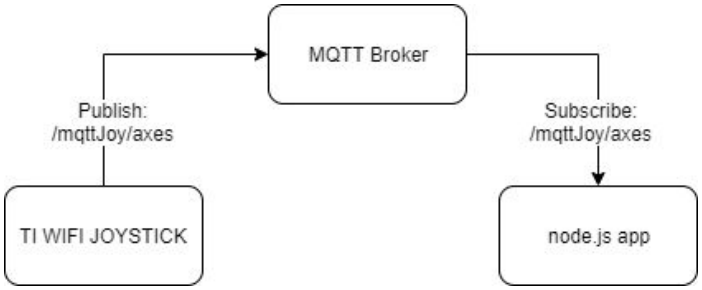
Informe de Actividades

Centro de Investigación, Desarrollo e Innovación en Ingeniería Mecatrónica

(https://www.imt.ucb.edu.bo/cidimec/)

Universidad Católica Boliviana "San Pablo" – Regional La Paz

Equipos empleados	MSP432P401R LaunchPad Texas Instruments BOOSTXL-EDUMKII Educational CC3100 Wi-Fi BoosterPack
Proyecto en el que los equipos fueron utilizados	Entrenamiento de usuarios para Interfaces Cerebro Computador empleando el paradigma de imaginación motora
Investigador encargado	Guillermo Sahonero Alvarez – Ingeniería Mecatrónica – Universidad Católica Boliviana
Estudiantes investigadores	Josmar Luis Suarez Loza – Ingeniería Mecatrónica – Universidad Católica Boliviana Anna Isabel Montevilla Shupikova – Ingeniería Mecatrónica – Universidad Católica Boliviana
Objetivo del proyecto	Proporcionar a los usuarios una interfaz intuitiva para el desarrollo de acciones motoras y su posterior imaginación sin la generación de artefactos motrices de alta potencia
Breve descripción del desarrollo	Los dispositivos empleados permiten la conexión intuitiva puesto que están desarrollados para posicionarse uno encima de otro, además, tanto el módulo BOOSTXL-EDUMKII como el CC3100 son módulos compatibles con el microcontrolador MSP432. En esencia, el módulo posee dos entradas analógicas correspondientes a la palanca de Joystick y dos entradas digitales para los botones. La lectura de ambas se realiza mediante las funciones *analogRead()* y *digitalRead()* respectivamente. Para la conexión Wifi se realizaron pruebas con el módulo CC3100. Se utilizaron los ejemplos de Energia relacionados a MQTT. Para el funcionamiento de este programa es necesario conocer el SSID de nuestra red, además de la contraseña, para conectarnos a Internet. Por otro lado, también se debe especificar la dirección del broker MQTT. En el procedimiento se utilizó un broker local mediante el uso de Mosquitto, por ende, la dirección del broker corresponde a la dirección IP del equipo que está corriendo Mosquitto. Una vez colocados estos datos en el programa se procede a realizar las publicaciones y subscripciones a cada tópico. En este caso, el programa solo necesita publicar datos y no requiere de una subscripción a tópicos. Finalmente, se utilizó el programa de prueba de Wifi como plantilla para realizar la combinación de ambos módulos. Solamente fue necesario añadir la lectura de los puertos

	analógicos y digitales correspondientes a la palanca y botones respectivamente. La información fue publicada a un puerto MQTT en forma de un archivo JSON.
Diagrama funcional del sistema desarrollado	Para la realización de este proyecto se siguió el siguiente esquema para la conexión.  Se utilizó un broker MQTT para el manejo de la información entre dispositivos. El Joystick se encarga de publicar en un tópico denominado /mqttJoy/axes la información referente al Joystick. Entre los datos enviados se tienen la posición en los ejes X y Y de la palanca de Joystick, y el estado de los dos botones que posee. En cuanto a la parte de la aplicación de Node se hizo uso de la librería mqtt, se conectó esta al mismo broker que el Joystick, y se realizó la subscripción al tópico /mqttJoy/axes.
Observaciones	Al momento de concluir el proyecto y probarlo se notó que existe un inconveniente de realizar la conexión simultanea de los módulos CC3100 y BOOSTXL-EDUMKII. Esto debido a que poseen un pin compartido, el cual corresponde al botón de hibernación del módulo Wifi y al botón de la palanca del módulo del joystick. Por este motivo resultaría posible que el usuario presione por error la palanca del joystick haciendo hibernar el módulo Wifi y por ende finalizando la conexión de este. Al momento, se investiga la forma de sobrellevar este inconveniente.
Código	#include <**SPI**.h> #include <WiFi.h> #include <**PubSubClient**.h> #include <aJSON.h> //--------------------------------------- //----------------Joystick----p------------- //--------------------------------------- // constants won't change. They're used here to

```
// set pin numbers:
const int joystickBtn2 = 32;//32;    // the number of the joystick select pin
const int joystickBtn1 = 33;
const int joystickX = 2;      // the number of the joystick X-axis analog
const int joystickY =  26;    // the number of the joystick Y-axis analog


// variables will change:
int joystickBtn2State = 0;     // variable for reading the joystick sel status
int joystickBtn1State = 0;
int joystickXState, joystickYState ;
//----------------------------------------
//------------WIFI and MQTT----------------
//----------------------------------------
// your network name also called SSID
//char ssid[] = "ZTE-eab738";
char ssid[] = "";    //Put your SSID here
// your network password
char password[] = ""; //Put network password here
// MQTTServer to use
char server[] = ""; //Use your IP direction if you are using a local broker,
or the broker direction if using a public broker

void callback(char* topic, byte* payload, unsigned int length) {
  //Do something if you are receiving an archive
}

WiFiClient wifiClient;
PubSubClient client(server, 1883, callback, wifiClient);

void setup()
{
  Serial.begin(115200);
  pinMode(joystickBtn2, INPUT_PULLUP);
  pinMode(joystickBtn1, INPUT_PULLUP);
  // Start Ethernet with the build in MAC Address
  // attempt to connect to Wifi network:
  Serial.print("Attempting to connect to Network named: ");
  // print the network name (SSID);
  Serial.println(ssid);
  // Connect to WPA/WPA2 network. Change this line if using open or
WEP network:
  WiFi.begin(ssid, password);
  while ( WiFi.status() != WL_CONNECTED) {
```

```
  // print dots while we wait to connect
  Serial.print(".");
  delay(300);
 }

 Serial.println("\nYou're connected to the network");
 Serial.println("Waiting for an ip address");

 while (WiFi.localIP() == INADDR_NONE) {
  // print dots while we wait for an ip addresss
  Serial.print(".");
  delay(300);
 }

 Serial.println("\nIP Address obtained");
 // We are connected and have an IP address.
 // Print the WiFi status.
 printWifiStatus();
}

void loop()
{
 // Reconnect if the connection was lost
 if (!client.connected()) {
  Serial.println("Disconnected. Reconnecting....");

  if(!client.connect("energiaClient")) {
   Serial.println("Connection failed");
  } else {
   Serial.println("Connection success");
   /*if(client.subscribe("inTopic")) { //Enable these lines to subscribe to a
topic
    Serial.println("Subscription successfull");
   }*/
  }
 }

 aJsonObject *msg = createMessage();
 char* myPayload = (char*)aJson.print(msg)+'\0';
 //----------------------------------------
 //--------------Joystick control----------
 //----------------------------------------
 // read the analog value of joystick x axis
 joystickXState = analogRead(joystickX);
```

```
  //Serial.print("Joystick X = ");
  //Serial.print(joystickXState);

  // read the analog value of joystick y axis
  joystickYState = analogRead(joystickY);


  // read the state of the joystick select button value:
  joystickBtn2State = digitalRead(joystickBtn2);
  joystickBtn1State = digitalRead(joystickBtn1);
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (joystickBtn2State == LOW) {
    Serial.println("   Joystick Center = pressed!");
  }
  if (joystickBtn1State == LOW) {
    Serial.println("   Btn1 = pressed!");
  }

  if(client.publish("/mqttJoy/axes",myPayload)) {
    //Serial.println("Publish success");
  } else {
    //Serial.println("Publish failed");
  }

  // Check if any message were received
  // on the topic we subsrcived to
  //Serial.println(myPayload);

    aJson.deleteItem(msg);
    free(myPayload);
  //client.poll();
  //Serial.println(availableMemory());
  delay(100);
}

aJsonObject *createMessage()
{
  aJsonObject *root = aJson.createObject();
  aJsonObject *d = aJson.createObject();
  aJson.addItemToObject(root, "d", d);
  aJson.addStringToObject(d, "myName", "Joystick");
  aJson.addNumberToObject(d, "rawX", joystickXState);
  aJson.addNumberToObject(d, "rawY", joystickYState);
```

```
  aJson.addNumberToObject(d, "Btn1", joystickBtn1State);
  aJson.addNumberToObject(d, "Btn2", joystickBtn2State);

  return root;
}

void printWifiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
int availableMemory() {
   // Use 1024 with ATmega168
   int size = 2048;
   byte *buf;
   while ((buf = (byte *) malloc(--size)) == NULL);
      free(buf);
   return size;
}
```