

Control de Brazo-robot con ESP32 CAM

El proyecto consiste en controlar un barzo robot con 3 servo motores mediante ESP32 CAM , que incluye lector de tarjetas SD Card, vía Internet de las Cosas (IoT). Utilizando el protocolo MQTT y el broker hivemq.

La información se ha obtenido de :

<https://community.appinventor.mit.edu/t/esp32-mqtt-broker-publish-subscribe-thingspeak/10490>

Materiales:

El kit del brazo y la tarjeta ESP32 CAM se puede comprar en aliexpress:
ESP32 CAM por menos de 5€

<https://es.aliexpress.com/item/4001133770461.html?spm=a2g0s.9042311.0.0.13ae63c0Dvqmh2o>

ESP32 y lector de tarjetas SD Card por menos de 10.

<https://es.aliexpress.com/item/4000152270368.html?spm=a2g0s.9042311.0.0.274263c0o6PsXX>
Brazo robot por 15€ o 27€ con servos

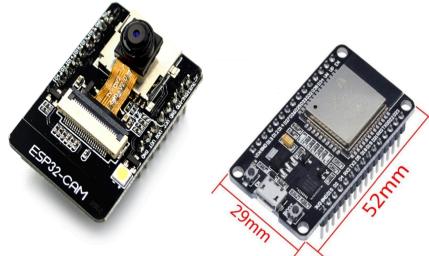
<https://es.aliexpress.com/item/32761263449.html?spm=a2g0s.9042311.0.0.274263c0NRhjbZ>

Placa control servos PCA9685 por unos 4€

<https://es.aliexpress.com/item/4000532870988.html?spm=a2g0o.cart.0.0.bbb73c00hK0myg&mp=1>

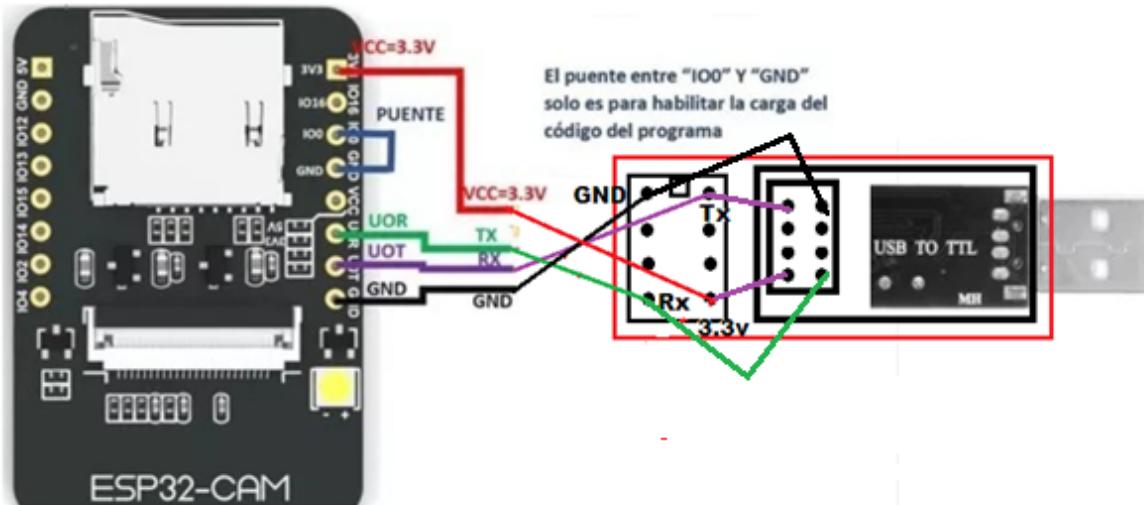
Alimentación 5V o 3,3v por menos de 2€

<https://es.aliexpress.com/item/4000689310993.html?spm=a2g0s.9042311.0.0.274263c0LFTCwf>
<https://es.aliexpress.com/item/32588261370.html?spm=a2g0s.9042311.0.0.274263c0LFTCwf>



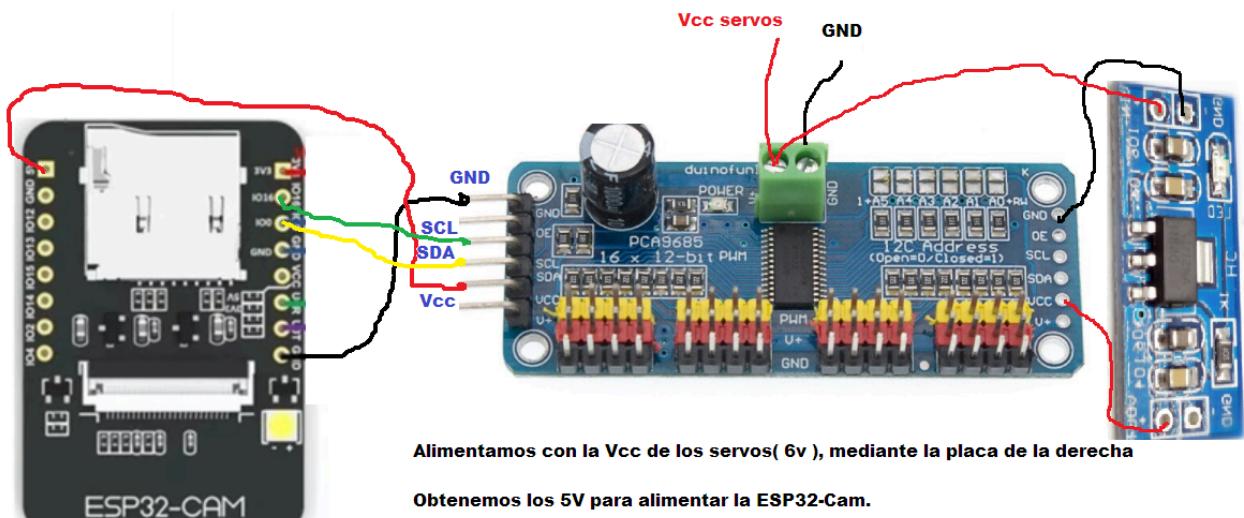
Conexiones:

Conexión para programarla placa ESP32:



Conexión de la placa programada:

Conectar ESP32 a placa de control servos: Base: 0 , Codo: 4, Pinza: 8
ESP32_CAM I2C: SDA=0, SCL= 16



Conexión con el broker.hivemq.com para publicar y suscribir:

- Conectaremos al **broker.hivemq.com** en el Port (1883) que es gratuito y no hace falta registrarse. También se pueden utilizar el broker **mqtt.eclipse.org** en el Port (1883).
- Para controlar el robot podemos utilizar la App gratuita de Google [Linear MQTT Dashboard](#).
- Se puede utilizar en el ordenador para comunicarnos la aplicación MQTTLens [Google Chrome Web Store](#)
- Necesitaremos la librería: [PubSubClient](#)

- Podemos crear nuestra aplicación con <http://ai2.appinventor.mit.edu/>, utilizamos la extensión MQTT: <http://ullisroboterseite.de/android-AI2-MQTT-de.html#publish> Al final esta el enlace al código fuente y apk compilada en el proyecto.
- Para suscribir : **jarp/servo**
- Para publicar: **jarp/posicion**

La información para configurar las aplicaciones y la información ampliada se pueden obtener en el enlace:

<https://community.appinventor.mit.edu/t/esp32-mqtt-broker-publish-subscribe-thingspeak/10490>

Funcionamiento:

El brazo-robot, se controla con la aplicación desarrollada que manda los siguientes comandos:

Comando		Comando	
1	Mueve Servo base + 5º	i,fM	Muestra Memoria desde i a f
2	Mueve Servo base - 5º	Memoria	Muestra Memoria
3	Mueve Servo codo + 5º	I	Servos a posición inicial
4	Mueve Servo codo - 5º	E	Ejecuta programa en EEPROM
5	Mueve Servo pinza + 5º	rG	Guarda posición servos r:retardo
6	Mueve Servo pinza - 5º	b,c,p,rA	Almacena la posición b,c,p y retardo
xb	Mueve Servo base a xº	nS	Salva EEPROM en SDCard como archivo n.txt
c	Mueve Servo codo a xº	nL	Carga EEPROM con el archivo n.txt desde la SDCard como
p	Mueve Servo pinza a xº	D	Muestra directorio de la SDCard
nR	Ver contenido archivo n.txt SDCard	b,c,cH	Mueve servos a b,c,p
d,vC	Cambia valor(v) de la dirección(d) de EEPROM	reset	Resetea la placa

Como podemos ver se puede:

- Mover los servos individualmente en +-5º.
- Mover cada servo con un Slider entre 0 y 180º.
- Mover los servos a su posición inicial (90º,90º,95º).
- Guardar la posición actual de los 3 servos más un retardo en s.
- Guardar una posición determinada por los valores de los Slider más un retardo en s.
- Abrir y cerrar la pinza.
- Ejecutar los movimientos almacenados en la EEPROM desde la dirección 4
- Modificar una posición de memoria de la EEPROM.

- Guardar los movimientos almacenados en la EEPROM en un archivo en la tarjeta SDCard.
- Cargar el contenido de un archivo de la tarjeta SDCard en la EEPROM.
- Mostrar los archivos guardados en la tarjeta SDCard.
- Ver el contenido de un archivo guardado en la tarjeta SDCard.
- Realizar un reset de la ESP32.
- Cualquier otra función que queramos realizar.

Cada movimiento se guardar en 4 posiciones de memoria EEPROM consecutivos, empezando por la posición 4.

En la posición 0 se almacena la primera dirección libre para guardar el siguiente movimiento y el final de programa a ejecutar, Al principio la memoria se inicializa a 255.

Las posición 1,2,3 se reservan para futuras ampliaciones.

PROGRAMACIÓN DE LA ESP32:

Utilizamos dos archivos por comodidad:

- ESP32Robot.ino: archivo principal.
- Robot.h : funciones de la tarjeta SDCard copiadas de <https://gist.github.com/youjunjer/b70b6e54ae7201a46387b8e73894ba51> con algunas modificaciones.

Librerias y variables

```
// Placa control servos
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include "driver/gpio.h"
#include "Arduino.h"
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);
// Para leer y escribir en flash memory
#include <EEPROM.h>
//sdcard
#include "FS.h"
#include "SD_MMC.h"
#include "robot.h"
// Para el ESP32 wifi
#include <WiFi.h> 2
WiFiClient WIFI_CLIENT;
// Para comunicarse mediante MQTT
#include <PubSubClient.h>
PubSubClient MQTT_CLIENT;

int grado;      //para lectura de caracter
// Posición de servos en la tarjeta de control
static const int servoPin_base = 0;
static const int servoPin_codo = 4;
static const int servoPin_pinza = 8;
const int numero_servos = 4 ; //nº de servos
int posicion_servos[numero_servos]={90,90,95,2}; // posición inicial servos base,codo,pinza,retardo

//CAMBIAR LOS VALORES DE LA RED WIFI A UTILIZAR
const char* ssid = "xxxxx"; // Nombre de red
const char* password ="xxxxx"; // Clave de red
```

```

byte direccion_EEPROM =4 ;// empieza a almacenar en la dirección 4 la cero para almacenar la libre al iniciar
byte ultima_dir_EEPROM = 100; // ultima posición de memoria a utilizar máx 255
#define EEPROM_SIZE ultima_dir_EEPROM// define the number of bytes you want to access
String posicion_motores;
String receivedChar = ""; //mensaje recibido
String nombre_archivo; //Guardar y leer de la SDCard

//I2C pines (GPIO0, GPIO16) como SDA y SCL SDL
#define I2C_SDA 0
#define I2C_SCL 16
//esp32 SDA=21, SCL=22
//esp32-cam SDA=0, SCL=16 o 4/16 0 2/16
unsigned int pos0 = 172; // ancho de pulso en cuentas para posicion 0°
unsigned int pos180 = 565; // ancho de pulso en cuentas para la posicion 180°
int datos_separados[20]; // Para devolver enteros separados por ,

```

Funciones:

```

void separar_datos_int(String dato);
void setServo(uint8_t n_servo, int angulo);
void guardar_archivo_EEPROM();
void listadoDir(fs::FS &fs, const char * dirname, uint8_t levels);
void Leer_archivoSD_EEPROM(fs::FS &fs, const char * path);
void Cargar_archivoSD_EEPROM(fs::FS &fs, const char * path);
void inicio();
void publica( String dato_publicar);
void Guardar_dato(byte retardo);
void Guardar_secuencia(String secuencia);
void Mover_secuencia(String secuencia);
void Ejecutar();
void ver_memoria(int n_inicial, int n_final);
void mover_servos();
void mover_servo(int servo, int gradosm);
void borra_EEPROM();
void callback(char* recibido, byte* payload, unsigned int length);

```

Alguna de las funciones:

setup:

```

void setup() {
  // Wire.begin(); // esp32 no hace falta por defecto SDA=21, SCL=22
  Wire.begin(I2C_SDA,I2C_SCL); //esp32CAM SDA=0, SCL=16
  servos.begin();
  servos.setPWMFreq(60); //Frecuencia PWM de 60Hz o T=16,66ms
  EEPROM.begin(EEPROM_SIZE);// initialize EEPROM with predefined size
  //direccion_EEPROM = EEPROM.read(0); carga dirección libre al iniciar
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.print("Conectando con ");
  Serial.println(ssid);
  setup_sdcard();
  WiFi.begin(ssid, password);
  inicio(); // Inicializa brazo a su posición inicial
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.print("WiFi conectada IP: ");
  Serial.println(WiFi.localIP());
}

```

```

// LLama Callback.
MQTT_CLIENT.setCallback(callback);
Serial.println(EEPROM.read(0)); //muestra posoción inicial memoria libre
} //setup

```

callback: Cuando recibe un dato de la aplicación según el datos realiza la función asignada.

```

// Cuando recibe dato
void callback(char* recibido, byte* payload, unsigned int length) {
    Serial.print("Message received: ");
    Serial.print(recibido);
    Serial.print(" ");
    receivedChar = "";
    for (int i=0;i<length;i++) {
        receivedChar += (char)payload[i];
    }
    Serial.println(receivedChar);
    // -----> RESET
    if (receivedChar == "reset") {resetFunc(); //call reset
    }
    //-----
    // --> MUEVE SERVOS +5º o -5º
    else if (receivedChar == "1") {mover_servo(0,posicion_servos[0] + 5 );} //base
    else if (receivedChar == "2") {mover_servo(0,posicion_servos[0] - 5 );}
    else if (receivedChar == "3") {mover_servo(1,posicion_servos[1] + 5 );} //codo
    else if (receivedChar == "4") {mover_servo(1,posicion_servos[1] - 5 );}
    else if (receivedChar == "5") {mover_servo(2,posicion_servos[2] + 5 );} //pinza
    else if (receivedChar == "6") {mover_servo(2,posicion_servos[2] - 5 );}
    //-----
    // --> SERVOS A POSICIÓN INICIAL
    else if (receivedChar == "I") {inicio();}
    //-----
    // --> EJECUTA MOVIMIENTOS EN EEPROM DESDE LA DIRECCIÓN direccion_EEPROM
    else if (receivedChar == "E") {Ejecutar();} //Ejecuta programa en memoria desde la dirección direccion_EEPROM
    //-----
    // --> MEMORIA MEMORIA
    else if (receivedChar == "memoria") {ver_memoria(0,0);} //muestra toda la memoria en serial, SOLO PARA DEPURACION VER PUBLICAR
    //-----
    //MUESTRA BLOQUE DE MEMORIA (Ejemplo 0,2M)
    else if (receivedChar.indexOf("M") != -1) { // El mensaje contiene M ***** VER QUE DETECTE M AL FINAL *****
        receivedChar = receivedChar.substring(0, receivedChar.length() - 1); // Borra último caracter
        // Extrae numeros separados por ,
        separar_datos_int(receivedChar);
        Serial.println(datos_separados[0]);
        Serial.println(datos_separados[1]);
        ver_memoria(datos_separados[0],datos_separados[1]); // posición inicial y final
    }
    //-----
    //CAMBIA POSICIÓN DE MEMORIA (Ejemplo p,vC)
    else if (receivedChar.indexOf("C") != -1) { // El mensaje contiene C
        receivedChar = receivedChar.substring(0, receivedChar.length() - 1); // Borra último caracter
        // Extrae numeros separados por , VER DE HACER FUNCION
        separar_datos_int(receivedChar);
        ver_memoria(datos_separados[0], datos_separados[0]); // Ver memoria antes del cambio
        EEPROM.write(datos_separados[0], datos_separados[1]); EEPROM.commit();
        ver_memoria(datos_separados[0], datos_separados[0]); // Ver memoria después del cambio
    }
    //-----
    // --> GUARDA EN EEPROM LA POSICIÓN ACTUAL DE LOS SERVOS. PASAMOS TIEMPO DE RETARDO EN S. (Ejemplo 2G)
    else if (receivedChar.indexOf("G") != -1) { // El mensaje contiene G
        receivedChar = receivedChar.substring(0, receivedChar.length() - 1); // Borra último caracter
        grado = receivedChar.toInt(); Guardar_dato(grado);
    }
    //-----
    // --> ALMACENA EL CONTENIDO DE EEPROM EN LA SDCard
    else if (receivedChar.indexOf("S") != -1) { // El mensaje contiene S
        receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
        nombre_archivo =receivedChar;
    }

```

```

guardar_archivo_EEPROM();
}

//-----
// --> GUARDA EN EEPROM LA POSICIÓN INDICADA DE LOS SERVOS (Ejemplo 90,160,25,2A) base,codo, pinza, espera
else if (receivedChar.indexOf("A") != -1) { // El mensaje contiene A
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    Guardar_secuencia(receivedChar);
}
//-----
// --> MUEVE SERVOS A LA POSICIÓN PASADA (Ejemplo 90,160,25H) base,codo, pinza
// Ejecuta secuencia (Ejemplo 90,160,25H) base,codo, pinza
else if (receivedChar.indexOf("H") != -1) { // El mensaje contiene H
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    Mover_secuencia(receivedChar);
}
//-----
// --> PUBLICA LISTADO DE ARCHIVOS
else if (receivedChar == "D") {
    listadoDir(SD_MMC, "/", 2);
} // listado SDCARD
//-----
// --> LEE ARCHIVO SDCard Y LO CARGA EN EEPROM
else if (receivedChar.indexOf("L") != -1) { // El mensaje contiene L lee archivo 1L
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    receivedChar = "/" + receivedChar + ".txt";
    char nombrebuffer[60] ; //convierte string a char
    receivedChar.toCharArray(nombrebuffer, 60);
    Cargar_archivoSD_EEPROM(SD_MMC,nombrebuffer);
}
//-----
// --> BORRAR ARCHIVO SDCard
else if (receivedChar.indexOf("B") != -1) { // El mensaje contiene B borra archivo 1B
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    receivedChar = "/" + receivedChar + ".txt";
    char nombrebuffer[60] ; //convierte string a char
    receivedChar.toCharArray(nombrebuffer, 60);
    deleteFile(SD_MMC, nombrebuffer);
    publica("Archivo borrado: " + receivedChar);
}
//-----
// --> VER ARCHIVO SDCard
else if (receivedChar.indexOf("R") != -1) { // El mensaje contiene R lee archivo 1R
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    receivedChar = "/" + receivedChar + ".txt";
    char nombrebuffer[60] ; //convierte string a char
    receivedChar.toCharArray(nombrebuffer, 60);
// readfile(SD_MMC,nombrebuffer);
    Leer_archivoSD_EEPROM(SD_MMC,nombrebuffer);
}
//-----
// --> BORRA MEMORIA EEPROM
else if (receivedChar == "borrar") { borrar_EEPROM();} // Borra programa en memoria EEPROM
//-----

// --> MUEVE SERVOS A LA POSICIÓN DADA (50b)
else if (receivedChar.indexOf("b") != -1) { //El mensaje contiene b Mueve base a los grados enviados
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); mover_servo(0,grado);
}

else if (receivedChar.indexOf("c") != -1) { //El mensaje contiene c Mueve codo a los grados enviados
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); mover_servo(1,grado);
}

else if (receivedChar.indexOf("p") != -1) { // El mensaje contiene p Mueve pinza a los grados enviados
    receivedChar= receivedChar.substring(0, receivedChar.length() - 1); // Delete last char p
    grado = receivedChar.toInt(); mover_servo(2,grado);
}

//--> CAMBIAR PIN DEL SERVO PARA PRUEBAS
if (receivedChar.indexOf("**") != -1) { // if message contain b
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); myservo_base.attach(grado);
}

```

```

}
if (receivedChar.indexOf("b") != -1) { // if message contain b
receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
grado = receivedChar.toInt(); myservo_codo.attach(grado);
}
//--> CAMBIAR PIN DEL SERVO PARA PRUEBAS
*/
// PUBLICA POSICIÓN SERVOS Y MEMORIA LIBRE
publica("b,c,p M");
String info = String(posicion_servos[0]) + "," + String(posicion_servos[1]) + "," + String(posicion_servos[2]) + " M:" +
String(EEPROM.read(0));
publica(info);

} //callback

```

loop:

```

void loop() {
if (!MQTT_CLIENT.connected()) {
reconnect();
}
MQTT_CLIENT.loop(); // Check Subscription.

} //loop

```

reconnect: Conecta con el broker broker.hivemq.com. No necesita registrarse ni autenticar.

```

// Reconecta con MQTT broker
void reconnect() {
MQTT_CLIENT.setServer("broker.hivemq.com", 1883);
//MQTT_CLIENT.setServer("mqtt.eclipse.org", 1883);
MQTT_CLIENT.setClient(WIFI_CLIENT);

// Intentando conectarse con Broker.
while (!MQTT_CLIENT.connected()) {
Serial.println("Intentando conectarse con Broker MQTT.");
MQTT_CLIENT.connect("jarp"); // No es necesario
MQTT_CLIENT.subscribe("jarp/servo"); // SUBSCRIBE.
delay(3000); // Espera para intentar volver a conectarse ...
}

Serial.println("Conectado a MQTT.");
}

```

Publica:

```

// Publica dato y muestra en el terminal
void publica( String dato_publicar){ // Publica dato
char datosbuffer[60] ;
dato_publicar.toCharArray(datosbuffer, 60);
Serial.println(datosbuffer);
MQTT_CLIENT.publish("jarp/posicion",datosbuffer);
}

```

Como podemos ver este ejemplo se puede utilizar para otras aplicaciones que podemos controlar facilmente desde nuestra aplicación mediante Wifi, con sencillos campos.

ANEXOS:

SCRIPs:

scrip: ESP32Robot.ino

```
/*
 * Control Brazo robot con 3 servos
 * Utilizamos ESP32-Cam por el lector de tarjetas SDCard no utilizamos la camara
 * Modulo control de servos PCA9685 (hasta 16 servos)
 * Control por I2c //esp32-cam SDA=2, SCL=16 //esp32 SDA=21, SCL=22
 * Contro mediante Wifi y MQTT
 * Podemos cargar y guardar movimientos en SDCard
 * Jarp 2020
 */
// Placa control servos
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include "driver/gpio.h"
//#include "esp_camera.h"
#include "Arduino.h"
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);
// Para leer y escribir en flash memory
#include <EEPROM.h>
//sdcard
#include "FS.h"
#include "SD_MMC.h"
#include "robot.h"
// Para el ESP3 wifi
#include <WiFi.h> 2
WiFiClient WIFI_CLIENT;
// Para comunicarse mediante MQTT
#include <PubSubClient.h>
PubSubClient MQTT_CLIENT;

int grado;      //para lectura de caracter
// Posición de servos en la tarjeta de control
static const int servoPin_base = 0;
static const int servoPin_codo = 4;
static const int servoPin_pinza = 8;
const int numero_servos = 4 ; //nº de servos
int posicion_servos[numero_servos]={90,90,95,2}; // posición inicial servos base,codo,pinza,retardo

//CAMBIAR LOS VALORES DE LA RED WIFI A UTILIZAR
const char* ssid = "xxxxx"; // Nombre de red
const char* password ="xxxxx"; // Clave de red

byte direccion_EEPROM =4 ;// empieza a almacenar en la dirección 4 la cero para almacenar la libre al iniciar
byte ultima_dir_EEPROM = 100; // ultima posición de memoria a utilizar máx 255
#define EEPROM_SIZE ultima_dir_EEPROM// define the number of bytes you want to access
String posicion_motores;
String receivedChar = ""; //mensaje recibido
String nombre_archivo; //Guardar y leer de la SDCard

//I2C pines (GPIO0, GPIO16) como SDA y SCL SDL
#define I2C_SDA 0
#define I2C_SCL 16
//esp32 SDA=21, SCL=22
//esp32-cam SDA=0, SCL=16 o 4/16 0/2/16
unsigned int pos0 = 172; // ancho de pulso en cuentas para posicion 0°
unsigned int pos180 = 565; // ancho de pulso en cuentas para la posicion 180°
int datos_separados[20]; // Para devolver enteros separados por ,

//.....PROTOTIPOS .....
void separar_datos_int(String dato);
void setServo(uint8_t n_servo, int angulo);
void guardar_archivo_EEPROM();
void listadoDir(fs::FS &fs, const char * dirname, uint8_t levels);
void Leer_archivosSD_EEPROM(fs::FS &fs, const char * path);
```

```

void Cargar_archivoSD_EEPROM(fs::FS &fs, const char * path);
void inicio();
void publica( String dato_publicar);
void Guardar_dato(byte retardo);
void Guardar_secuencia(String secuencia);
void Mover_secuencia(String secuencia);
void Ejecutar();
void ver_memoria(int n_inicial, int n_final);
void mover_servos();
void mover_servo(int servo, int gradosm);
void borra_EEPROM();
void callback(char* recibido, byte* payload, unsigned int length);
//.....
void setup() {
    // Wire.begin(); // esp32 no hace falta por defecto SDA=21, SCL=22
    Wire.begin(I2C_SDA,I2C_SCL); //esp32CAM SDA=0, SCL=16
    servos.begin();
    servos.setPWMDutyCycle(60); //Frecuencia PWM de 60Hz o T=16,66ms
    EEPROM.begin(EEPROM_SIZE); // initialize EEPROM with predefined size
    //direccion_EEPROM = EEPROM.read(0); carga dirección libre al iniciar
    Serial.begin(115200);
    delay(10);
    Serial.println();
    Serial.print("Conectando con ");
    Serial.println(ssid);
    setup_sdcard();
    WiFi.begin(ssid, password);
    inicio(); // Inicializa brazo a su posición inicial
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.print("WiFi conectada IP: ");
    Serial.println(WiFi.localIP());

    // Llama Callback.
    MQTT_CLIENT.setCallback(callback);
    Serial.println(EEPROM.read(0)); //muestra posoción inicial memoria libre
} //setup

//.....
// RESET por botón
void(* resetFunc) (void) = 0;//declare reset function at address 0
//.....
// Separa string de numeros separados por comas *****VER DE PASAR CARACTER DE SEPARACIÓN*****
// Los almacena en array int datos_separados
void separar_datos_int(String dato){
    char dato_buffer[30] ;
    dato.toCharArray(dato_buffer, 30);
    char *puntero = dato_buffer; //puntero
    char *str_dato;

    int i=0;
    while ((str_dato = strtok_r(puntero, ",", &puntero)) != NULL) { // Divide datos separados con ,
        datos_separados[i]= atol(str_dato); // atol: char a entero
        i++;
    }
}

//.....
// Mover servo a un angulo
void setServo(uint8_t n_servo, int angulo) {
    int duty;
    duty = map(angulo, 0, 180, pos0, pos180);
    servos.setPWM(n_servo, 0, duty);
}

//Guarda programa en EEPROM a SDcard
void guardar_archivo_EEPROM(){
    nombre_archivo = "/" + nombre_archivo + ".txt";
    char nombrebuffer[60];
    nombre_archivo.toCharArray(nombrebuffer, 60); // Pasamos a string
}

```

```

//crear_archivo();
File file = SD_MMC.open(nombrebuffer);
if(!file) {
    Serial.println("El archivo no existe");
    Serial.println("Creando archivo... ");
    writeFile(SD_MMC, nombrebuffer, ""); //para que no de error al no pasar dato
}
else {
    Serial.println("El archivo ya existe");
}
file.close();
//Guarda EEPROM
byte direccion = direccion_EEPROM; //Dirección inicial del programa
Serial.println(EEPROM.read(0));
String dato_EEPROM; //linea para Guardar
while(EEPROM.read(direccion)!= 255){ //termina al encontrar 255
    for (int i=0; i < numero_servos ;i++){ //datos de los 3 servos b,c,p ,retardo
        byte dato_memoria = EEPROM.read(direccion);
        if (i==0) {dato_EEPROM = String(dato_memoria);} // quita la coma inicial
        else {dato_EEPROM += "," + String(dato_memoria);}
        direccion++;
    }
    dato_EEPROM += "\r\n"; //fin de linea
    Serial.print("Dato guardado: ");
    Serial.println(dato_EEPROM);
    appendFile(SD_MMC, nombrebuffer, dato_EEPROM.c_str()); //c_str() accede al puntero de la cadena, no se debe utilizar la
    cadena más
}
Serial.println("Fin guardar SDcard: " + nombre_archivo);
publica("Fin guardar SDcard: " + nombre_archivo);
}

//.....
//Listado de archivos en SDcard
void listadoDir(fs::FS &fs, const char * dirname, uint8_t levels){
String dato_publica;
//Serial.printf("Listado directorio: %s\n", dirname);
publica("Listado directorio: " );
//publica(dirname);
File root = fs.open(dirname);
if(!root){
    //Serial.println("Error al abrir directorio");
    publica("Error al abrir directorio");
    return;
}
if(!root.isDirectory()){
    //Serial.println("No es directorio");
    publica("No es directorio");
    return;
}

File file = root.openNextFile();
while(file){
    if(file.isDirectory()){
        // Serial.print(" DIR: ");
        dato_publica = " DIR: ";
        // Serial.println(file.name());
        dato_publica += file.name();
        publica(dato_publica);
        if(levels){
            listDir(fs, file.name(), levels -1);
        }
    } else {
        dato_publica = " FILE: ";
        dato_publica += file.name();
        dato_publica += " SIZE: ";
        dato_publica += file.size();
        // Serial.print(" FILE: ");
        // Serial.print(file.name());
        // Serial.print(" SIZE: ");
        // Serial.println(file.size());
    }
    publica(dato_publica);
    file = root.openNextFile();
}
}

```

```

//.....  

//Lee archivo en SD card  

// Formato del archivo b,c,p,retardo  

void Leer_archivoSD_EEPROM(fs::FS &fs, const char * path){  

int fin_memoria = 0;  

Serial.printf("Leer archivo: %s\n", path);  

publica("Leer archivo:");  

publica(path);  

File file = fs.open(path);  

if(!file){  

    Serial.println("Error leyendo archivo");  

    publica("Error leyendo archivo");  

    return;  

}  
  

Serial.println("Leyendo desde archivo a EEPROM: ");  

char linea[20]; // linea entera del archivo  

// Lee archivo hasta el final  

while(file.available()){  

    // Lee linea  

linea[file.readBytesUntil('\r', linea, sizeof(linea) - 1)] = 0; //lee caracteres hasta \r c10  

file.read(); // Descarta '\n' para así leer la siguiente linea correctamente quita el retorno de linea c14.  

publica(linea);  

// extrae caracter y lo guarda en EEPROM  

fin_memoria = fin_memoria + 4; //apunta a proxima dirección libre (cada linea 4 posiciones de memoria b,c,p,retardo)  

}  
//while  

EEPROM.write(0, fin_memoria); EEPROM.commit(); //actualiza fin de programa  

//Serial.println("Fin cargar EEPROM");  

publica("Fin leer archivo");  

}  
//.....  

//Carga archivo a EEPROM dede la SD card  

// Formato del archivo b,c,p,retardo  

void Cargar_archivoSD_EEPROM(fs::FS &fs, const char * path){  

borra_EEPROM() ; // borra memoria  

int fin_memoria = 0;  

Serial.printf("Leer archivo: %s\n", path);  

publica("Leer archivo:");  

publica(path);  

File file = fs.open(path);  

if(!file){  

    Serial.println("Error leyendo archivo");  

    publica("Error leyendo archivo");  

    return;  

}  
  

Serial.println("Leyendo desde archivo a EEPROM: ");  

char linea[20]; // linea entera del archivo  

// Lee archivo hasta el final  

while(file.available()){  

    // Lee linea  

linea[file.readBytesUntil('\r', linea, sizeof(linea) - 1)] = 0; //lee caracteres hasta \r c10  

file.read(); // Descarta '\n' para así leer la siguiente linea correctamente quita el retorno de linea c14.  

// Serial.println(linea);  

publica(linea);  

// extrae caracter y lo guarda en EEPROM  

fin_memoria = fin_memoria + 4; //apunta a proxima dirección libre (cada linea 4 posiciones de memoria b,c,p,retardo)  

Guardar_secuencia(linea);  

}  
//while  

EEPROM.write(0, fin_memoria); EEPROM.commit(); //actualiza fin de programa  

//Serial.println("Fin cargar EEPROM");  

publica("Fin cargar EEPROM");  

}  
//.....  

// Lleva robot a la posición inicial  

void inicio(){  

    posicion_servos[0] = 90;  

    posicion_servos[1] = 90;  

    posicion_servos[2] = 95; //cierra pinza  

    mover_servos();  

}  
//.....  

// Publica dato y muestra en el terminal  

void publica( String dato_publicar){ // Publica dato  

    char datosbuffer[60];

```

```

dato_publicar.toCharArray(datosbuffer, 60);
Serial.println(datosbuffer);
MQTT_CLIENT.publish("jarp/posicion",datosbuffer);
}

//.....
//void publica_depurar( String dato_publicar){ // Publica dato para depurar REVISAR ¿No deja pasar nombre a ubicar?
// char datosbuffer[60] ;
// dato_publicar.toCharArray(datosbuffer, 60);
// MQTT_CLIENT.publish("jarp/depurar",datosbuffer);
//}

//.....



void Guardar_dato(byte retardo){ //rG: Guarda posición del brazo en la EEPROM
byte direccion = EEPROM.read(0);
for (int i=0; i < numero_servos -1 ;i++){
    EEPROM.write(direccion , posicion_servos[i]); EEPROM.commit();
    direccion++;
    Serial.println(posicion_servos[i]);
// publica(posicion_servos[i]);
}
EEPROM.write(direccion, retardo); EEPROM.commit(); //Almacena posición retardo
direccion++;
EEPROM.write(0, direccion); EEPROM.commit(); //Almacena siguiente posición libre

}

//.....
//Guarda movimientos en la EEPROM (b,c,p,retardo)
void Guardar_secuencia(String secuencia){
char secuencia_buffer[30];
secuencia.toCharArray(secuencia_buffer, 30);
char *p = secuencia_buffer; //puntero
char *str;
String info1;
int i=0;
while ((str = strtok_r(p, " ", &p)) != NULL) { // Divide datos separados con ,
posicion_servos[i]= atol(str); //0:b, 1:c, 2:p, 3: retardo atol: char a entero
info1 += " " + String(str);
i++;
}
Guardar_dato(posicion_servos[3]); //valor de retardo
publica(info1);
}
//.....
//Mueve servos según secuencia(b,c,p) b,c,pH . Se pueden omitir los ultimos
void Mover_secuencia(String secuencia){
char secuencia_buffer[30];
secuencia.toCharArray(secuencia_buffer, 30);
char *p = secuencia_buffer; //puntero
char *str;
String info1;
int i=0;
while ((str = strtok_r(p, " ", &p)) != NULL) { // Divide datos separados con ,
posicion_servos[i]= atol(str); //0:b, 1:c, 2:p, 3: espera atol: char a entero
info1 += " " + String(str);
i++;
}
mover_servos();
publica(info1);
}
//.....
//Ejecuta programa en EEPROM
void Ejecutar(){
String retardo;
byte direccion = direccion_EEPROM;
Serial.println(EEPROM.read(0));
publica("Fin programa en memoria: " + String(EEPROM.read(0)));
publica("M,b,c,p,R");
while(EEPROM.read(direccion)!= 255){ //termina al encontrar 255
    String dato_EEPROM = String(direccion); //para publicar
    Serial.print(" Dirección ");
    Serial.print(dato_EEPROM);
    for (int i=0; i < numero_servos -1 ;i++){ //datos de los 3 servos b,c,p
        byte dato_memoria = EEPROM.read(direccion);
        Serial.print(" Dato " + String(i) + ":" );
        Serial.print(dato_memoria);
    }
}
}

```

```

Serial.print(" ");
mover_servo(i,dato_memoria);
dato_EEPROM += ", " + String(dato_memoria);
direccion++;
}
Serial.print(" ");
Serial.println(String(EEPROM.read(direccion)));
dato_EEPROM += ", " + (String(EEPROM.read(direccion)));
delay(EEPROM.read(direccion)*1000); //retardo
direccion++;
publica(dato_EEPROM);
} // while
// Serial.println("Fin secuencia");
publica("Fin secuencia");
}
//.....
// Muestra memoria de n_inicial a n_final
void ver_memoria(int n_inicial, int n_final){
if (n_final == 0) {n_final = EEPROM.read(0);} // muestra hasta la 1ª libre
Serial.println(n_inicial);
Serial.println(n_final);
for (int i= n_inicial; i< n_final + 1 ;i++){ //muestra memoria
byte dato_memoria = EEPROM.read(i);
// Serial.print(" (" + String(i) + "): ");
// Serial.print(dato_memoria);
// Serial.println(" ");
publica(" (" + String(i) + "): " + dato_memoria);
}
}
//.....
// Mueve servos a la posición guardada en posicion_servos[]
void mover_servos(){
if (posicion_servos[0] < 0 ) {posicion_servos[0] = 0;}
if (posicion_servos[0] > 180 ) {posicion_servos[0] = 180;}
setServo(servоСPin_base,posicion_servos[0]);

if (posicion_servos[1] < 0 ) {posicion_servos[1] = 0;}
if (posicion_servos[1] > 180 ) {posicion_servos[1] = 180;}
setServo(servоСPin_codo,posicion_servos[1]);

if (posicion_servos[2] < 25 ) {posicion_servos[2] = 25;}
if (posicion_servos[2] > 95 ) {posicion_servos[2] = 95;}
setServo(servоСPin_pinza, posicion_servos[2]);
}
//.....
// Mueve servo a la posición pasada
void mover_servo(int servo, int gradosm){
switch (servo){
case 0:
if (gradosm < 0 ) {gradosm = 0;}
if (gradosm > 180 ) {gradosm = 180;}
posicion_servos[0] = gradosm;
setServo(servоСPin_base, gradosm);
break;
case 1:
if (gradosm < 0 ) {gradosm = 0;}
if (gradosm > 180 ) {gradosm = 180;}
posicion_servos[1] = gradosm;
setServo(servоСPin_codo, gradosm);
break;
case 2:
if (gradosm < 25 ) {gradosm = 25;}
if (gradosm > 95 ) {gradosm = 95;}
posicion_servos[2] = gradosm;
setServo(servоСPin_pinza, gradosm);
break;
}
}
//.....
void borra_EEPROM(){
EEPROM.write(0, direccion_EEPROM); EEPROM.commit(); //pone el puntero al inicio
for (int i = direccion_EEPROM; i< ultima_dir_EEPROM +1 ;i++){ // Borra memoria poniendola a 255
EEPROM.write(i, 255); EEPROM.commit();
}
}

```

```

    publica("Memoria de programa Borrada");
}
//.....
// Cuando recibe dato
void callback(char* recibido, byte* payload, unsigned int length) {
    Serial.print("Message received: ");
    Serial.print(recibido);
    Serial.print(" ");
    receivedChar = "";
    for (int i=0;i<length;i++) {
        receivedChar += (char)payload[i];
    }
    Serial.println(receivedChar);
    // -----> RESET
    if (receivedChar == "reset") {resetFunc(); //call reset
    }
//-----
// --> MUEVE SERVOS +5º o -5º
else if (receivedChar == "1") {mover_servo(0,posicion_servos[0] + 5 );} //base
else if (receivedChar == "2") {mover_servo(0,posicion_servos[0] - 5 );}
else if (receivedChar == "3") {mover_servo(1,posicion_servos[1] + 5); } //codo
else if (receivedChar == "4") {mover_servo(1,posicion_servos[1] - 5); }
else if (receivedChar == "5") {mover_servo(2,posicion_servos[2] + 5);} //pinza
else if (receivedChar == "6") {mover_servo(2,posicion_servos[2] - 5);}
//-----
// --> SERVOS A POSICIÓN INICIAL
else if (receivedChar == "I") {inicio();}
//-----
// --> EJECUTA MOVIMIENTOS EN EEPROM DESDE LA DIRECCIÓN direccion_EEPROM
else if (receivedChar == "E") {Ejecutar();} //Ejecuta programa en memoria desde la dirección direccion_EEPROM
//-----
// --> MEMORIA MEMORIA
else if (receivedChar == "memoria") {ver_memoria(0,0);} //muestra toda la memoria en serial, SOLO PARA DEPURACION VER PUBLICAR
//-----
//MUESTRA BLOQUE DE MEMORIA (Ejemplo 0,2M)
else if (receivedChar.indexOf("M") != -1) { // El mensaje contiene M ***** VER QUE DETECTE M AL FINAL*****
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1); // Borra último caracter
    // Extrae numeros separados por ,
    separar_datos_int(receivedChar);
    Serial.println(datos_separados[0]);
    Serial.println(datos_separados[1]);
    ver_memoria(datos_separados[0],datos_separados[1]); // posición inicial y final
}
//-----
//CAMBIA POSICIÓN DE MEMORIA (Ejemplo p,vC)
else if (receivedChar.indexOf("C") != -1) { // El mensaje contiene C
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1); // Borra último caracter
    // Extrae numeros separados por , VER DE HACER FUNCION
    separar_datos_int(receivedChar);
    ver_memoria(datos_separados[0], datos_separados[0]); // Ver memoria antes del cambio
    EEPROM.write(datos_separados[0], datos_separados[1]); EEPROM.commit();
    ver_memoria(datos_separados[0], datos_separados[0]); // Ver memoria después del cambio
}
//-----
// --> GUARDA EN EEPROM LA POSICIÓN ACTUAL DE LOS SERVOS. PASAMOS TIEMPO DE RETARDO EN S. (Ejemplo 2G)
else if (receivedChar.indexOf("G") != -1) { // El mensaje contiene G
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1); // Borra último caracter
    grado = receivedChar.toInt(); Guardar_dato(grado);
}
//-----
// --> ALMACENA EL CONTENIDO DE EEPROM EN LA SDCard
else if (receivedChar.indexOf("S") != -1) { // El mensaje contiene S
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    nombre_archivo =receivedChar;
    guardar_archivo_EEPROM();
}
//-----
// --> GUARDA EN EEPROM LA POSICIÓN INDICADA DE LOS SERVOS (Ejemplo 90,160,25,2A) base,codo, pinza, espera
else if (receivedChar.indexOf("A") != -1) { // El mensaje contiene A
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    Guardar_secuencia(receivedChar);
}
//-----
// --> MUEVE SERVOS A LA POSICIÓN PASADA (Ejemplo 90,160,25H) base,codo, pinza

```

```

// Ejecuta secuencia (Ejemplo 90,160,25H) base,codo, pinza
else if (receivedChar.indexOf("H") != -1) { // El mensaje contiene H
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    Mover_secuencia(receivedChar);
}
//-----
// --> PUBLICA LISTADO DE ARCHIVOS
else if (receivedChar == "D") {
    listadoDir(SD_MMC, "/", 2);
} // listado SDCARD
//-----
// --> LEE ARCHIVO SDCard Y LO CARGA EN EEPROM
else if (receivedChar.indexOf("L") != -1) { // El mensaje contiene L lee archivo 1L
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    receivedChar = "/" + receivedChar + ".txt";
    char nombrebuffer[60] ; //convierte string a char
    receivedChar.toCharArray(nombrebuffer, 60);
    Cargar_archivoSD_EEPROM(SD_MMC,nombrebuffer);
}
//-----
// --> BORRAR ARCHIVO SDCard
else if (receivedChar.indexOf("B") != -1) { // El mensaje contiene B borra archivo 1B
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    receivedChar = "/" + receivedChar + ".txt";
    char nombrebuffer[60] ; //convierte string a char
    receivedChar.toCharArray(nombrebuffer, 60);
    deleteFile(SD_MMC, nombrebuffer);
    publica("Archivo borrado: " + receivedChar);
}
//-----
// --> VER ARCHIVO SDCard
else if (receivedChar.indexOf("R") != -1) { // El mensaje contiene R lee archivo 1R
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    receivedChar = "/" + receivedChar + ".txt";
    char nombrebuffer[60] ; //convierte string a char
    receivedChar.toCharArray(nombrebuffer, 60);
// readFile(SD_MMC,nombrebuffer);
    Leer_archivoSD_EEPROM(SD_MMC,nombrebuffer);
}
//-----
// --> BORRA MEMORIA EEPROM
else if (receivedChar == "borrar") { borrar_EEPROM();} // Borra programa en memoria EEPROM
//-----

// --> MUEVE SERVOS A LA POSICIÓN DADA (50b)
else if (receivedChar.indexOf("b") != -1) { //El mensaje contiene b Mueve base a los grados enviados
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); mover_servo(0,grado);
}

else if (receivedChar.indexOf("c") != -1) { //El mensaje contiene c Mueve codo a los grados enviados
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); mover_servo(1,grado);
}

else if (receivedChar.indexOf("p") != -1) { // El mensaje contiene p Mueve pinza a los grados enviados
    receivedChar= receivedChar.substring(0, receivedChar.length() - 1); // Delete last char p
    grado = receivedChar.toInt(); mover_servo(2,grado);
}
//-----
/*
//--> CAMBIAR PIN DEL SERVO PARA PRUEBAS
if (receivedChar.indexOf("**") != -1) { // if message contain b
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); myservo_base.attach(grado);
}
if (receivedChar.indexOf("/") != -1) { // if message contain b
    receivedChar = receivedChar.substring(0, receivedChar.length() - 1);
    grado = receivedChar.toInt(); myservo_codo.attach(grado);
}
//--> CAMBIAR PIN DEL SERVO PARA PRUEBAS
*/
// PUBLICA POSICIÓN SERVOS Y MEMORIA LIBRE
publica("b,c,p M");
String info = String(posicion_servos[0]) + "," + String(posicion_servos[1]) + "," + String(posicion_servos[2]) + " M:" +

```

```

String(EEPROM.read(0));
publica(info);

} //callback

//.....
void loop() {
  if (!MQTT_CLIENT.connected()) {
    reconnect();
  }
  MQTT_CLIENT.loop(); // Check Subscription.

} //loop

//.....
// Reconecta con MQTT broker
void reconnect() {
  MQTT_CLIENT.setServer("broker.hivemq.com", 1883);
  //MQTT_CLIENT.setServer("mqtt.eclipse.org", 1883);
  MQTT_CLIENT.setClient(WIFI_CLIENT);

  // Intentando conectarse con Broker.
  while (!MQTT_CLIENT.connected()) {
    Serial.println("Intentando conectarse con Broker MQTT.");
    MQTT_CLIENT.connect("jarp"); // No es necesario
    MQTT_CLIENT.subscribe("jarp/servo"); // HERE SUBSCRIBE.
    delay(3000); // Espera para intentar volver a conectarse ...
  }

  Serial.println("Conectado a MQTT.");
}

```

scrip: Robot.h

Copiado de

<https://github.com/v12345vtm/CameraWebserver2SD/blob/master/CameraWebserver2SD/CameraWebserver2SD.ino>

```

// This post referred to this git. I just trimmed cam and wifi part.
// https://github.com/v12345vtm/CameraWebserver2SD/blob/master/CameraWebserver2SD/CameraWebserver2SD.ino

##include "FS.h"
##include "SD_MMC.h"

// Libraries for SD card esp32 sin cam

##include "SD.h"
##include <SPI.h>

//List dir in SD card
void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
  Serial.printf("Listing directory: %s\n", dirname);

  File root = fs.open(dirname);
  if(!root){
    Serial.println("Failed to open directory");
    return;
  }
  if(!root.isDirectory()){
    Serial.println("Not a directory");
    return;
  }
}
```

```

File file = root.openNextFile();
while(file){
    if(file.isDirectory()){
        Serial.print(" DIR : ");
        Serial.println(file.name());
        if(levels){
            listDir(fs, file.name(), levels -1);
        }
    } else {
        Serial.print(" FILE: ");
        Serial.print(file.name());
        Serial.print(" SIZE: ");
        Serial.println(file.size());
    }
    file = root.openNextFile();
}
}

//Create a dir in SD card
void createDir(fs::FS &fs, const char * path){
    Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
        Serial.println("Dir created");
    } else {
        Serial.println("mkdir failed");
    }
}

//delete a dir in SD card
void removeDir(fs::FS &fs, const char * path){
    Serial.printf("Removing Dir: %s\n", path);
    if(fs.rmdir(path)){
        Serial.println("Dir removed");
    } else {
        Serial.println("rmdir failed");
    }
}

//Read a file in SD card
void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);

    File file = fs.open(path);
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }

    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
}

//Write a file in SD card
void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }

    //fwrite(fb->buf, 1, fb->len, file);
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
}

```

```

}

//Append to the end of file in SD card
void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
}

//Rename a file in SD card
void renameFile(fs::FS &fs, const char * path1, const char * path2){
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2)) {
        Serial.println("File renamed");
    } else {
        Serial.println("Rename failed");
    }
}

//Delete a file in SD card
void deleteFile(fs::FS &fs, const char * path){
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("Archivo borrado");
//        publica_depurar("Archivo borrado");
    } else {
        Serial.println("Error borrado");
    }
}

//Test read and write speed using test.txt file
void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file){
        len = file.size();
        size_t flen = len;
        start = millis();
        while(len){
            size_t toRead = len;
            if(toRead > 512){
                toRead = 512;
            }
            file.read(buf, toRead);
            len -= toRead;
        }
        end = millis() - start;
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }

    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
}

```

```

size_t i;
start = millis();
for(i=0; i<2048; i++){
    file.write(buf, 512);
}
end = millis() - start;
Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
file.close();
}

void setup_sdcard() {

Serial.println("SDcard Testing....");

if(!SD_MMC.begin("/sdcard", true)){ //para que no luzca tanto el led al utilizar el p4
//if(!SD_MMC.begin()){
    Serial.println("Card Mount Failed");
    return;
}
uint8_t cardType = SD_MMC.cardType();

if(cardType == CARD_NONE){
    Serial.println("No SD_MMC card attached");
    return;
}

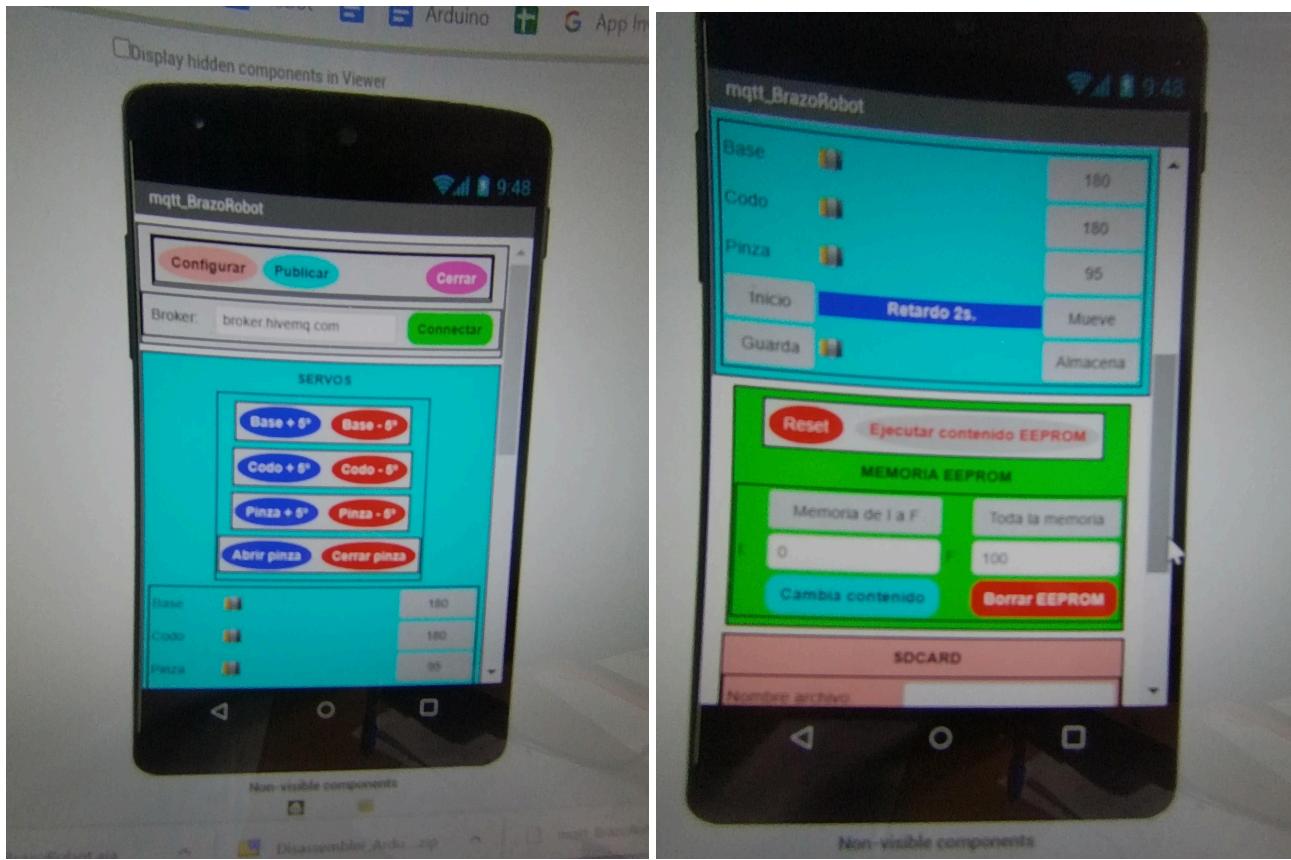
Serial.print("SD_MMC Card Type: ");
if(cardType == CARD_MMC){
    Serial.println("MMC");
} else if(cardType == CARD_SD){
    Serial.println("SDSC");
} else if(cardType == CARD_SDHC){
    Serial.println("SDHC");
} else {
    Serial.println("UNKNOWN");
}

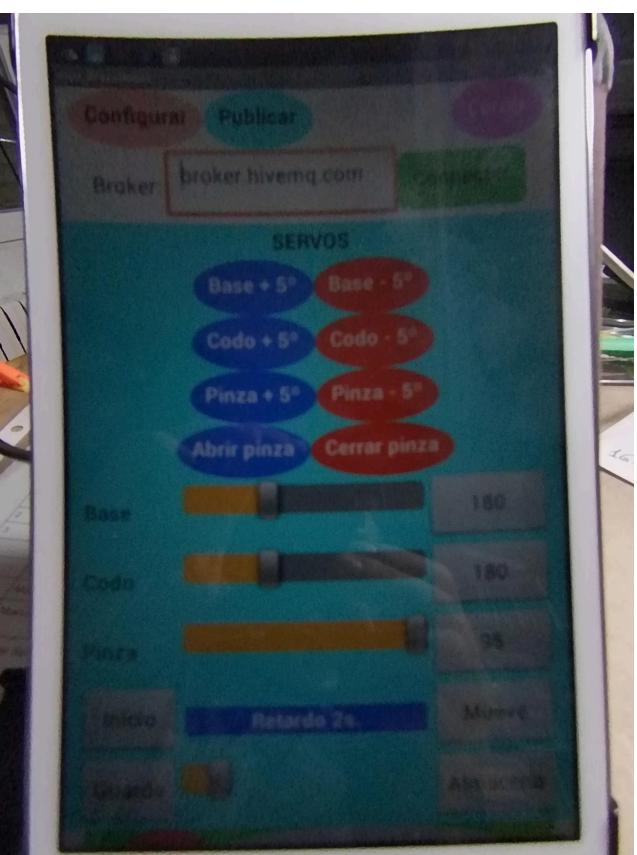
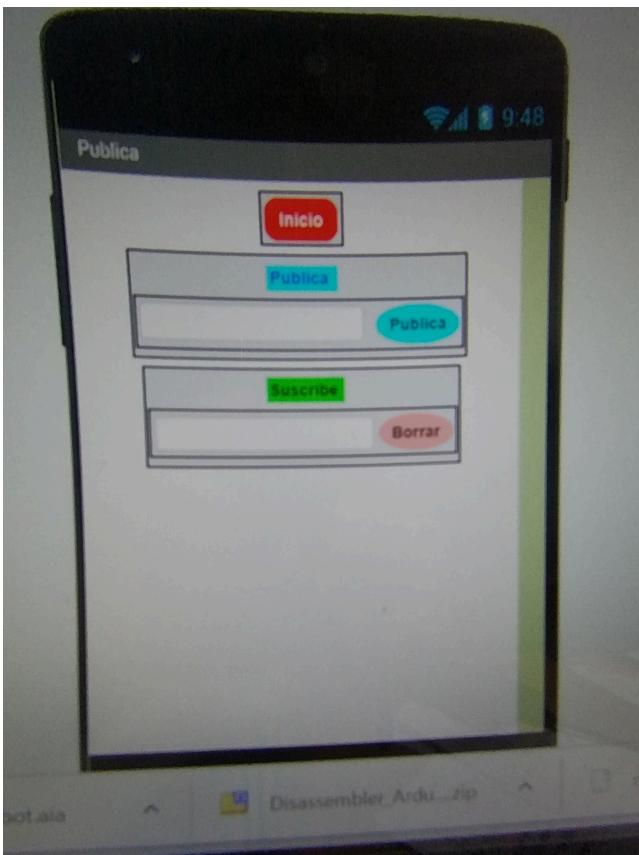
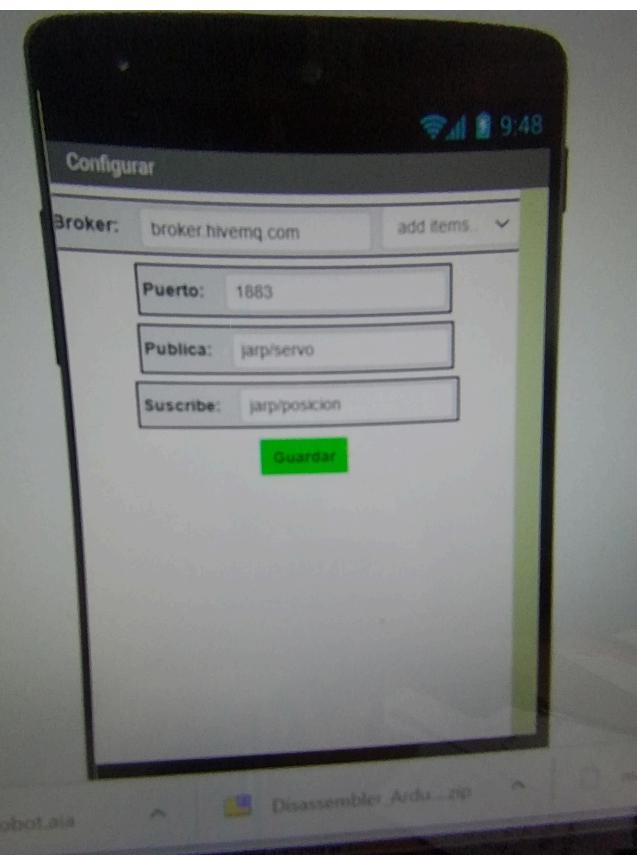
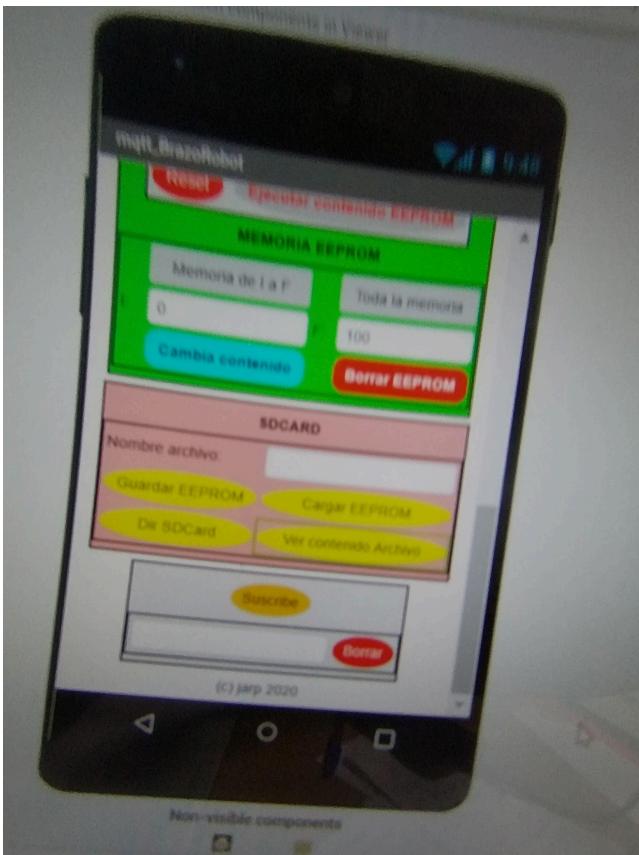
uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);

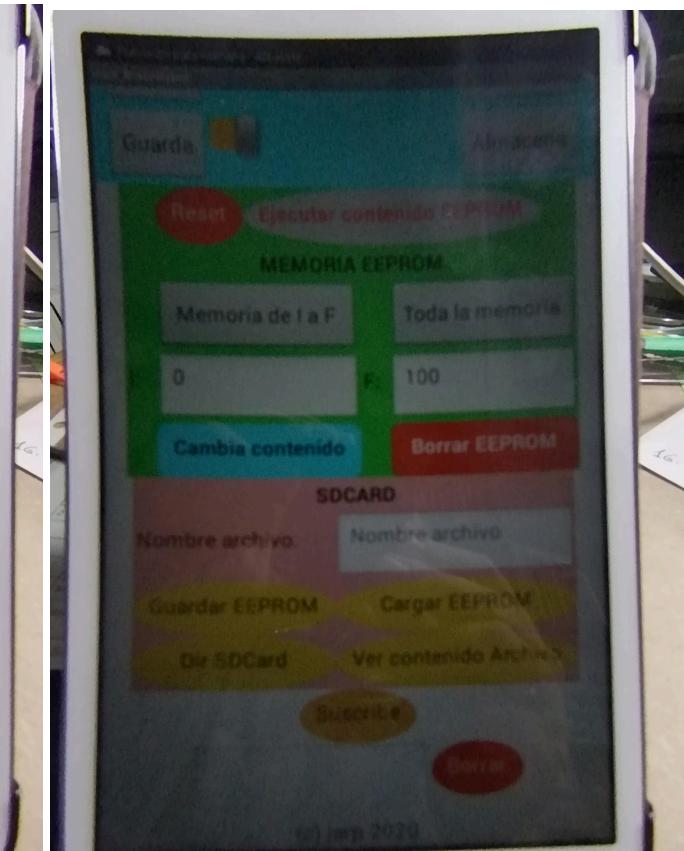
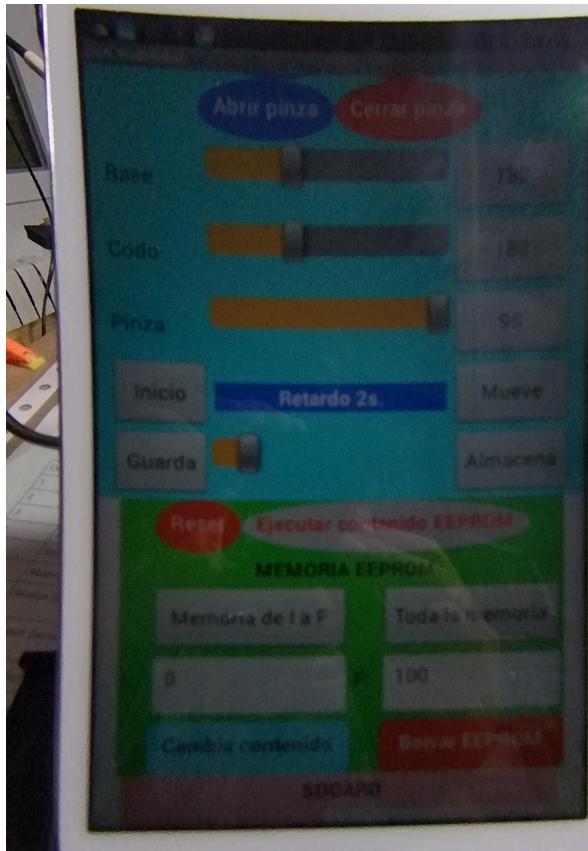
// listDir(SD_MMC, "/");
// createDir(SD_MMC, "/mydir");
listDir(SD_MMC, "/", 0);
// removeDir(SD_MMC, "/mydir");
listDir(SD_MMC, "/", 2);
// writeFile(SD_MMC, "/hello.txt", "Hello ");
// appendFile(SD_MMC, "/hello.txt", "World!\n");
// readFile(SD_MMC, "/hello.txt");
// deleteFile(SD_MMC, "/foo.txt");
// renameFile(SD_MMC, "/hello.txt", "/foo.txt");
// readFile(SD_MMC, "/foo.txt");
// testFileIO(SD_MMC, "/test.txt");
Serial.printf("Total space: %lluMB\n", SD_MMC.totalBytes() / (1024 * 1024));
Serial.printf("Used space: %lluMB\n", SD_MMC.usedBytes() / (1024 * 1024));
}
}

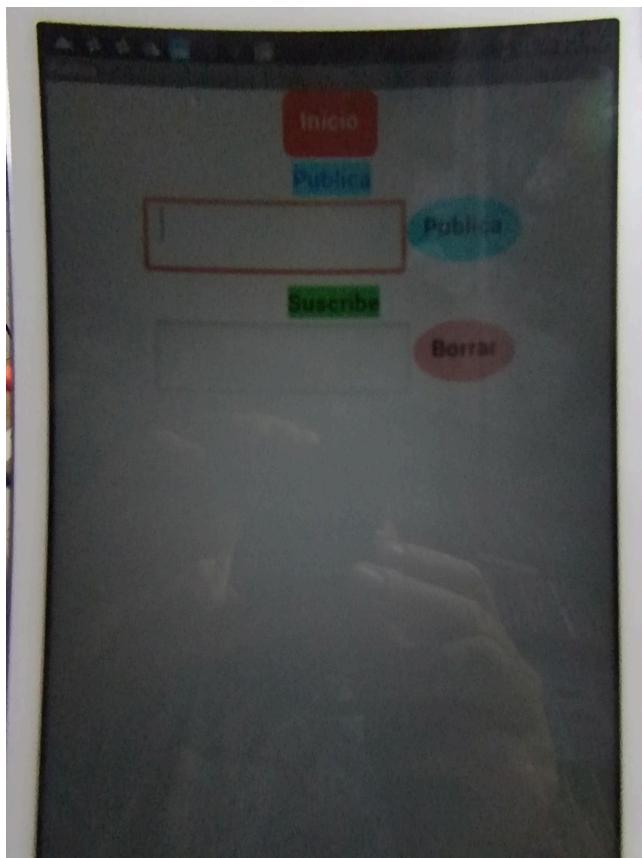
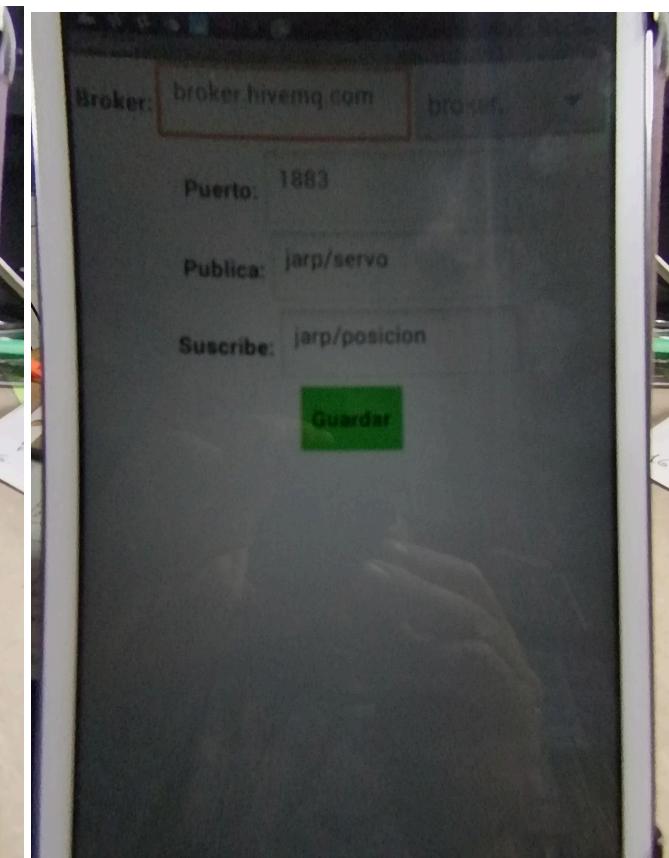
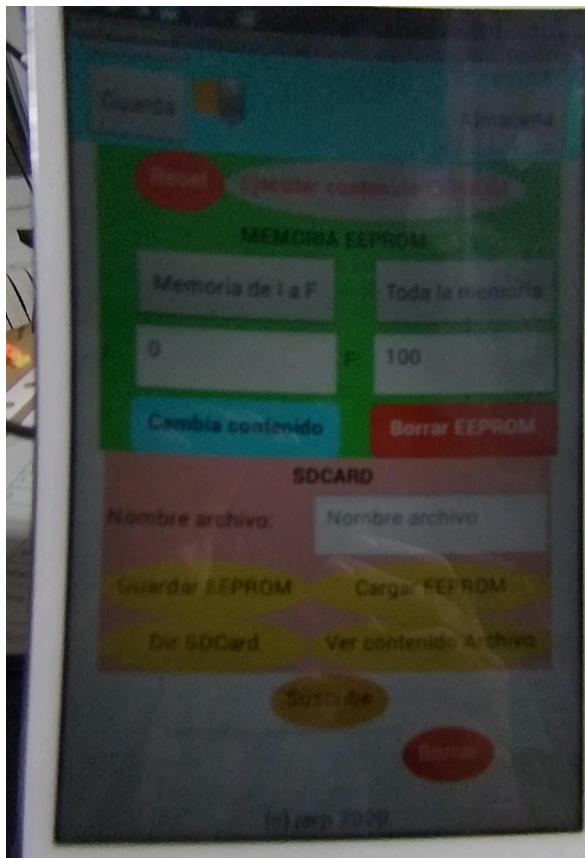
```

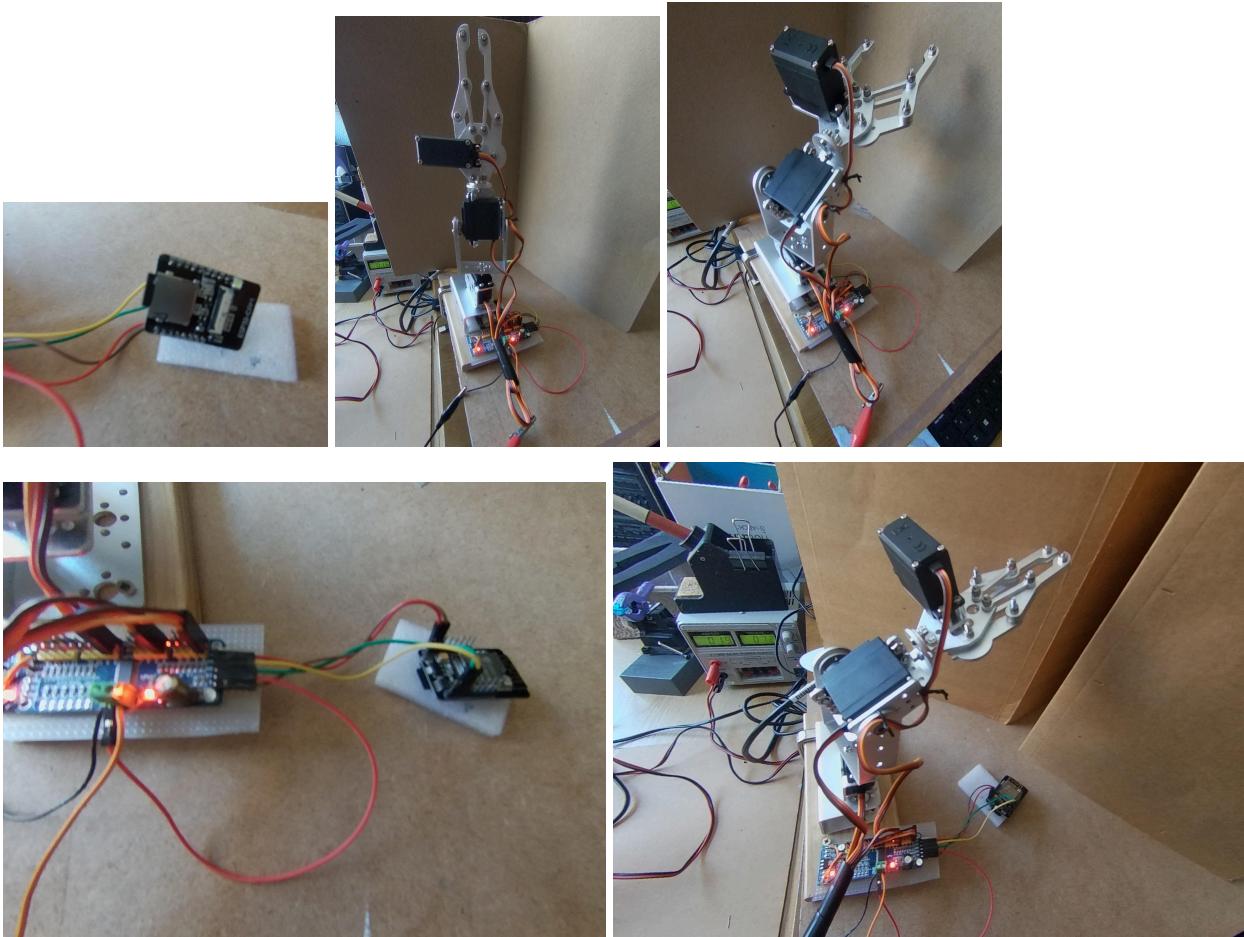
IMAGENES:











ARCHIVOS:

[ESP32Robot.zip](#)

[mqtt_BrazoRobot.aia](#)

[mqtt_BrazoRobot.apk](#)

