



C5.2 Interface Node-RED y NodeMCU ESP32

Arquitectura Cliente-Servidor, utilizando interface Node-red, ESP32 y un actuador



Instrucciones

- De acuerdo con la información presentada por el asesor referente al tema, desarrollar lo que se indica dentro del apartado siguiente.
- Toda actividad o reto se deberá realizar utilizando el estilo **MarkDown con extension .md** y el entorno de desarrollo VSCode, debiendo ser elaborado como un documento **single page**, es decir si el documento cuenta con imágenes, enlaces o cualquier documento externo debe ser accedido desde etiquetas y enlaces.
- Es requisito que el archivo .md contenga una etiqueta del enlace al repositorio de su documento en Github, por ejemplo **Enlace a mi GitHub**
- Al concluir el reto el reto se deberá subir a github el archivo .md creado.
- Desde el archivo **.md** se debe exportar un archivo **.pdf** con la nomenclatura **C5.2_NombreAlumno_Equipo.pdf**, el cual deberá subirse a classroom dentro de su apartado correspondiente, para que sirva como evidencia de su entrega; siendo esta plataforma **oficial** aquí se recibirá la calificación de su actividad por individual.
- Considerando que el archivo .pdf, fue obtenido desde archivo .md, ambos deben ser idénticos y mostrar el mismo contenido.
- Su repositorio ademas de que debe contar con un archivo **readme.md** dentro de su directorio raíz, con la información como datos del estudiante, equipo de trabajo, materia, carrera, datos del asesor, e incluso logotipo o imágenes, debe tener un apartado de contenidos o indice, los cuales realmente son ligas o **enlaces a sus documentos .md**, *evite utilizar texto* para indicar enlaces internos o externo.
- Se propone una estructura tal como esta indicada abajo, sin embargo puede utilizarse cualquier otra que le apoye para organizar su repositorio.

```
| readme.md  
| | blog  
| | | C5.1_TituloActividad.md  
| | | C5.2_TituloActividad.md  
| | img  
| | docs  
| | | A5.1_TituloActividad.md  
| | | A5.2_TituloActividad.md
```



Desarrollo

1. Basado en las actividades referentes a actuadores, y protocolos de comunicación realice un Dashboard utilizando la interface Node-red y el NodeMCU ESP32; el cual permitirá a un cliente a través de su

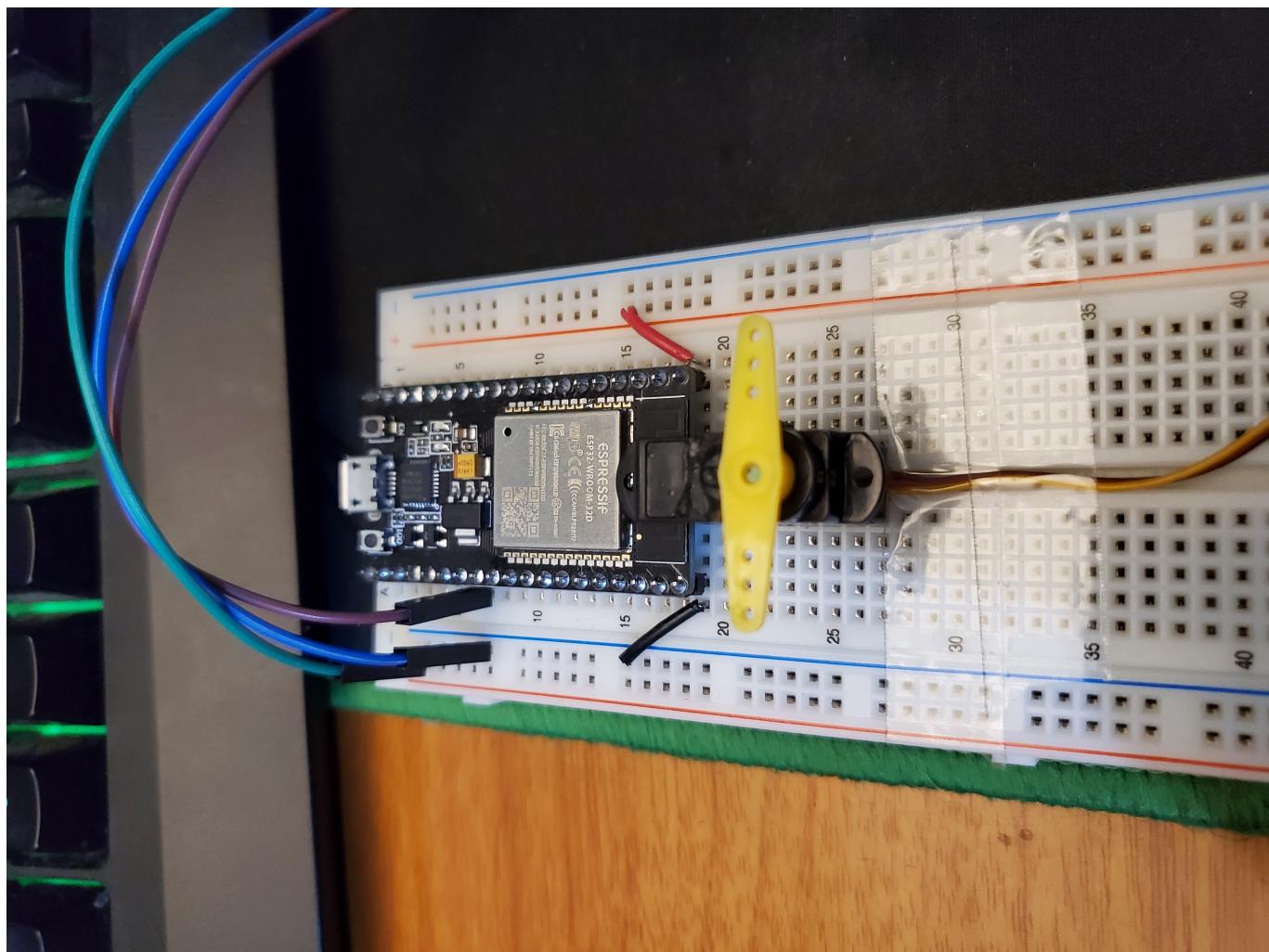
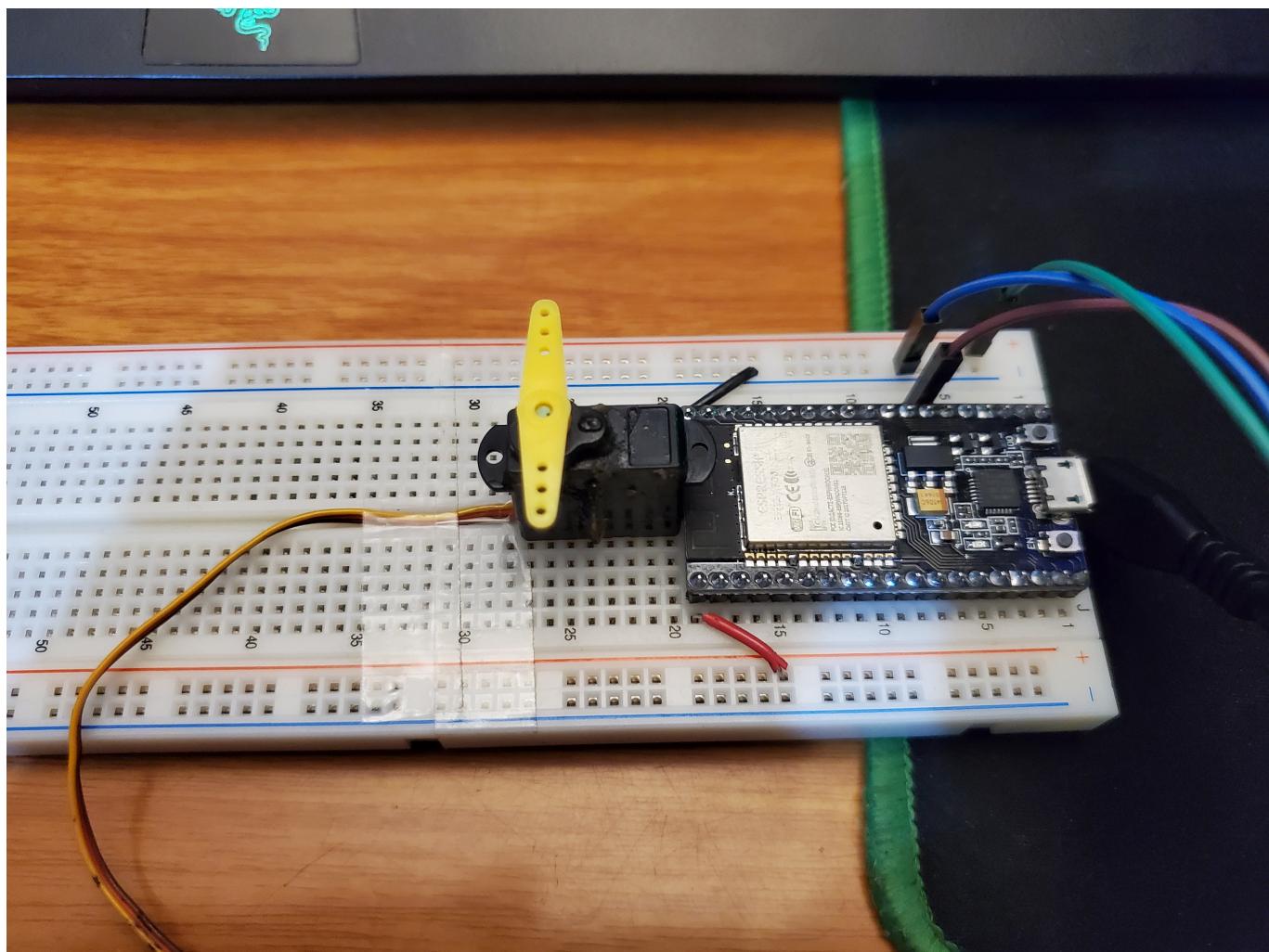
dispositivo móvil acceder a una dirección de un servidor Web local, y al ingresar al sitio deberá poder interactuar y visualizar el comportamiento de un actuador en tiempo real.

2. Para poder observar que la interface muestra el comportamiento del actuador deberá a este someterse a condiciones cambiantes a fin de observar esas variaciones en la interface.
3. Agregue el programa creado para lograr la condición solicitada.

```
#include <Servo.h> //libreria para el servomotor
//librerías para el broker
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <WiFi.h> //librería para conexión WIFI
const char* ssid = "INFINITUM2732_2.4"; //puntero que indica la SSID de la red
const char* password = "76cf6uNbpu"; //puntero que indica la contraseña de la red
Servo miServo; //objeto de tipo Servo
int pos = 0; //variable contador para indicar la posición del servomotor
//Aquí se define información necesaria para el broker
#define HOST          "io.adafruit.com"
#define PORT          1883
#define USERNAME      "EduardoMJ99"
#define PASSWORD      "aio_DZPC32ZKfzr8Y5XgK3yd0MzvqIgf"
WiFiClient client; //objeto de tipo WiFiClient
Adafruit_MQTT_Client mqtt(&client, HOST, PORT, USERNAME, PASSWORD);
Adafruit_MQTT_Publish pos_servo = Adafruit_MQTT_Publish(&mqtt, USERNAME
"/feeds/pos_servo");
void MQTT_connect(); //Realizamos la conexión con el broker
void setup() {
  WiFi.mode(WIFI_STA);
  Serial.begin(115200); //inicia comunicación serial
  Serial.println("Try Connecting to "); //mensaje que indica que se intenta comunicar a una red con la siguiente SSID
  Serial.println(ssid);
  // Connect to your wi-fi modem
  WiFi.begin(ssid, password); //inicia comunicación por WiFi
  while (WiFi.status() != WL_CONNECTED) { //mientras no haya comunicación por WiFi añade un delay
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected successfully"); //mensaje que indica que se conectó exitosamente y que se obtuvo una IP
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP()); //Muestra IP del ESP32
  miServo.attach(04); //asigna el pin 4 para el servo
}
void loop() {
  MQTT_connect(); //inicia la conexión al broker
  for (pos = 0; pos<=180; pos+=10) { //ciclo For que manipula la posición del
```

```
servomotor
    miServo.write(pos); //cambia el ángulo del servomotor mediante el
    contador pos
    pos_servo.publish(pos); //pública la información del ángulo actual
    delay(2000); //delay de 2 segundos debido a las limitaciones de la
    página
}
for(pos = 180; pos >= 0; pos -= 10) { //ciclo For que manipula la posición
    del servomotor para que realice una rotación inverso
    miServo.write(pos); //cambia el ángulo del servomotor mediante el
    contador pos
    pos_servo.publish(pos); //pública la información del ángulo actual
    delay(2000); //delay de 2 segundos debido a las limitaciones de la
    página
}
}
void MQTT_connect() { //Este es el método de conexión con el broker
int8_t ret;
if (mqtt.connected()) { //Mientras el broker esté conectado, el trabajo aquí
está hecho
    return;
}
//Cuando no está conectado, entonces iniciamos la conexión
Serial.print("Connecting to MQTT... ");
uint8_t retries = 3; //byte
//mqtt.connect() devuelve 0 se conecta
while ((ret = mqtt.connect()) != 0) { // En caso de que no se conecte
    //Mostramos el error por el cual no está conectado
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Retrying MQTT connection in 5 second... ");
    mqtt.disconnect(); //Desconectamos el broker
    delay(5000); //Esperamos 5 segundos
    retries--; //Disminuimos el valor del byte
    if (retries == 0){
        Serial.println("No Conectado"); //Desplegamos el mensaje de que no
se pudo establecer la conexión
        while(1);
    }
}
Serial.println("MQTT Connected!"); //Si se pudo restablecer la conexión
}
```

4. Agregue imágenes del circuito electrónico ensamblado.



5. Inserte aquí las imágenes que considere como evidencias para demostrar el resultado obtenido y las reuniones del equipo de trabajo.

ITN_SistemasProgramables

code_developers

Añadir un tema

Hilos de conversaciones

- Todos los mensajes directos
- Menciones y reacciones
- Más
- Canales

code_developers

general

varios

+ Añadir canales

Mensajes directos

- Slackbot
- EDUARDO MORGADO JA...
- CESAR ISAAC SOTO GAR...
- JESUS MANUEL COTA VIL...
- JUAN PABLO SANDOVAL ...
- Leonardo Enriquez
- Luis Alejandro Sanchez Gall...
- LUIS DIEGO FLORES GON...
- MARTIN HERNANDEZ Q...
- + Añadir compañeros de equi...

Aplicaciones

GitHub

+ Añadir aplicaciones

ELDEN HUMBERTO CRUZ VERA 20:15
me ire conectando a discord para terminar lo que falta de la practica

EDUARDO MORGADO JACOME 20:15
Va tambien yo

ABNER JESUS PERALES NIEBLA 20:15
Ahí les voy 😊

ABNER JESUS PERALES NIEBLA 20:30
Aqui está la página

<https://flows.nodered.org/node/node-red-dashboard>



Y aquí está el comando

```
npm i node-red-dashboard
```

EDUARDO MORGADO JACOME 20:31
Va perfecto ahora lo instalo

ELDEN HUMBERTO CRUZ VERA 20:35
<https://www.youtube.com/watch?v=yfwkHLDCoow>



Enviar mensaje a code_developers

B I ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹

DISCORD

Projects

sistemas-programables

CANALES DE TEXTO

- # general
- # musica

CANALES DE VOZ

- TEA
- # tea
- General
- Working

SISTEMAS PROGRAMABLES

sistemas-programables

- General
- Abner
- Eduardo Morgado
- Elden Cruz

TALLER DE INVESTIGACION II

- # taller-2
- General

Elden Cruz 09/01/2021
<https://www.youtube.com/watch?v=XI2i2LqHAew>

YouTube

David Castillo

ESP32 con Node-Red [TUTORIAL][ESPAÑOL][INGLES]



Eduardo Morgado ayer a las 20:36
https://github.com/DavidCastilloAlvarado/Node-red_server_and_ESP32_telemetry_accelerometer_Data/blob/master/No

GitHub

DavidCastilloAlvarado/Node-red_server_and_ESP32_telemetry_accelerom...

Contribute to DavidCastilloAlvarado/Node-red_server_and_ESP32_telemetry_accelerometer_Data



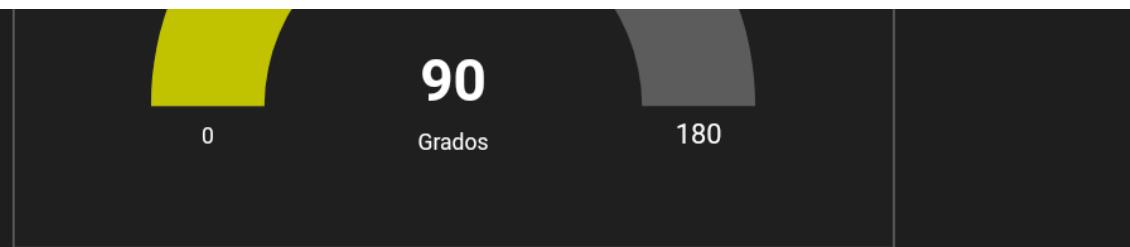
The screenshot shows a desktop setup with several open windows:

- GitHub Browser Tab:** A post by Abner titled "ESP8266 and Node-RED with MQTT | Random Nerd Tutorials". The post describes how to control ESP8266 outputs and display sensor data from Node-RED using MQTT.
- Discord Window:** Shows a server named "SISTEMAS PROGRAMABLES" with a channel "sistemas-programables" where "General" is selected. It also shows a video feed of a servo motor connected to an ESP8266 board.
- Node-RED Dashboard:** A flow titled "Flow 1" is shown, which includes nodes like range, template, delay, trigger, exec, and rbe, connected to an MQTT input node "EduardoMJ99/feeds/pos_servo" and a servo output node.
- Adafruit IO Feed:** A feed titled "IO - Feed: pos_servo" showing data points from January 12, 2021, at 1:46 PM. The data table is as follows:

Created at	Value	Location
2021/01/12 1:46:53PM	90	
2021/01/12 1:46:51PM	80	
2021/01/12 1:46:49PM	70	
2021/01/12 1:46:47PM	60	
2021/01/12 1:46:45PM	50	
2021/01/12 1:46:43PM	40	
2021/01/12 1:46:41PM	30	
2021/01/12 1:46:39PM	20	

The mobile device screen displays a servo motor control application with the following features:

- Top Bar:** Includes icons for messaging, star, location (IP address: 192.168.1.74), signal strength, battery level, and time (5:18).
- Header:** "Servomotor" and "Dashboard".
- Section Header:** "Posicion del servomotor".
- Control:** A large circular gauge indicating the servo position at 90 degrees.
- Bottom Footer:** "7 / 9".



 Rubrica

Criterios	Descripción	Puntaje
Instrucciones	Se cumple con cada uno de los puntos indicados dentro del apartado Instrucciones?	20
Desarrollo	Se respondió a cada uno de los puntos solicitados dentro del desarrollo de la actividad?	80

 [Ir a inicio](#) [Repositorio de Github de Morgado Jacome Eduardo](#)  [Repositorio de Github de Cruz Vera Elden Humberto](#)  [Repositorio de Github de Perales Niebla Abner Jesús](#) 