rollAngle = calculateAverage(rollBuffer, bufferSize);

  // Mapear los valores de pitch y roll a los rangos del servo
  int tiltPosition = map(pitchAngle, -90, 90, tiltServoMin, tiltServoMax);   // Mapea el ángulo de
inclinación al servo de tilt
  int panPosition = map(-rollAngle, -90, 90, panServoMin, panServoMax);   // Mapea el ángulo de giro
al servo de pan

  // Asegurar que los valores estén dentro de los límites de los servos
  tiltPosition = constrain(tiltPosition, tiltServoMin, tiltServoMax);
  panPosition = constrain(panPosition, panServoMin, panServoMax);

  // Mover los servos a las posiciones calculadas
  tiltServo.write(tiltPosition);
  panServo.write(panPosition);

  // Imprimir los ángulos y las posiciones de los servos para depuración
  Serial.print("Pitch: ");
  Serial.print(pitchAngle);
  Serial.print("\tRoll: ");
  Serial.print(rollAngle);
  Serial.print("\tTilt: ");
  Serial.print(tiltPosition);
```

```
  Serial.print("\tPan: ");
  Serial.println(panPosition);

  delay(25); // Espera breve para permitir que los servos se muevan
}

// Función para calcular el promedio de un buffer
float calculateAverage(float buffer[], int size) {
  float sum = 0;
  for (int i = 0; i < size; i++) {
  sum += buffer[i];
  }
  return sum / size;
}
```

Anexo 14: Codigo arduino del nivelador laser, mpu y servos.