MANUAL DE PLATAFORMA

1. Introducción

Este manual proporciona una descripción detallada de la plataforma de automatización para el sistema de pesaje y distribución de bolsas de veneno para ratas desarrollado para la empresa Biota Lab. Este sistema asegura que cada bolsa cumpla con los requisitos de peso especificados antes de ser empacada. En caso de desviaciones en el peso, las bolsas son redirigidas para un nuevo llenado y pesaje.

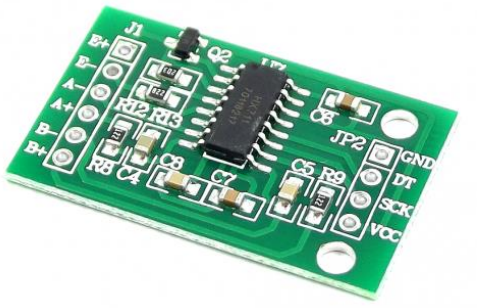
El objetivo principal es garantizar la calidad y consistencia de los productos mediante un sistema automatizado basado en componentes IoT. Además, el sistema cuenta con una aplicación móvil desplegada en AWS, que permite monitorear en tiempo real los pesos y la cantidad de paquetes procesados.

2. Descripción del Sistema

2.1. Componentes Principales

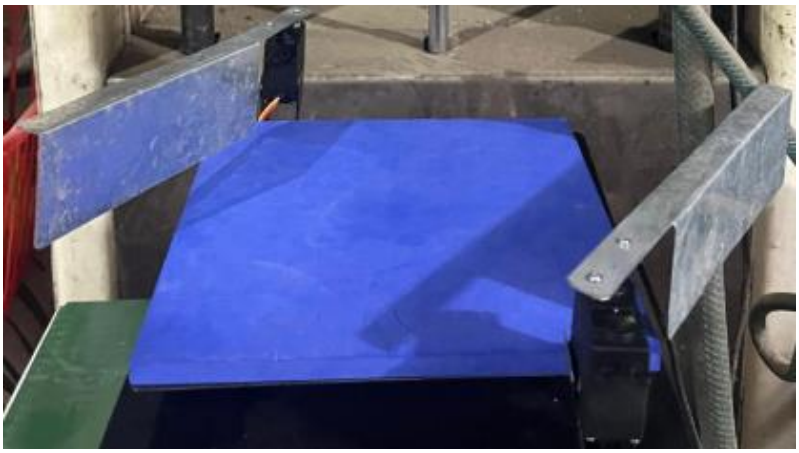
- Microcontrolador ESP32:
  - ✓ Procesa la información del sensor de peso (HX711) y controla los servomotores.
  - ✓ Facilita la conectividad a redes inalámbricas para la supervisión remota.

- Sensor HX711 (Amplificador de Celda de Carga):
  - ✓ Mide el peso de las bolsas de veneno para ratas.
  - ✓ Envía las lecturas al ESP32 para su procesamiento.



- Servomotores (2 unidades):
  - ✓ Controlan los "brazos" que direccionan las bolsas a su ubicación correspondiente.
    - Brazo 1: Redirige las bolsas con peso correcto hacia la caja de empaque.
    - Brazo 2: Redirige las bolsas con peso incorrecto hacia la zona de llenado.

- Plataforma de Pesaje:
  - ✓ Diseñada para contener las bolsas durante el proceso de pesado.
  - ✓ Conectada al sensor HX711.



- Fuente de Alimentación:
  - ✓ Proporciona energía al ESP32, el sensor HX711 y los servomotores.

- Infraestructura de Red y Aplicación Móvil:
  - ✓ La infraestructura permite el monitoreo en tiempo real mediante una aplicación móvil desplegada en AWS.
  - ✓ La aplicación muestra datos de pesos y la cantidad de bolsas procesadas.

3. Instalación y Configuración

3.1. Requisitos Previos

- Herramientas necesarias:
  - ✓ Soldador y estaño.
  - ✓ Multímetro.
  - ✓ Cables Dupont y conectores.

- Software:
  - ✓ Arduino IDE o plataforma compatible para programar el ESP32.
  - ✓ Librerías: HX711, Servo, WiFi y Firebase (opcional para conectividad con AWS).

3.2. Montaje del Hardware

- **ESP32**:
  - ✓ Conectar el ESP32 al sensor HX711 siguiendo este esquema:
    - HX711 VCC → ESP32 3.3V
    - HX711 GND → ESP32 GND
    - HX711 DT → ESP32 GPIO21
    - HX711 SCK → ESP32 GPIO22

- **Sensor HX711**:
  - ✓ Montar la celda de carga en la plataforma de pesaje y calibrar.

- **Servomotores**:
  - ✓ Conectar los pines de señal al ESP32 (GPIO18 y GPIO19).
  - ✓ Alimentar los servomotores con una fuente de 5V.

- **Fuente de Alimentación**:
  - ✓ Proporcionar energía suficiente para los componentes: 5V para los servos y 3.3V para el ESP32.

3.3. Configuración del Software

1. Abrir Arduino IDE y configurar la placa ESP32.
2. Instalar las librerías necesarias desde el gestor de librerías.
3. Subir el siguiente código base:

```
#include <ArduinoJson.h>  // Asegúrate de que esta es la librería correcta
#include "HX711.h"
#include <ESP32Servo.h>
#include <WiFi.h>
#include <PubSubClient.h>

// Definir los pines de la balanza
const int DOUT = 19;
const int CLK = 18;
```

```
// Crear objeto para la balanza
HX711 balanza;

// Definir los pines y objetos para los servomotores
Servo servo1;
Servo servo2;
int pinServo1 = 2;  // Pin para el primer servomotor
int pinServo2 = 4;  // Pin para el segundo servomotor

// Datos de conexión Wi-Fi y MQTT
const char* ssid = "DENIS 2.4";  // Cambia esto por el nombre de tu red Wi-Fi
const char* password = "46970383";  // Cambia esto por tu contraseña Wi-Fi

const char* mqttServer = "broker.emqx.io";  // Dirección del servidor MQTT
const int mqttPort = 1883;  // Puerto del servidor MQTT
const char* mqttUser = "gpti";  // Usuario MQTT
const char* mqttPassword = "untels123";  // Contraseña MQTT
const char* mqttTopic = "monitoreopaquetes/gpti";  // Tópico al que publicar

WiFiClient espClient;
PubSubClient client(espClient);

// Variables para acumular el movimiento del servomotor
int acumulacionMovimiento = 0;  // Contador de movimientos
const int limiteMovimiento = 50;  // Limite para considerar "contenedor lleno"
int contenedoresLlenos = 0;  // Contador de contenedores llenos

void setup() {
  // Iniciar comunicación serial
  Serial.begin(115200);

  // Conectar a la red Wi-Fi una sola vez
  connectToWiFi();

  // Configurar el cliente MQTT
  client.setServer(mqttServer, mqttPort);
  client.setCallback(mqttCallback);

  // Inicializar el sensor de la balanza
  balanza.begin(DOUT, CLK);
  Serial.print("Lectura del valor del ADC:  ");
  Serial.println(balanza.read());
  Serial.println("No ponga ningun objeto sobre la balanza");
  Serial.println("Destarando...");
  Serial.println("...");

  // Establecer la escala de la balanza
```

```
  balanza.set_scale(-453.11665);  // Establecemos la escala adecuada
  balanza.tare(20);  // Establece la tara (pone la balanza a cero)

  // Inicializar los servos
  servo1.attach(pinServo1);
  servo2.attach(pinServo2);

  // Colocar los servos a 0 grados inicialmente
  servo1.write(0);
  servo2.write(0);

  // Espera para que los servos se posicionen
  delay(1000);

  Serial.println("Listo para pesar");
}

void loop() {
  // Verificar la conexión MQTT
  if (!client.connected()) {
    reconnectMQTT();  // Reconectar solo si es necesario
  }
  client.loop();  // Mantener la comunicación MQTT

  // Leer el peso
  float peso = balanza.get_units(20);  // Obtener el peso promedio de 20 lecturas

  // Imprimir el peso en gramos
  Serial.print("Peso: ");
  Serial.print(peso, 3);
  Serial.println(" g");

  // Crear el objeto JSON para enviar
  StaticJsonDocument<200> doc;
  doc["peso"] = peso;
  doc["contenedorLleno"] = false;  // Inicialmente, no está lleno
  doc["cantidad"] = acumulacionMovimiento;
  doc["contenedoresLlenos"] = contenedoresLlenos;  // Agregar cantidad de contenedores llenos

  // Verificar si el peso es menor a 50 gramos (mover servo1)
  if (peso < 25 && peso > 10) {
    // Mover el primer servomotor
    servo1.write(90);
    Serial.println("Moviendo Servo 1 a 110 grados");
    delay(800);  // Esperar 1 segundo

    // Regresar el primer servomotor a 0 grados
    servo1.write(0);
```

```
    Serial.println("Regresando Servo 1 a 0 grados");
    delay(800);  // Esperar 1 segundo

    // No se publica nada ni se actualizan variables
  }
  // Verificar si el peso es mayor o igual a 50 gramos (mover servo2 y publicar datos)
  else if (peso >= 25) {
    // Mover el segundo servomotor
    servo2.write(110);
    Serial.println("Moviendo Servo 2 a 110 grados");
    delay(800);  // Esperar 1 segundo

    // Regresar el segundo servomotor a 0 grados
    servo2.write(0);
    Serial.println("Regresando Servo 2 a 0 grados");
    delay(800);  // Esperar 1 segundo

    // Acumular el movimiento
    acumulacionMovimiento++;

    // Verificar si la acumulación de movimientos alcanza el límite
    if (acumulacionMovimiento >= limiteMovimiento) {
      // El contenedor está lleno
      doc["contenedorLleno"] = true;
      Serial.println("Contenedor lleno");

      // Resetear la acumulación
      acumulacionMovimiento = 0;

      // Incrementar el contador de contenedores llenos
      contenedoresLlenos++;
    }

    // Convertir el objeto JSON a un string
    char jsonBuffer[256];
    serializeJson(doc, jsonBuffer);

    // Publicar el peso en el tópico MQTT en formato JSON
    client.publish(mqttTopic, jsonBuffer);
  }
}


// Función para conectar a Wi-Fi
void connectToWiFi() {
  Serial.println("Conectando a Wi-Fi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(3000);
```

```
    Serial.print(".");
  }
  Serial.println("Conectado a Wi-Fi");
}

// Función para reconectar al servidor MQTT
void reconnectMQTT() {
  while (!client.connected()) {
    Serial.print("Conectando al servidor MQTT...");
    if (client.connect("ESP32Client", mqttUser, mqttPassword)) {
      Serial.println("Conectado!");
    } else {
      Serial.print("Error de conexión. Intentando nuevamente en 5 segundos...");
      delay(5000);
    }
  }
}

// Callback que se ejecuta cuando se recibe un mensaje del servidor MQTT
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  // Esta función la puedes personalizar si necesitas manejar mensajes recibidos
  Serial.println("Mensaje recibido en el tópico:");
  Serial.println(topic);
}
```

4. Configurar la conexión a AWS para el monitoreo de datos en tiempo real.

4. Operación del Sistema

4.1. Inicio del Sistema

1. Energizar la fuente de alimentación.
2. Esperar a que el ESP32 inicie y calibra el sensor HX711.
3. Verificar que los servomotores estén en sus posiciones iniciales.

4.2. Procedimiento de Funcionamiento

1. Colocar la bolsa de veneno sobre la balanza.
2. El sensor HX711 medirá el peso y enviará la lectura al ESP32.
3. El ESP32 tomará decisiones basadas en el peso:
   - ✓ Si el peso está dentro del rango permitido, se activa el **Brazo 1**.
   - ✓ Si el peso está fuera del rango, se activa el **Brazo 2**.
4. El proceso se repite hasta que todas las bolsas sean procesadas.
5. La aplicación móvil desplegada en AWS permite el monitoreo del proceso en tiempo real.

5. Mantenimiento del Sistema

5.1. Tareas Periódicas

- Calibrar el sensor HX711 mensualmente.
- Revisar la alineación y lubricación de los servomotores.
- Limpiar la plataforma de pesaje para evitar acumulación de residuos.

5.2. Resolución de Problemas

- **Problema**: El sensor no mide correctamente.
  - **Solución**: Verificar las conexiones y recalibrar.

- **Problema**: Los servomotores no responden.
  - **Solución**: Revisar la fuente de alimentación y conexiones.