#include <Servo.h>

#include <Arduino\_FreeRTOS.h>

#include <LiquidCrystal\_I2C.h>

#define configTOTAL\_HEAP\_SIZE ((size\_t)(1024))

const int potenciometroPinMotor = A0; // Pin analógico conectado al potenciómetro

const int potenciometroPinServo = A2;

const int fotoPin = 2;

const int pinServo = 3;

const int pinButOnOff = 12;

const int pinButOn1 = 13;

const int pinButOn2 = A1;

const int motorPinStart = 4;

const int motorPinEnd = 11;

const int IR\_PIN = 13;

const bool ButOnOff = false;

volatile bool ButOn1 = false;

volatile bool ButOn2 = false;

volatile int revoluciones = 0;

volatile int revoluciones2 = 0;

volatile bool ModeAutoEnabled = false;

volatile float rpm = 0;

volatile float rpm2 = 0;

unsigned long tiempoAnterior = 0;

unsigned long tiempoAnterior2 = 0;

volatile int lastButtonSpeed = 0;

volatile int velocidades1\_2 = 0;

volatile int velocidad\_variable=0;

TaskHandle\_t displayControlTaskHandle;

TaskHandle\_t ControlTaskHandle;

// Objetos de las bibliotecas

Servo miServo;

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Dirección 0x27, pantalla de 16x2

void setup() {

miServo.attach(pinServo);

pinMode(potenciometroPinMotor, INPUT);

pinMode(potenciometroPinServo, INPUT);

pinMode(pinButOnOff, INPUT\_PULLUP);

pinMode(pinButOn1, INPUT\_PULLUP);

pinMode(pinButOn2, INPUT\_PULLUP);

pinMode(fotoPin, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(fotoPin), contarRevolucion, FALLING);

for (int pin = motorPinStart; pin <= motorPinEnd; ++pin) {

pinMode(pin, OUTPUT);

}

Serial.begin(9600);

lcd.init(); // Inicializa la pantalla LCD

lcd.backlight(); // Enciende la retroiluminación

xTaskCreate(controlTask, "Motor y Servo Control", 128, NULL, 1, &ControlTaskHandle);

xTaskCreate(displayControlTask, "Display Control", 128, NULL, 1, &displayControlTaskHandle);

Serial.println("Creamos los hilos!");

vTaskStartScheduler();

Serial.println("Scheduler no iniciado!");

}

void loop() {

// El bucle principal no debe hacer nada, todo se maneja en los hilos.

}

void controlTask(void \*pvParameters) {

while (true) {

int valorPotenciometroServo = analogRead(potenciometroPinServo);

int valorPotenciometroMotor = analogRead(potenciometroPinMotor);

if (ModeAutoEnabled) {

int velocidadMotor = lastButtonSpeed;

setMotorSpeed(velocidadMotor);

}else if(ButOn1){

int velocidadMotor = velocidades1\_2;

setMotorSpeed(velocidadMotor);

}else if(ButOn2){

int velocidadMotor = velocidades1\_2;

setMotorSpeed(velocidadMotor);

} else {

int velocidadMotor = map(valorPotenciometroMotor, 0, 1023, 0, 255);

lastButtonSpeed = velocidadMotor;

setMotorSpeedSimple(velocidadMotor);

}

servoControlTask(valorPotenciometroServo);

//vTaskDelay(pdMS\_TO\_TICKS(100));

}

}

int anngulo\_prev = 0;

void servoControlTask(int potValue) {

//int angulo = map(potValue, 0, 1023, 0, 180);

int angulo = map(potValue, 0, 1023, 140, 180);

if (abs(angulo - anngulo\_prev) > 2) {

miServo.write(angulo);

anngulo\_prev = angulo;

}

}

void setMotorSpeed(int speed) {

int searched\_rpm = speed \* 10; //velocidad a la que quremos llegar

if (millis() - tiempoAnterior2 >= 1000) {

rpm2 = (revoluciones2 / 2.0) \* 60.0;

revoluciones2 = 0;

tiempoAnterior2 = millis();

if (rpm2 < searched\_rpm ) {

Serial.println("Aumentamos velocidad");

lcd.setCursor(0, 3);

lcd.print("Aumentamos vel");

//int dif = (searched\_rpm - rpm2) / 10;

if (searched\_rpm- rpm2 > 100) {

velocidad\_variable = velocidad\_variable + 100;

}else{

velocidad\_variable = velocidad\_variable + 10;

}

Serial.println("velocidad\_variable: " + String(velocidad\_variable));

Serial.println("searched\_rpm: " + String(searched\_rpm));

Serial.println("rpm2: " + String(rpm2));

setMotorSpeedSimple(velocidad\_variable/10);

} else if (rpm2 > searched\_rpm ) {

Serial.println("Reducimos velocidad");

lcd.setCursor(0, 3);

lcd.print("Reducimos vel ");

//int dif = (rpm2 - searched\_rpm) / 10;

if (rpm2-searched\_rpm > 100) {

velocidad\_variable = velocidad\_variable - 100;

}else{

velocidad\_variable = velocidad\_variable - 10;

}

Serial.println("velocidad\_variable: " + String(velocidad\_variable));

Serial.println("searched\_rpm: " + String(searched\_rpm));

Serial.println("rpm2: " + String(rpm2));

setMotorSpeedSimple(velocidad\_variable/10);

} else {

Serial.println("Control Conseguido");

lcd.setCursor(0, 3);

lcd.print("Conseguido");

}

}

}

void setMotorSpeedSimple(int speed) {

int binaryArray[8];

for (int i = 0; i < 8; i++) {

binaryArray[i] = (speed >> i) & 1;

}

int pos = 0;

for (int pin = motorPinStart; pin <= motorPinEnd; ++pin) {

digitalWrite(pin, binaryArray[pos] == 1 ? HIGH : LOW);

pos++;

}

}

void displayControlTask(void \*pvParameters) {

while (true) {

bool buttonOnOff = digitalRead(pinButOnOff) == LOW; // Botón presionado

bool button1 = digitalRead(pinButOn1) == LOW; // Botón presionado

bool button2 = digitalRead(pinButOn2) == LOW; // Botón presionado

if (buttonOnOff) {

ModeAutoEnabled = !ModeAutoEnabled; // Cambiar el modo automático

velocidad\_variable=rpm;

ButOn1 = false;

ButOn2 = false;

vTaskDelay(pdMS\_TO\_TICKS(200)); // Debounce delay

}else if(button1){

Serial.println("OPAAA1");

ButOn1 = !ButOn1;

velocidades1\_2 = 20;

ModeAutoEnabled = false;

ButOn2 = false;

vTaskDelay(pdMS\_TO\_TICKS(200)); // Debounce delay

}else if(button2){

Serial.println("OPAAA2");

ButOn2 = !ButOn2;

velocidades1\_2 = 100;

ModeAutoEnabled = false;

ButOn1 = false;

vTaskDelay(pdMS\_TO\_TICKS(200)); // Debounce delay

}

if (millis() - tiempoAnterior >= 1000) {

rpm = (revoluciones / 2.0) \* 60.0;

revoluciones = 0;

lcd.setCursor(0, 0);

lcd.print("RPM: ");

lcd.print(rpm);

lcd.print(" "); // Añade espacios en blanco para borrar dígitos previos

tiempoAnterior = millis();

}

lcd.setCursor(0, 1);

if (ModeAutoEnabled) {

lcd.print("Cruise speed ON ");

} else {

lcd.print("Cruise speed OFF");

}

vTaskDelay(pdMS\_TO\_TICKS(100)); // Retardo entre verificaciones del control remoto

}

}

void contarRevolucion() {

revoluciones++;

revoluciones2++;

}