// Include Libraries for ESP-NOW Communications

#include <esp\_now.h>

#include <WiFi.h>

// Define Joystick Connections (some joystick models reverse X & Y)

#define X\_AXIS\_PIN 33

#define Y\_AXIS\_PIN 32

#define SWITCH\_PIN 27

// MAC Address of responder - edit as required

uint8\_t broadcastAddress[] = { 0xE8, 0x31, 0xCD, 0xE6, 0x31, 0x90 };

// Define a data structure for received data

typedef struct struct\_message\_rcv {

  bool motorMode;

  int mecanumMode;

} struct\_message\_rcv;

// Create a structured object for sent data

typedef struct struct\_message\_xmit {

  int xAxis;

  int yAxis;

  bool pbSwitch;

} struct\_message\_xmit;

// Create a structured object for sent data

struct\_message\_xmit xmitData;

// ESP-NOW Peer info

esp\_now\_peer\_info\_t peerInfo;

// Variable for Motor Mode

volatile byte motorModeValue = B00000000;

// Variable for Mecanum Mode

volatile int mecanumModeValue = 0;

// Variables for Joystick values

int joyXaxis = 127;

int joyYaxis = 127;

// Variable for Joystick pushbutton state

bool joySwitchState = HIGH;

// Variable for connection error  - HIGH is error state

volatile bool connectError = LOW;

// Variable for connection status string

String connectStatus = "NO INFO";

void setup() {

  // Set up Serial Monitor

  Serial.begin(115200);

  // Set joystick pin as input with Pullup

  pinMode(SWITCH\_PIN, INPUT\_PULLUP);

  // Set ESP32 as a Wi-Fi Station

  WiFi.mode(WIFI\_STA);

  // Disable WiFi Sleep mode

  WiFi.setSleep(false);

  // Initilize ESP-NOW

  if (esp\_now\_init() != ESP\_OK) {

    connectStatus = "ESP-NOW Error";

    connectError = HIGH;

    return;

  }

  else {

    connectStatus = "ESP-NOW OK";

    connectError = LOW;

  }

  // Register the send callback

  esp\_now\_register\_send\_cb(OnDataSent);

  // Register peer

  memcpy(peerInfo.peer\_addr, broadcastAddress, 6);

  peerInfo.channel = 0;

  peerInfo.encrypt = false;

  // Add peer

  if (esp\_now\_add\_peer(&peerInfo) != ESP\_OK) {

    connectStatus = "No peer added";

    connectError = HIGH;

    return;

  }

  else {

    connectStatus = "ESP-NOW Ready";

    connectError = LOW;

  }

  // Enter the Loop with connectError set HIGH to avoid intial display flicker

  connectError = HIGH;

}

void loop() {

  // Check connection status

  if (connectError == LOW) {

    // Get joystick values and convert them

    joyXaxis = analogRead(X\_AXIS\_PIN);

    joyYaxis = analogRead(Y\_AXIS\_PIN);

  }

  else {

    // Send "zero" values as joystick data

    joyXaxis = 127;

    joyYaxis = 127;

  }

  // Check and set switch status

  if (digitalRead(SWITCH\_PIN) == LOW) {

    // Switch was pressed

    joySwitchState = true;

  }

  else {

    joySwitchState = false;

  }

  // Format structured data

  xmitData.xAxis = joyXaxis;

  xmitData.yAxis = joyYaxis;

  xmitData.pbSwitch = joySwitchState;

  // Send message via ESP-NOW

  esp\_err\_t result = esp\_now\_send(broadcastAddress, (uint8\_t \*)&xmitData, sizeof(xmitData));

  // If switch was pressed delay longer to debounce

  if (joySwitchState == true) {

    delay(200);

  } else {

    delay(50);

  }

}