BOTZILLA P.I.S.

David Gustavo Toledo Vega
Universidad Nacional de Loja
david.toledo@unl.edu.ec

César Daniel Ramos Merchán
Universidad Nacional de Loja
cesar.d.ramos@unl.edu.ec

Marco Omar Chiliguano Buri
Universidad Nacional de Loja
marco.chiliguano@unl.edu.ec

César Stif López Cevallos
Universidad Nacional de Loja
cesar.s.lopez@unl.edu.ec

Alexis David Jara Armijos
Universidad Nacional de Loja
alexis.jara@unl.edu.ec

I. Tema

Control Remoto de un Brazo Robótico a través de Software (TeleRobot)

II. Objetivo general

- Desarrollar e implementar un sistema de control remoto a través de una aplicación web para manejar telerobot, con la finalidad de un mejor control y automatización para realizar tareas específicas como es la recolección de basura.

III. Objetivos específicos

- Diseñar una aplicación web sencilla y comprensible para los usuarios para poder controlar el brazo robótico de manera remota, dicha interfaz deberá ofrecer controles precisos para cada articulación, a su vez guardar información como, datos de peso, posición de movimiento, transmisión en tiempo real y reconocimiento de objetos.
- Configurar y programar el módulo ESP32 para la implementación segura y estable a través de internet, permitiendo de esta manera una comunicación en tiempo real entre la aplicación web y el brazo robótico, el ESP 32 actuará como el puente para recibir instrucciones de la aplicación.
- Implementar un sistema de recolección de datos que registre y guarde información sobre los pesos de la basura que recolecta y a su vez las posiciones exactas a las que se moverá durante su operación, dichos datos se usarán para analizar la eficiencia del brazo y optimizar su rendimiento y asegurar dicha precisión y manipulación de los residuos.

IV. Problemática

La Universidad Nacional de Loja, constantemente pone a prueba sus estudiantes asegurando una formación profesional de calidad, mediante P.I.S (Proyecto Integrador de Saberes) mide las capacidades de los estudiantes y cómo se desenvuelven en la resolución de una necesidad real. La necesidad que se nos fue asignada se relaciona con la seguridad del personal de UNL, el personal diariamente se expone a diferentes riesgos tóxicos que pueden atentar contra su salud, por lo que mediante un brazo robótico denominado "Telerobot" el cual pueda ser manejado mediante una aplicación sea capaz de realizar los movimientos requeridos y recolectar desechos de manera eficiente, de esta manera procuramos bajar el nivel de riesgo.

V. Propuesta

Introducción

El desarrollo tecnológico hoy en día es algo que nos presenta diferentes oportunidades y abre un mundo nuevo, en este mundo nosotros tenemos la capacidad de pensar en grande y llevarlo al mundo real, una prueba clara de esto se realiza mediante la Universidad Nacional de Loja (UNL), la universidad mencionada realizó la implementación de un proyecto integrador de saberes denominado como P.I.S, por medio de este proyecto nosotros los estudiantes nos enfrentamos a un desafío con el fin de poner a prueba nuestras destrezas y cómo aplicar de manera efectiva los conocimientos adquiridos, la problemática que se nos fue asignada se relaciona con brindar ayuda al personal de UNL, el cual se expone diariamente a la recolección de desechos peligrosos, como grupo optamos por la creación de un brazo robótico el cual se podrá controlar mediante una aplicación y ayude al personal de UNL con la reducción del peligro al que se enfrentan diariamente.

Desarrollo

Lenguajes de programación: Para la programación del ESP32 se usará Arduino IDE, que permite la manipulación de servomotores, sensores y motores paso a paso mediante el controlador "Driver Motor Pasos A4988 Cnc Nema Ardu".

Brazo Robótico: En la construcción de los diferentes componentes de software que intervendrán en funcionamiento y manejo remoto del brazo robótico se usarán diferentes tecnologías que se especializan en tareas específicas.

Arduino: En este lenguaje basado en C/C++ incluye librerías para programar el funcionamiento de los distintos módulos y componentes:

- **Sensores**:
  - AdafruitSensor.h: Proporciona una base para trabajar con diferentes sensores, especialmente los de Adafruit.
  - NewPing.h: Para sensores de distancia ultrasónicos como el HC-SR04.

- **Servomotores**:
  - ESP32Servo.h: Biblioteca específica para controlar servomotores con el ESP32.
  - VarSpeedServo.h: Biblioteca para controlar servos con velocidad variable.

Comunicación WiFi:

- WiFi.h: Biblioteca estándar para manejar la conectividad WiFi en ESP32.
- WiFiManager.h: Permite gestionar la configuración WiFi de manera sencilla.
- ESPAsyncWebServer.h: Permite crear servidores web asíncronos en el ESP32.
- AsyncTCP.h: Biblioteca necesaria para servidores web asíncronos.
- PubSubClient.h: Para la comunicación MQTT sobre Wi-Fi.
- WiFiClientSecure.h: Para conexiones WiFi seguras utilizando TLS.

Funciones principales del programa en Arduino:

- Conectar el brazo robótico con la aplicación web, a través de un servidor que conforma un nodo de intercambio de información entre ambos componentes, para este fin la placa ESP32 integra un módulo WIFI que al conectarse a una red local generará una IP local con la que se vincula el brazo con el servidor.
- Generar el movimiento, el software del brazo está diseñado para usar las órdenes que recibe de la aplicación web, como parámetros para realizar los movimientos.
- Enviar información como el estado del brazo, errores de funcionamiento, información de los objetos recogidos y el video obtenido a través de ESP32 CAM.

Aplicación Web: Para la aplicación web se usará lenguaje para el frontend HTML y CSS, React, y para la parte de backend se usará Django, que es un framework de desarrollo web de código abierto escrito en Python que a su vez implementa MySQL para guardar los datos del sensor de peso.

HTML/CSS y JavaScript (React): Para el diseño y la construcción de la aplicación web (Front end), HTML para la maquetación, CSS para dar un estilo llamativo y amigable, y con Javascript se definirán las funcionalidades de visualización, de interacción y de envío de comandos que ordenarán el movimiento del robot. Esta comunicación será posible gracias a la IP obtenida del módulo WIFI contenido en la placa.

CÓDIGO JAVASCRIPT

```
const sliderCam = document.getElementById('camera');
const anguloCam = document.getElementById('anguloCam');

const sliderBas = document.getElementById('base');
const anguloBas = document.getElementById('anguloBas');

const sliderHom = document.getElementById('hombro');
const anguloHom = document.getElementById('anguloHom');

const sliderCod = document.getElementById('codo');
const anguloCod = document.getElementById('anguloCod');

const sliderMun = document.getElementById('muneca');
const anguloMun = document.getElementById('anguloMun');

const sliderPin = document.getElementById('pinza');
const anguloPin = document.getElementById('anguloPin');

const buttonForward = document.getElementById('forward');
const buttonBackward = document.getElementById('backward');
const buttonLeft = document.getElementById('left');
const buttonRight = document.getElementById('right');

let directionValue = {
    forward: 0,
    backward: 0,
    left: 0,
    right: 0
};

const reset = document.getElementById('reset');

function fetchGet() {
    let formControl = new FormData(document.getElementById('formcontrol'));
    formControl.append('camera', sliderCam.value);
    formControl.append('base', sliderBas.value);
    formControl.append('hombro', sliderHom.value);
    formControl.append('codo', sliderCod.value);
    formControl.append('muneca', sliderMun.value);
    formControl.append('pinza', sliderPin.value);
    formControl.append('forward', directionValue.forward);
    formControl.append('backward', directionValue.backward);
    formControl.append('left', directionValue.left);
    formControl.append('right', directionValue.right);

    formControl.append('esp32_ip', document.getElementById('esp32_ip').value);

```

```
    let parametros = new URLSearchParams(formControl).toString();
    let url = `http://127.0.0.1:8000/control/?${parametros}`;

    fetch(url, {
        method: 'GET',
        headers: {
            'Accept': 'text/plain',
        }
    }).then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok ' + response.statusText);
        }
        return response.text();
    }).then(data => {
        console.log('Success:', data);
    }).catch(error => {
        console.error('Error:', error);
    });

    console.log(url);
}

let sliders = [sliderCam, sliderBas, sliderHom, sliderCod, sliderMun, sliderPin];

function updateSliderValue(slider, angulo) {
    slider.addEventListener('input', function() {
        angulo.textContent = slider.value;
    });
    angulo.textContent = slider.value;
}

updateSliderValue(sliderCam, anguloCam);
updateSliderValue(sliderBas, anguloBas);
updateSliderValue(sliderHom, anguloHom);
updateSliderValue(sliderCod, anguloCod);
updateSliderValue(sliderMun, anguloMun);
updateSliderValue(sliderPin, anguloPin);

sliders.forEach(slider => {
    slider.addEventListener('input', function() {
        fetchGet();
    });
});

reset.addEventListener('click', function() {
    sliders.forEach(slider => {
        slider.value = 90;
        document.getElementById(`angulo${slider.id.charAt(0).toUpperCase() + slider.id.slice(1)}`).textContent = 90;
    });
    fetchGet();
});

buttonForward.addEventListener('mousedown', function() {
    directionValue.forward = 1;
    fetchGet();
});
buttonForward.addEventListener('mouseup', function() {
    directionValue.forward = 0;
    fetchGet();
});

buttonBackward.addEventListener('mousedown', function() {
    directionValue.backward = 1;
    fetchGet();
});
buttonBackward.addEventListener('mouseup', function() {
    directionValue.backward = 0;
    fetchGet();
});

buttonLeft.addEventListener('mousedown', function() {
    directionValue.left = 1;
    fetchGet();
});
buttonLeft.addEventListener('mouseup', function() {
    directionValue.left = 0;
    fetchGet();
});

buttonRight.addEventListener('mousedown', function() {
    directionValue.right = 1;
    fetchGet();
});
buttonRight.addEventListener('mouseup', function() {
    directionValue.right = 0;
    fetchGet();
});

document.addEventListener('keydown', function(event) {
    switch(event.key) {
        case 'w':
        case 'W':
            directionValue.forward = 1;
            fetchGet();
            break;
        case 's':
        case 'S':
            directionValue.backward = 1;
            fetchGet();
            break;
        case 'a':
        case 'A':
            directionValue.left = 1;
            fetchGet();
            break;
        case 'd':
        case 'D':
            directionValue.right = 1;
            fetchGet();
            break;
```

```
    }
});

document.addEventListener('keyup',
    function(event) {
    switch(event.key) {
        case 'w':
        case 'W':
            directionValue.forward = 0;
            fetchGet();
            break;
        case 's':
        case 'S':
            directionValue.backward = 0;
            fetchGet();
            break;
        case 'a':
        case 'A':
            directionValue.left = 0;
            fetchGet();
            break;
        case 'd':
        case 'D':
            directionValue.right = 0;
            fetchGet();
            break;
    }
});
```

Django: Este framework de Python será la herramienta usada para crear el backend, en este lenguaje se escribirá el código que organiza los datos de operación del robot en categorías como sesiones de trabajo, objetos recogidos, peso y eventos como errores y/o excepciones. A la vez que se diseña una interfaz de administración de la aplicación, lo que permitirá un fácil manejo tanto de los datos obtenidos como de la información de los usuarios que operen el telerobot.

SQLite: SQLite es un sistema de gestión de bases de datos relacionales, que se utiliza para almacenar la información de una aplicación. SQLite es un sistema de gestión de bases de datos ligero, rápido y fácil de usar. Se caracteriza por su diseño sin servidor, lo que significa que no requiere una configuración o administración compleja. Además, es completamente autocontenido, lo que facilita su implementación en aplicaciones móviles y embebidas.

ArduinoIDE: ArduinoIDE es un entorno de desarrollo integrado, que se utiliza para programar las placas de Arduino. ArduinoIDE nos permite escribir, compilar y cargar programas en las placas de Arduino, mediante el uso de un lenguaje de programación basado en C/C++. Proporciona una interfaz amigable que simplifica el desarrollo de proyectos con Arduino, permitiendo a los usuarios centrarse en la lógica de sus proyectos en lugar de en detalles técnicos complejos.

CÓDIGO C++

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <WiFi.h>
#include <ESPAsyncWebServer.h>

// Definir los l mites del servo
#define SERVOMIN  150 // Min pulso
#define SERVOMAX  600 // Max pulso

// Definici n de la clase Articulacion
class Articulacion {
private:
    Adafruit_PWMServoDriver* pwm;
    int canal;
    int posInicial;
    int posNueva;
    int velocidad;
    bool enMovimiento;
    unsigned long tiempoInicio;
    int paso;
    int destino;

public:
    Articulacion(Adafruit_PWMServoDriver*
        _pwm, int _canal, int
        _posInicial = 0, int _velocidad =
        20) {
        pwm = _pwm;
        canal = _canal;
        posInicial = _posInicial;
        posNueva = _posInicial;
        velocidad = _velocidad;
        enMovimiento = false;
    }

    void inicializar() {
        mover(posInicial);
    }

    void mover(int angulo) {
        if (angulo < 0) angulo = 0;
        if (angulo > 180) angulo = 180;
        posNueva = angulo;
        if (posInicial != posNueva && !
            enMovimiento) {
            enMovimiento = true;
            paso = (posNueva > posInicial
                ) ? 1 : -1;
            destino = posNueva;
            tiempoInicio = millis();
        }
    }

    void detener() {
        enMovimiento = false;
    }

    void actualizar() {
        if (enMovimiento) {
            unsigned long tiempoActual =
                millis();
            unsigned long
                tiempoTranscurrido =
                tiempoActual -
                tiempoInicio;

            if (tiempoTranscurrido >=
                velocidad) {
                posInicial += paso;
                if ((paso > 0 &&
                    posInicial >= destino
                    ) || (paso < 0 &&
                    posInicial <= destino
```

```
                    )) {
61                      posInicial = destino;
62                      enMovimiento = false;
63                  }
64                  int pulse = map(
                        posInicial, 0, 180,
                        SERVOMIN, SERVOMAX);
65                  pwm->setPWM(canal, 0,
                        pulse);
66                  tiempoInicio =
                        tiempoActual;
67              }
68          }
69      }
70  };
71
72  // Configuraci n de pines del motor paso
        a paso con A4988
73  #define STEP_PIN 12    // Pin STEP del
        A4988
74  #define DIR_PIN 13     // Pin DIR del
        A4988
75  #define ENABLE_PIN 14 // Pin ENABLE del
        A4988 (opcional)
76
77  // Configuraci n de pines del segundo
        motor paso a paso con A4988
78  #define STEP_PIN2 27  // Pin STEP del
        segundo A4988
79  #define DIR_PIN2 26    // Pin DIR del
        segundo A4988
80
81  // Variables de control del motor
82  int stepDelay = 1000; // Tiempo de espera
        entre pasos (en microsegundos)
83  int stepCount = 0;    // Contador de pasos
84
85  // Configuraci n de red WiFi
86  const char* ssid = "Marco Omar";
87  const char* password = "1720500139M";
88
89  AsyncWebServer server(80);
90
91  // Crear objeto para el controlador
        PCA9685
92  Adafruit_PWMServoDriver pwm =
        Adafruit_PWMServoDriver();
93
94  // Instancia de la clase Articulacion
        para cada servo
95  Articulacion camera(&pwm, 0);
96  Articulacion base(&pwm, 1);
97  Articulacion hombro(&pwm, 2);
98  Articulacion codo(&pwm, 3);
99  Articulacion muneca(&pwm, 4);
100 Articulacion pinza(&pwm, 5);
101
102 bool forward = false;
103 bool backward = false;
104
105 void connectToWiFi() {
106   Serial.print("Connecting to WiFi");
107   WiFi.begin(ssid, password);
108   int attempts = 0;
109   while (WiFi.status() != WL_CONNECTED &&
          attempts < 20) {
110     delay(500);
111     Serial.print(".");
112     attempts++;
113   }
114   if (WiFi.status() == WL_CONNECTED) {
115     Serial.println("\nConnected to WiFi")
          ;
116     Serial.println(WiFi.localIP());
117   } else {
118     Serial.println("\nFailed to connect
          to WiFi");
119     // Intentar reconectar despu s de un
          tiempo
120     delay(10000);
121     ESP.restart(); // Reinicia el ESP32
122   }
123 }
124
125 void setup() {
126   Serial.begin(115200);
127
128   // Intentar conectar a WiFi
129   connectToWiFi();
130
131   // Inicializaci n de PCA9685
132   pwm.begin();
133   pwm.setPWMFreq(60);  // Frecuencia para
          servos
134
135   // Inicializaci n de servos
136   camera.inicializar();
137   base.inicializar();
138   hombro.inicializar();
139   codo.inicializar();
140   muneca.inicializar();
141   pinza.inicializar();
142
143   // Configuraci n de pines del motor
          paso a paso con A4988
144   pinMode(STEP_PIN, OUTPUT);
145   pinMode(DIR_PIN, OUTPUT);
146   pinMode(ENABLE_PIN, OUTPUT); // Si usas
          ENABLE_PIN
147
148   // Configuraci n de pines del segundo
          motor paso a paso
149   pinMode(STEP_PIN2, OUTPUT);
150   pinMode(DIR_PIN2, OUTPUT);
151
152   // Habilitar los controladores de los
          motores
153   digitalWrite(ENABLE_PIN, LOW); // O
          HIGH si tu configuraci n lo
          requiere
154
155   // Inicializa los pines en LOW
156   digitalWrite(STEP_PIN, LOW);
157   digitalWrite(DIR_PIN, LOW);
158   digitalWrite(STEP_PIN2, LOW);
159   digitalWrite(DIR_PIN2, LOW);
160
161   // Configuraci n del servidor web
162   server.on("/", HTTP_GET, [](
          AsyncWebServerRequest *request){
163     if (request->hasParam("camera")
164     && request->hasParam("base")
165     && request->hasParam("hombro")
166     && request->hasParam("codo")
```

```
    && request->hasParam("muneca")
    && request->hasParam("pinza")
    && request->hasParam("forward")
    && request->hasParam("backward")) {

      int cameraVal = request->getParam("camera")->value().toInt();
      int baseVal = request->getParam("base")->value().toInt();
      int hombroVal = request->getParam("hombro")->value().toInt();
      int codoVal = request->getParam("codo")->value().toInt();
      int munecaVal = request->getParam("muneca")->value().toInt();
      int pinzaVal = request->getParam("pinza")->value().toInt();
      forward = request->getParam("forward")->value().toInt();
      backward = request->getParam("backward")->value().toInt();

      // Imprimir valores para depuraci n
      Serial.printf("Camera: %d, Base: %d, Hombro: %d, Codo: %d, Muneca: %d, Pinza: %d\n",
                    cameraVal, baseVal, hombroVal, codoVal, munecaVal, pinzaVal);

      // Mover servos
      camera.mover(cameraVal);
      base.mover(baseVal);
      hombro.mover(hombroVal);
      codo.mover(codoVal);
      muneca.mover(munecaVal);
      pinza.mover(pinzaVal);

      request->send(200, "text/plain", "Servomotores movidos");
    } else {
      request->send(400, "text/plain", "Faltan par metros");
    }
  });

  server.begin();
}

int lastStepCount = 0;

void loop() {
  // Actualizar estado de los servos
  camera.actualizar();
  base.actualizar();
  hombro.actualizar();
  codo.actualizar();
  muneca.actualizar();
  pinza.actualizar();

  // Control del motor paso a paso
  if (forward) {
    digitalWrite(DIR_PIN, HIGH);
    digitalWrite(STEP_PIN, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(STEP_PIN, LOW);
    delayMicroseconds(stepDelay);
    stepCount++;
  } else if (backward) {
    digitalWrite(DIR_PIN, LOW);
    digitalWrite(STEP_PIN, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(STEP_PIN, LOW);
    delayMicroseconds(stepDelay);
    stepCount--;
  }

  // Imprimir el n mero de pasos cada 1000 pasos
  if (stepCount % 1000 == 0 && stepCount != lastStepCount) {
    Serial.print("Pasos realizados: ");
    Serial.println(stepCount);
    lastStepCount = stepCount;
  }
}
```

VI. Cálculo de posición

Beneficios de Cálculos matemáticos: Es verdad que puede llegar verse complejo captar el movimiento solo con cálculos sin uso de un sensor físico, pero a decir verdad se lo puede implementar en nuestro brazo lo que se necesitaba para poder realizar esta implementación es entender la lógica de cómo funcionan los componentes además de aplicar matemática analítica para poder realizar estos cálculos: Nuestro equipo tiene pensado mostrar en un plano cartesiano teniendo dos puntos los cuales llamaremos (x, y) donde aplicamos:

$$x = x_{\text{actual}} + \text{distancia} \cdot \sin(\text{ángulo});$$

$$y = y_{\text{actual}} + \text{distancia} \cdot \cos(\text{ángulo});$$

Donde x serían los puntos en el plano cartesiano y se irán actualizando conforme avance nuestro brazo robótico multiplicando con el ángulo que pongamos para mover nuestros servomotores. Como sabemos la fórmula de velocidad es de $V = \frac{d}{t}$ donde d es distancia y t es tiempo. Despejamos la distancia quedando $d = V \cdot t$ y sustituyendo en nuestra fórmula anterior nos quedaría:

$$x = x_{\text{actual}} + (V \cdot t) \cdot \sin(\text{ángulo});$$

$$y = y_{\text{actual}} + (V \cdot t) \cdot \cos(\text{ángulo});$$

Siendo así como se movería nuestro brazo robótico para ubicarlo en diferentes posiciones en el espacio cartesiano.

VII. Componentes del Sistema Mecánico y Electrónico

El sistema mecánico y electrónico del brazo robot incluye una variedad de componentes esenciales los cuales mencionaremos a continuación:

-A. *Impresora BIQU Thunder*

La impresora 3D BIQU Thunder se utilizó para fabricar las piezas del brazo robótico. A continuación se presentan sus características principales. Figura 1:

- **Área de Impresión**: 300 x 300 x 400 mm.
- **Velocidad de Impresión**: Hasta 100 mm/s.
- **Precisión de Impresión**: 0.1 mm.
- **Diámetro del Nozzle**: 0.4 mm (intercambiable).
- **Materiales Compatibles**: PLA, ABS, PETG, TPU, entre otros.
- **Sistema de Nivelación**: Nivelación automática de la cama.
- **Interfaz de Usuario**: Pantalla táctil a color.
- **Conectividad**: USB, Tarjeta SD.

Figura 1. Impresora 3D BIQU Thunder.

-B. *Material para Impresión 3D*

El brazo robot se fabrica utilizando un material biodegradable llamado PLA para la impresión 3D, lo que proporciona una estructura robusta y ligera además de amigable con el ambiente, en la Figura 2 podemos observar este filamento.

Figura 2. Ácido Poliláctico (PLA).

-C. *Control de Servomotores con PCA9685*

Basándonos en el datasheet del PCA9685, este controlador ofrece las siguientes características. Figura 3:

- 16 canales de PWM de resolución de 12 bits.
- Frecuencia de PWM ajustable hasta 1.6 kHz.
- Comunicación a través de la interfaz I2C.
- Voltaje de operación de 2.3V a 5.5V.
- Protección contra sobrecorriente y sobretemperatura.

Uso del PCA9685 para Controlar Servomotores: Para utilizar el PCA9685 para controlar servomotores, se debe configurar cada canal PWM para generar la señal de control adecuada. Un ejemplo de uso sería:

- Conectar el PCA9685 a un microcontrolador compatible con I2C.
- Configurar la frecuencia del PWM en el rango adecuado para servomotores (típicamente 50 Hz).
- Ajustar el ciclo de trabajo de los canales PWM para controlar la posición de los servomotores.
- Enviar comandos a través de la interfaz I2C para cambiar las posiciones de los servomotores según sea necesario.

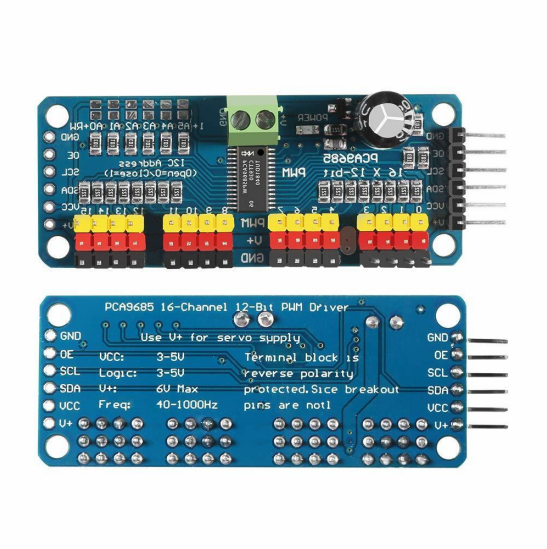

Figura 3. PCA9685.

-D. Servomotores

Los servomotores utilizados en el brazo robot son:

MG995: El MG995 es un servomotor de alta torsión adecuado para aplicaciones de robótica y control remoto se lo puede observar en la Figura 4. Sus principales características son:

- **Voltaje de operación**: 4.8-7.2V
- **Torsión**: 10 kg/cm a 6V
- **Velocidad**: 0.20 s/60° a 6V
- **Peso**: 55g
- **Ángulo de rotación**: 180°
- **Dimensiones**: 40.7 x 19.7 x 42.9 mm

Figura 4. Servomotor MG995.

SPT5435: El SPT5435 es un servomotor digital de alto rendimiento, diseñado para aplicaciones que requieren gran fuerza y precisión se lo puede observar en la Figura 5. Sus especificaciones incluyen:

- **Voltaje de operación**: 4.8-8.4V
- **Torsión**: 35 kg/cm a 7.4V
- **Velocidad**: 0.14 s/60° a 7.4V
- **Peso**: 80g
- **Ángulo de rotación**: 180°
- **Dimensiones**: 40.7 x 20.5 x 39.5 mm

Figura 5. Servomotor SPT5435.

SG90: El SG90 es un micro servo de 9g, ampliamente utilizado en proyectos de robótica y aeromodelismo se lo puede observar en la Figura 6. Sus características son:

- **Voltaje de operación**: 4.8-6V
- **Torsión**: 1.8 kg/cm a 6V
- **Velocidad**: 0.12 s/60° a 6V
- **Peso**: 9g
- **Ángulo de rotación**: 180°
- **Dimensiones**: 23 x 12.2 x 29 mm

Figura 6. Servomotor SG90.

DS3230: El DS3230 es un servomotor de alta precisión, ideal para aplicaciones que requieren movimientos suaves y precisos se lo puede observar en la Figura 7. Sus especificaciones son:

- **Voltaje de operación**: 4.8-6V
- **Torsión**: 30 kg/cm a 6V
- **Velocidad**: 0.16 s/60° a 6V

- **Peso**: 65g
- **Ángulo de rotación**: 180°
- **Dimensiones**: 40 x 20 x 40.5 mm

Figura 7. Servomotor DS3230.

-E. Módulo HX711 y Sensor de Peso de 5kg

El módulo HX711 es un conversor analógico a digital (ADC) utilizado para leer datos de sensores de peso. El sensor de peso de 5kg se utiliza para medir la fuerza o el peso aplicado. Figuras 8 y 9.

- **Voltaje de operación (HX711)**: 2.6-5.5V
- **Rango de medida (Sensor de 5kg)**: Hasta 5kg
- **Precisión**: 24 bits
- **Interfaz**: Serial

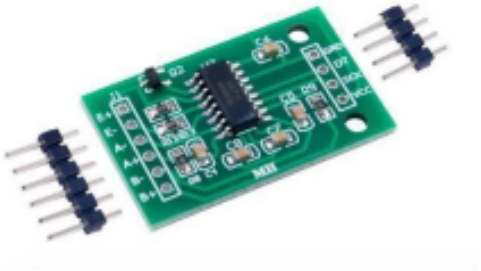

Figura 8. Modulo HX711.

Figura 9. Sensor de peso 5KG.

-F. Drivers A4988 para Motor Paso a Paso OUKEDA OK42STH47-204A1XH

El A4988 es un controlador de motor paso a paso que permite controlar la velocidad y el torque del motor paso a paso OUKEDA OK42STH47-204A1XH. Figura 10.

- **Voltaje de operación (A4988)**: 8-35V
- **Corriente de salida**: Hasta 2A por fase
- **Modo de pasos**: Completo, 1/2, 1/4, 1/8 y 1/16
- **Motor Paso a Paso OUKEDA OK42STH47-204A1XH**:
  - **Corriente por fase**: 2A
  - **Torsión**: 0.59 Nm
  - **Ángulo de paso**: 1.8°
  - **Dimensiones**: 42 x 42 x 47 mm

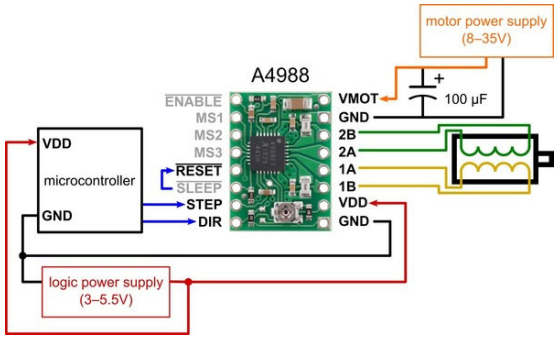

Figura 10. Driver A4988.

-F1. Motor Paso a Paso OUKEDA OK42STH47-204A1XH: Es un tipo de motor eléctrico que se utiliza para aplicaciones que requieren control preciso de posición y velocidad. Este motor es conocido por su alta precisión y capacidad de mantener una posición fija cuando no está en movimiento. Figura 11.

- **Corriente por fase**: 2A
- **Torsión**: 0.59 Nm
- **Ángulo de paso**: 1.8°
- **Voltaje**: 3.96V

- **Resistencia por fase**: 1.98Ω
- **Inductancia por fase**: 4.5mH
- **Dimensiones**: 42 x 42 x 47 mm
- **Peso**: 0.4 kg
- **Diámetro del eje**: 5mm
- **Longitud del eje**: 24mm

- **Voltaje de operación**: 2.3-5.5V
- **Frecuencia PWM**: 40-1000Hz
- **Canales PWM**: 16
- **Interfaz**: I2C

Figura 11. Motor Paso a Paso OUKEDA OK42STH47-204A1XH.

-F2. ESP-WROOM-32: El ESP-WROOM-32 es un módulo de microcontrolador con capacidades Wi-Fi y Bluetooth. Figura 12.

- **Procesador**: Tensilica Xtensa LX6 dual-core
- **Velocidad de reloj**: Hasta 240 MHz
- **Memoria RAM**: 520 KB
- **Memoria Flash**: 4 MB
- **Interfaz**: Wi-Fi, Bluetooth, GPIO, UART, SPI, I2C
- **Voltaje de operación**: 3.3V

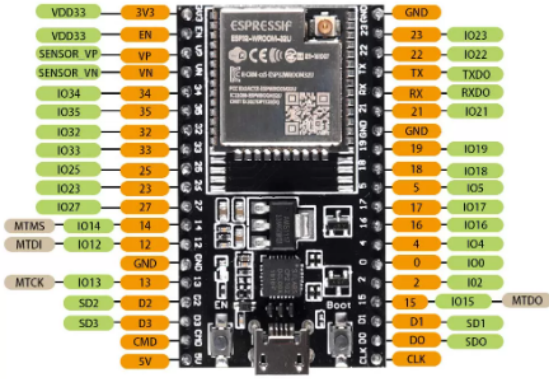

Figura 12. ESP-WROOM-32

-F3. ESP32-CAM: El ESP32-CAM es un módulo de cámara basado en el ESP32 con capacidades de procesamiento de imágenes. Figura 13.

- **Procesador**: Tensilica Xtensa LX6 dual-core
- **Velocidad de reloj**: Hasta 240 MHz
- **Memoria RAM**: 520 KB
- **Memoria Flash**: 4 MB
- **Cámara**: OV2640 2MP
- **Interfaz**: Wi-Fi, Bluetooth, GPIO, UART, SPI, I2C
- **Voltaje de operación**: 3.3V

Figura 13. ESP32-CAM

-F4. ESP32-CAM-MB: El ESP32-CAM-MB es una placa de desarrollo que integra el módulo ESP32-CAM con una interfaz USB para facilitar la programación y la comunicación. Figura 14

- **Procesador**: Tensilica Xtensa LX6 dual-core
- **Velocidad de reloj**: Hasta 240 MHz
- **Memoria RAM**: 520 KB
- **Memoria Flash**: 4 MB
- **Cámara**: OV2640 2MP

- **Interfaz**: USB, Wi-Fi, Bluetooth, GPIO, UART, SPI, I2C
- **Voltaje de operación**: 3.3V

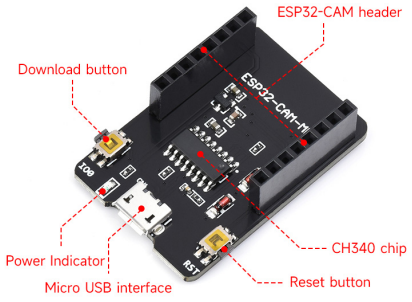

Figura 14. ESP32-CAM-MB

VIII. Agradecimientos

Los autores desean expresar su más sincero agradecimiento a todas las personas e instituciones que hicieron posible la realización de este proyecto. En primer lugar, agradecemos a la Universidad Nacional de Loja por presentarnos este proyecto para mejorar nuestros conocimientos y habilidades dentro de nuestra carrera y brindarnos el apoyo en el desarrollo del proyecto.

- Alexis David Jara Armijos, por su aporte a que el sistema se pueda adaptar a cualquier dispositivo.
- César Stif López Cevallos, por sus aportes en las gráficas de los datos para el robot.
- César Daniel Ramos Merchán, por su aporte en estilos de la pagina web usando CSS y ensamblador.
- David Gustavo Toledo Vega, por su aporte en el diseño e impresión 3D y la parte electronica del robot.
- Marco Omar Chiliguano Buri, por su aporte de la conexión de hardware a software y la creación de la pagina web.

Finalmente, extendemos nuestro agradecimiento a nuestras familias y amigos por su paciencia y apoyo incondicional durante todo el proceso para el desarrollo e implementación del proyecto a lo largo del ciclo. Figura 15.

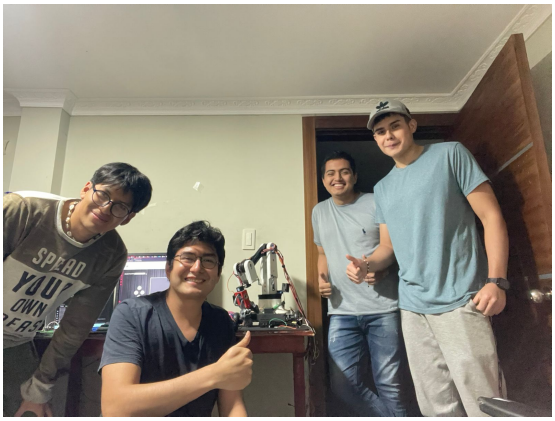

Figura 15. Grupo 4

IX. Conclusiones

El desarrollo de este proyecto permitirá a los estudiantes de la Universidad Nacional de Loja adquirir experiencia práctica en la programación y control de dispositivos electrónicos y mecánicos, así como en el diseño y construcción de aplicaciones web. La implementación de un brazo robótico controlado de forma remota proporcionará una herramienta útil para la recolección de desechos peligrosos, mejorando la seguridad del personal de la universidad y contribuyendo a la automatización de tareas repetitivas y riesgosas.

Referencias

[1] .Ebay listing: Stepper Motor OUKEDA OK42STH47-204A1XH,.ªvailable at: https://www.ebay.com/itm/284645887022.
[2] ”Sicnova3D: ¿Qué es el PLA en impresión 3D y para qué se utiliza?,.ªvailable at: https://sicnova3d.com/blog/experiencias-3d/que-es-el-pla-en-impresion-3d-y-para-que-se-utiliza/.
[3] ”The 3D Printer Store: BIQU Thunder 3D Printer,.ªvailable at: https://the3dprinterstore.com/products/biqu-thunder-3d-printer.
[4] ”Programar Fácil: ESP32-CAM,.ªvailable at: https://programarfacil.com/esp32/esp32-cam/.
[5] ”Sigma Electrónica: ESP32-WROOM-32D,.ªvailable at: https://www.sigmaelectronica.net/producto/esp32-wroom-32d/.
[6] ”Naylamp Mechatronics: Tutorial Módulo Controlador de Servos PCA9685 con Arduino,.ªvailable at: https://naylampmechatronics.com/blog/41$_{t}utorial - modulo - controlador - de - servos - pca9685 - con - arduino.html$.
[7] ”Diario Electrónico Hoy: Descripción del Driver A4988,.ªvailable at: https://www.diarioelectronicohoy.com/blog/descripcion-del-driver-a4988.
[8] ”Megatrónica: Sensor de Peso/Fuerza, Celda de Carga de 5kg,.ªvailable at: https://megatronica.cc/producto/sensor-de-peso-fuerza-celda-de-carga-de-5kg-2/.
[9] Ürany: Conoce el Funcionamiento de los Servomotores,.ªvailable at: https://urany.net/blog/conoce-el-funcionamiento-de-los-servomotores: :text=
[10] ”Digital House: HTML, CSS y JavaScript: ¿Para qué sirve cada lenguaje?,.ªvailable at: https://www.digitalhouse.com/blog/html-css-y-javascript-para-que-sirve-cada-lenguaje/.
[11] ”MDN Web Docs: Introduction to Django,.ªvailable at: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction.
[12] ÏONOS: SQLite,.ªvailable at: https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/sqlite/.
[13] .Aprendiendo Arduino: IDE,.ªvailable at: https://www.aprendiendoarduino.com/tag/ide/.