UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Ingeniería en computación

PIS

"Control Remoto de un Brazo Robótico a través de Software"

Segundo ciclo paralelo "A"

Tutor Encargado:

Ing. César Iñiguez

Integrantes:

José Francisco Riofrío Maldonado

Iván Alexander Fernández Cañar

Ariel Ismael González Astudillo

Jorge Luis Luzuriaga Betancourt

Richard Vicente Cajas Riofrío

Período:

Abril - Agosto

2024

Introducción

Antecedentes

De acuerdo al Reglamento de Higiene y Seguridad de la Universidad Nacional de Loja que determina las directrices para el manejo de productos químicos, los desechos de tipo biológico, el manejo de objetos cortantes y punzantes, y manipulación de cargas, es necesario propuestas de manejo seguro de estos elementos para evitar enfermedades ocupacionales, en el personal de la UNL. Debido a que en la Institución no cuenta con un sistema para la recolección de basura automatizada que permita el manejo seguro de los residuos contaminantes o tóxicos antes mencionados, se nos ha planteado a los estudiantes de la carrera de ingeniería en computación, plantear posibles soluciones a esta problemática. Teniendo en cuenta que, la propuesta debe tener como prioridad el contexto de la sostenibilidad y la protección del medio ambiente, la implementación de un robot recolector de basura contribuiría a la reducción de la contaminación y la promoción de entornos urbanos más limpios y saludables.

Objetivos

Objetivo General

- Desarrollar e implementar un sistema de control remoto mediante software y hardware para un brazo robótico, con el fin de facilitar la manipulación precisa y segura de objetos a distancia, promoviendo la automatización y la eficiencia en tareas operativas.

Objetivos Específicos

- Diseñar e implementar una interfaz de control de software intuitivo y eficiente que permita a los usuarios operar el brazo robótico de forma remota, garantizando la precisión y la seguridad en la manipulación de objetos.
- Integrar sistemas de hardware de retroalimentación en tiempo real para proporcionar información detallada sobre el objeto, su posición y otros parámetros relevantes durante la operación remota del brazo robótico, con el fin de mejorar la percepción y el control del usuario.
- Validar el sistema de control remoto mediante pruebas de funcionamiento del prototipo, evaluando su desempeño en escenarios de prueba y garantizando su fiabilidad y eficacia en entornos de trabajo a distancia.

Propuesta

Teniendo como antecedentes el proyecto realizado en el ciclo anterior (brazo robótico semiautomático), proponemos darle continuidad al mismo implementando mejoras al diseño del brazo robótico para convertirlo en su mayor parte, en un brazo autónomo, hablamos de asignarle funciones de manejo de desechos inorgánicos dañinos para el personal de limpieza, mediante la aplicación web, misma que permitirá a los usuarios controlar los movimientos del brazo, obteniendo datos en tiempo real para una mejor precisión y eficiencia al momento de toparse con obstáculos, además de visualizar dentro del software los movimientos realizados por el robot, con la ayuda de un ESP32 Cam, que transmite la señal mediante wifi y bluetooth.

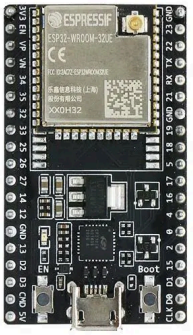
MATERIALES

Arduino UNO

Es una placa de desarrollo de microcontroladores basada en el ATmega328P, muy utilizada en proyectos realizados por principiantes, por su facilidad de manejo de datos, que nos ofrece 14 pines digitales de entrada/salida, 6 entradas analógicas, un conector USB para programación y alimentación, un conector de alimentación, un botón de reinicio y un cristal de 16 MHz

ESP32

Microcontrolador que a diferencia de Arduino, incluye Wifi y Bluetooth. Además incluye otros beneficios como su facilidad para conectarse al módulo de cámara OV2640 o su compatibilidad con el Arduino Ide, software el cual ya se utilizó para la programación del brazo.



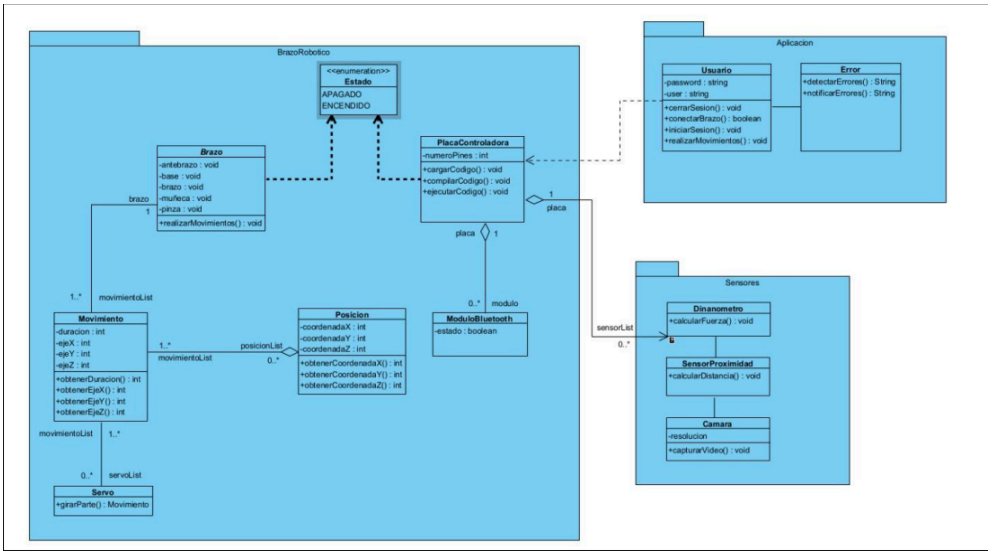
Cámara OV2640

Mini módulo de cámara compacta de 2 MP que permitirá ver el entorno del brazo cuando se encuentre alejado del usuario, la cual presentará imágenes digitales con alta resolución y baja potencia, además fácil de usar gracias a su flexibilidad con ESP32. Presenta una resolución de 1600x1200 y su tamaño de píxel es de 2.2 x 2.2 micrómetros.

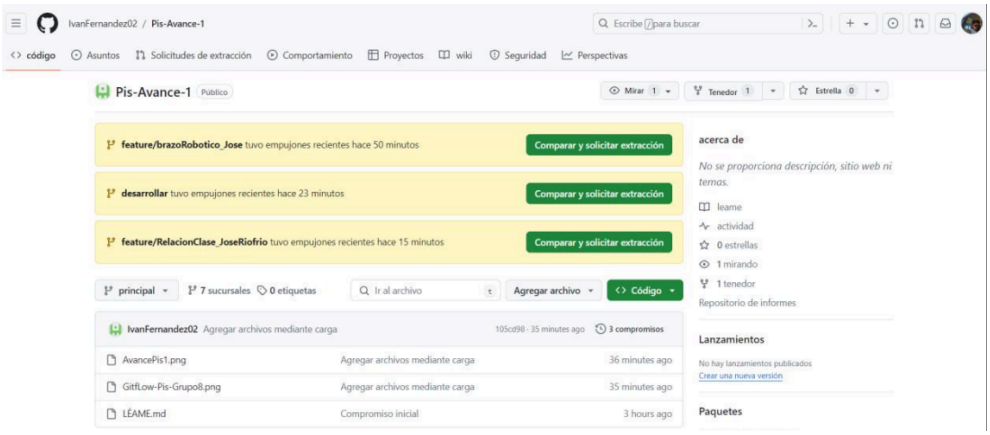
Metodología

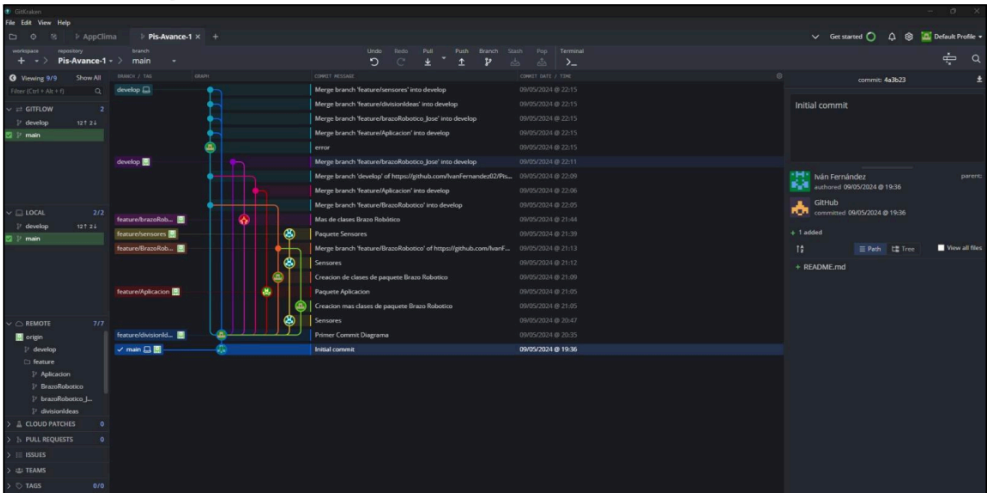
El proyecto que se pretende llevar a cabo por parte de nuestro equipo consiste en la elaboración de un software para el manejo del brazo robótico cuya función sea la recolección de basura y visualización de los movimientos realizados.

Primeramente se realizó el diagrama UML para darnos una idea de la dirección que debía tomar nuestro proyecto, quedándonos de la siguiente manera:

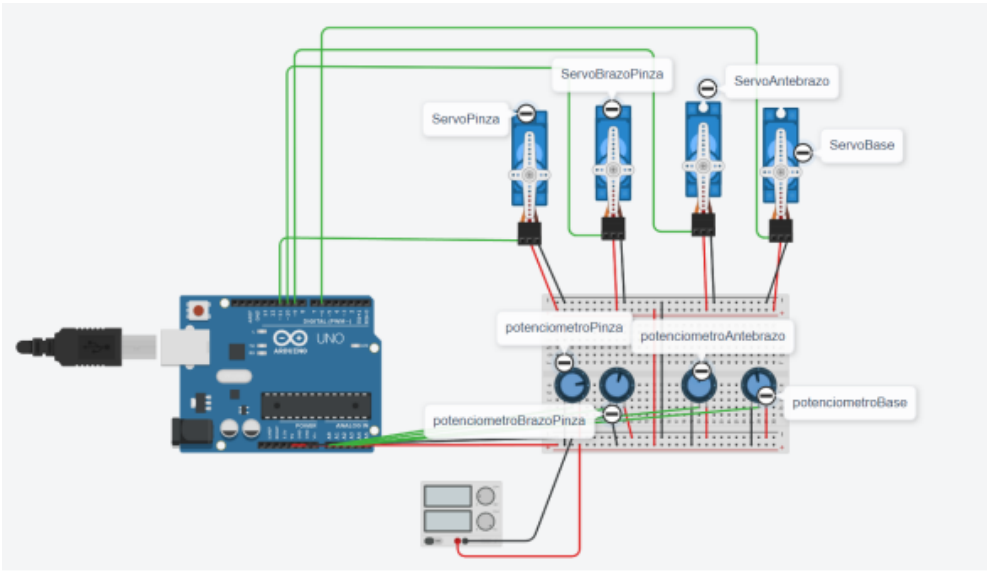


Posteriormente creamos nuestro repositorio en Git Flow y Git Kraken:





Además de ello, realizamos un diseño de las conexiones que se realizaron para la conexión de los servos con la placa Arduino Uno:



Link de Tinkercad: :
https://www.tinkercad.com/things/9YsqC1ymD0v-brazoroboticogrupo8

Posterior a eso, empezamos a darle vida al proyecto, creando la aplicación web y agregando nuevas funcionalidades al brazo robótico:

El código utilizado en arduino es el siguiente:

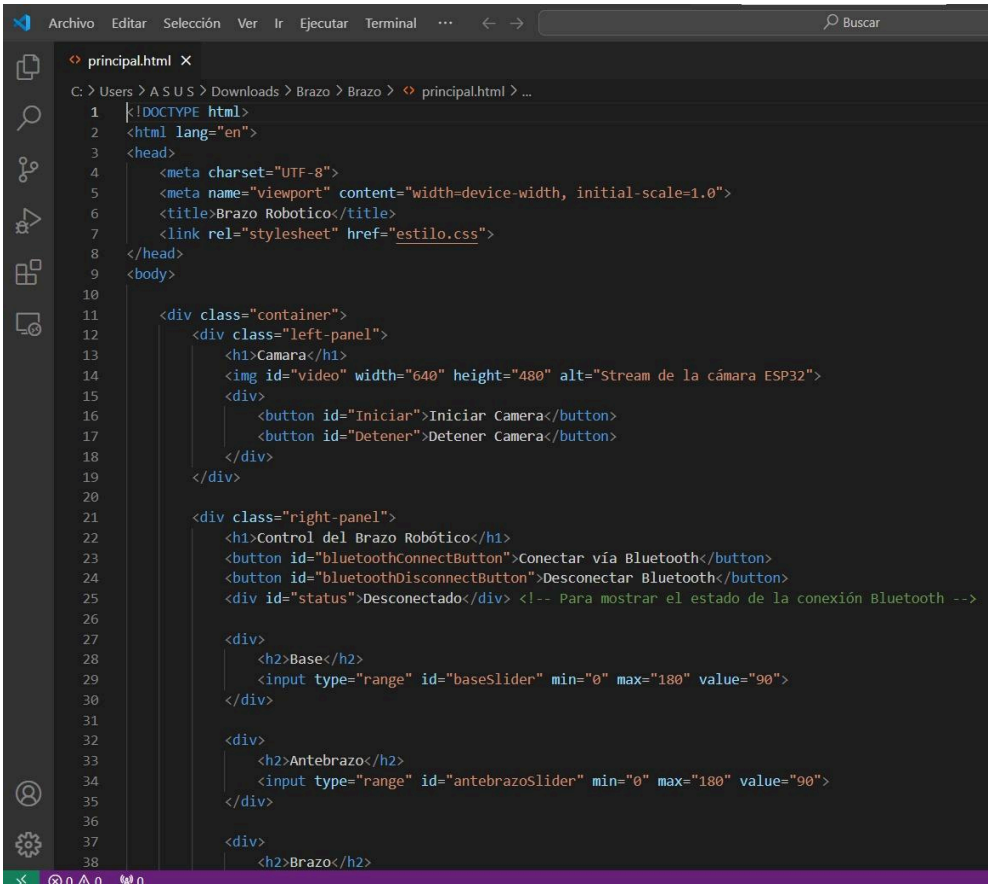
```
#include <Servo.h>
#include <SoftwareSerial.h>

Servo servo1; // base
Servo servo2; // brazo
Servo servo3; // antebrazo
Servo servo4; // pinza

SoftwareSerial miBT(6, 7); // rx, tx

int pinservo1 = 8;
int pinservo2 = 12;
int pinservo3 = 10;
int pinservo4 = 11;

char car; // caracter recibido vía BT
String incode = ""; // código recibido
boolean encode = false; // fin del recepción

int num1 = 0; // centena 100 grados
int num2 = 0; // decena 10-90 grados
int num3 = 0; // unidad 0-9 grados
int num = 0; // numero recibido 0-180 grados: num1+num2+num3

void setup() {
  servo1.attach(pinservo1);
  servo2.attach(pinservo2);
  servo3.attach(pinservo3);
  servo4.attach(pinservo4);

  miBT.begin(9600);
  delay(2000);

  servo1.write(90);
  delay(450);
  servo2.write(120);
  delay(450);
  servo3.write(180);
  delay(450);
  servo4.write(0);
  delay(450);
}

void loop() {
  if (miBT.available()) {
    car = miBT.read();
    if (car == 'x') { // x marca el fin del numero enviado via BT
      encode = true;
    } else {
      incode += car;
    }
  }

  if (encode) {
    int longitud = incode.length();
    char sel = incode.charAt(0); // a, b, c ó d
    num1 = incode.charAt(1) - '0'; // centena de los grados
    num2 = (longitud > 2) ? incode.charAt(2) - '0' : 0; // decena de los grados
    num3 = (longitud > 3) ? incode.charAt(3) - '0' : 0; // unidad de los grados
```

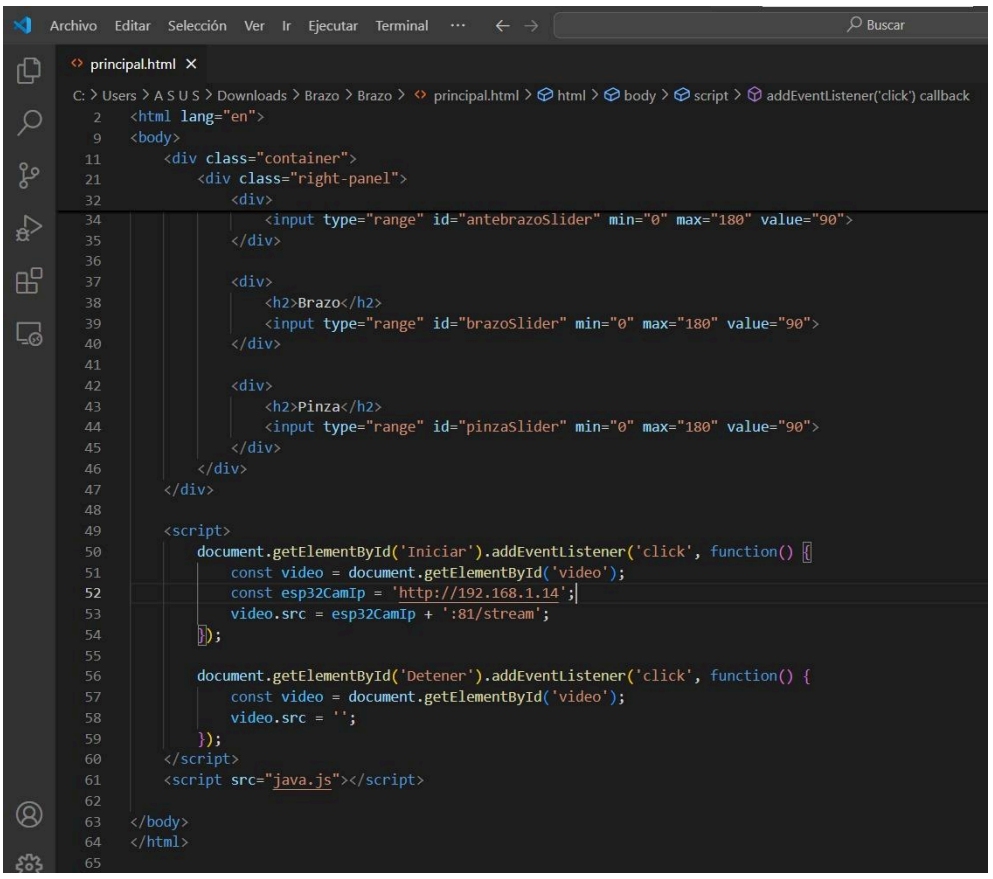
```
    if (longitud == 4) { // p.e. a180
      num = num1 * 100 + num2 * 10 + num3;
    } else if (longitud == 3) { // p.e. c94
      num = num1 * 10 + num2;
    } else if (longitud == 2) { // p.e. d5
      num = num1;
    }

    switch (sel) { // incode.charAt(0); // a, b, c ó d
      case 'a':
        servo4.write(num); // mueva servo 4 los grados recibidos
        break;
      case 'b':
        servo3.write(num); // mueva servo 3 los grados recibidos
        break;
      case 'c':
        servo2.write(num); // mueva servo 2 los grados recibidos
        break;
      case 'd':
        servo1.write(num); // mueva servo 1 los grados recibidos
        break;
    }

    // Reset de variables
    car = '\0';
    incode = "";
    encode = false;
  }
}
```

En el cual se utilizó la librería Servo.h para el manejo de los servomotores con los que dispone el robot, y la librería SoftwareSerial.h para la comunicación serial de los pines digitales del Arduino, a excepción de los pines de hardware serial (0 y 1), para establecer la comunicación Bluetooth, permitiendo que el Arduino reciba comandos inalámbricamente para controlar el brazo robótico.

En la función setup, inicializamos los servos y configuramos su posición inicial, además inicializamos la comunicación vía bluetooth.

Para la creación de la aplicación Web se utilizó el siguiente código:





Para la cámara y para la obtención del ip utilizamos este código:

lacamaritafinal.ino | app_httpd.cpp | camera_index.h | camera_pins.h | ci.json

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//
//            You must select partition scheme from the board menu that has at least 3MB APP space.
//            Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes up from 15
//            seconds to process single frame. Face Detection is ENABLED if PSRAM is enabled as well

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE  // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_CAMS3_UNIT  // Has PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
//#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
```

En esta parte incluimos las bibliotecas que utilizaremos y además definimos el modelo de la cámara utilizada:

- **esp_camera.h:** Para el manejo de la camara ESP32.
- **WiFi.h:** Para la conexión wifi.

lacamaritafinal.ino | app_httpd.cpp | camera_index.h | camera_pins.h | ci.json

```
//#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char *ssid = "NettplusJames";
const char *password = "jamesalam7";

void startCameraServer();
void setupLedFlash(int pin);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
```

lacamaritafinal.ino | app_httpd.cpp | camera_index.h | camera_pins.h | ci.json

```
53    config.pin_d0 = Y2_GPIO_NUM;
54    config.pin_d1 = Y3_GPIO_NUM;
55    config.pin_d2 = Y4_GPIO_NUM;
56    config.pin_d3 = Y5_GPIO_NUM;
57    config.pin_d4 = Y6_GPIO_NUM;
58    config.pin_d5 = Y7_GPIO_NUM;
59    config.pin_d6 = Y8_GPIO_NUM;
60    config.pin_d7 = Y9_GPIO_NUM;
61    config.pin_xclk = XCLK_GPIO_NUM;
62    config.pin_pclk = PCLK_GPIO_NUM;
63    config.pin_vsync = VSYNC_GPIO_NUM;
64    config.pin_href = HREF_GPIO_NUM;
65    config.pin_sccb_sda = SIOD_GPIO_NUM;
66    config.pin_sccb_scl = SIOC_GPIO_NUM;
67    config.pin_pwdn = PWDN_GPIO_NUM;
68    config.pin_reset = RESET_GPIO_NUM;
69    config.xclk_freq_hz = 20000000;
70    config.frame_size = FRAMESIZE_UXGA;
71    config.pixel_format = PIXFORMAT_JPEG;  // for streaming
72    //config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
73    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
74    config.fb_location = CAMERA_FB_IN_PSRAM;
75    config.jpeg_quality = 12;
76    config.fb_count = 1;
77
78    // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
79    //                      for larger pre-allocated frame buffer.
```

lacamaritafinal.ino | app_httpd.cpp | camera_index.h | camera_pins.h | ci.json

```
80    if (config.pixel_format == PIXFORMAT_JPEG) {
81      if (psramFound()) {
82        config.jpeg_quality = 10;
83        config.fb_count = 2;
84        config.grab_mode = CAMERA_GRAB_LATEST;
85      } else {
86        // Limit the frame size when PSRAM is not available
87        config.frame_size = FRAMESIZE_SVGA;
88        config.fb_location = CAMERA_FB_IN_DRAM;
89      }
90    } else {
91      // Best option for face detection/recognition
92      config.frame_size = FRAMESIZE_240X240;
93  #if CONFIG_IDF_TARGET_ESP32S3
94      config.fb_count = 2;
95  #endif
96    }
97
98  #if defined(CAMERA_MODEL_ESP_EYE)
99    pinMode(13, INPUT_PULLUP);
100   pinMode(14, INPUT_PULLUP);
101 #endif
102
103   // camera init
104   esp_err_t err = esp_camera_init(&config);
105   if (err != ESP_OK) {
106     Serial.printf("Camera init failed with error 0x%x", err);
```

lacamaritafinal.ino | app_httpd.cpp | camera_index.h | camera_pins.h | ci.json

```
    return;
  }

  sensor_t *s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1);        // flip it back
    s->set_brightness(s, 1);   // up the brightness just a bit
    s->set_saturation(s, -2);  // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  if (config.pixel_format == PIXFORMAT_JPEG) {
    s->set_framesize(s, FRAMESIZE_QVGA);
  }

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
  s->set_vflip(s, 1);
#endif

// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
```

lacamaritafinal.ino | app_httpd.cpp | camera_index.h | camera_pins.h | ci.json

```
// Setup LED FLash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
  setupLedFlash(LED_GPIO_NUM);
#endif

  WiFi.begin(ssid, password);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Do nothing. Everything is done in another task by the web server
  delay(10000);
}
```

En la función *setup* tenemos varias configuraciones como:

- La inicialización de la comunicación serial.
- Configuración e inicialización de cámara.
- Conexión wifi.
- Inicio del servidor de cámara.
- Impresión de IP, entre otras.

Resultados

En este proyecto desarrollamos un brazo robótico mismo que es controlado mediante una aplicación web, elaborada por nosotros, que utiliza un microcontrolador ESP32CAM integrado en la pinza para obtener una mejor precisión en los datos recopilados durante la ejecución del proyecto. Hallamos que:

- La integración de la cámara nos permite mayor precisión al momento de agarrar los objetos.
- El manejo del brazo a través de la aplicación web, resulta muy intuitivo para los usuarios además de que su uso es sencillo.
- La combinación del ESP32CAM y el Arduino UNO, nos permitió un eficiencia en los componentes, facilitando su comunicación y control del brazo.

Conclusiones

- Logramos crear un software intuitivo y eficiente para el manejo del brazo robótico, mismo que presenta datos precisos de los movimientos realizados.
- Se integró al brazo una ESP32CAM misma que presenta el video de los movimientos realizados por el brazo, dentro de la aplicación web, permitiendo tener mayor precisión en el manejo de los objetos.
- Realizamos distintas pruebas de funcionamiento tanto del brazo, la cámara y la aplicación web, lo cuál nos permitió observar ciertos errores en su ejecución que fueron resueltos uno por uno, validando de esta manera, su correcto funcionamiento.

Bibliografía

[1] RoboTecs. ¿Cómo CONTROLAR BRAZO Robótico por Bluetooth? Arduino, MIT App Inventor (en sólo 8 minutos). (9 de octubre de 2022). Accedido el 28 de julio de 2024. [Video en línea]. Disponible: https://www.youtube.com/watch?v=JCw84Sp7uBI

[2] JuanDavid_Dev. Como crear un Login de usuarios Guía paso a paso. (3 de octubre de 2023). Accedido el 29 de julio de 2024. [Video en línea]. Disponible: https://www.youtube.com/watch?v=dbiS_7h1YRs

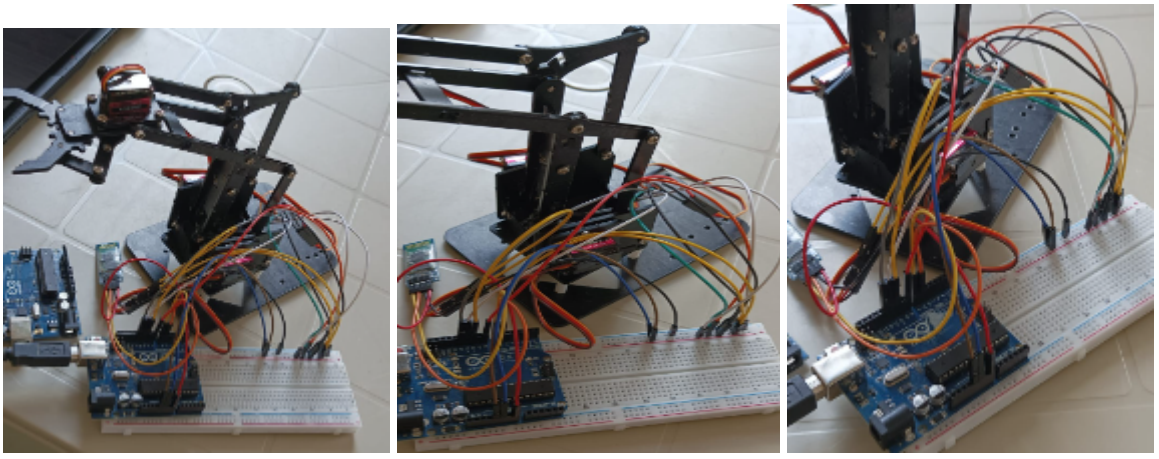
[3] AllDatasheet.Es. “SG90 Datasheet(PDF)”. ALLDATASHEET.ES - Sitio de Búsqueda de Datasheet, Sitio de Búsqueda de Datasheet de Componentes Electrónicos y Semiconductores y otros semiconductores. Accedido el 29 de julio de 2024. [En línea]. Disponible: https://www.alldatasheet.es/datasheet-pdf/pdf/1572383/ETC/SG90.html

[4] UnitElectronics. “HC-06 Bluetooth Esclavo”. UnitElectronics. Accedido el 28 de julio de 2024. [En línea]. Disponible: https://uelectronics.com/producto/hc-06-bluetooth-esclavo/

[5] Arsys. “phpMyAdmin: ¿qué es y cómo usarlo? | Blog de Arsys”. Arsys. Accedido el 29 de julio de 2024. [En línea]. Disponible: https://www.arsys.es/blog/phpmyadmin
[6] D. Herrero. “Introducción a sistemas embebidos con Arduino”. CePETel. Accedido el 28 de julio de 2024. [En línea]. Disponible: https://www.cepetel.org.ar/wp-content/uploads/2021/10/Introducción-a-Sistemas-Embebidos-con-Arduino-1era-Parte.pdf

Anexos

Hardware



Funcionamiento del brazo

Funcionamiento de la cámara ESP32 dentro de la aplicación