#include <Arduino\_FreeRTOS.h>

#include <time.h>

#pragma GCC optimize ("03")

#include <Servo.h>

#include <LiquidCrystal.h>

LiquidCrystal lcd(7,6,5,4,3,2);

Servo myservo;

int SensorData; // Sensors

int ledred = 26; //RED LED is connected at pin26

int ledgreen = 25; //GREEN LED is connected at pin25

int ledaqua = 44; //BLUE LED is connected at pin44

int buzzer1 = 29; //BUZZER is connected at pin29

int servo = 23; //Servo Motor is connected at pin23

long duration;

int distance;

int pos = 0; // Position is set to value 0

int ldr = A0; //LDR is connected at analog pin 0

int val = 0; //initial value of ldr is set to 0

#define trigPin1 27

#define echoPin1 28

#define trigPin2 31

#define echoPin2 32

TaskHandle\_t TaskSensorFrontH; //Task handler for Task of Ultrasonic

TaskHandle\_t TaskRedH; //Task handler for Task of Red LED

TaskHandle\_t TaskGreenH; //Task handler for Task of Green LED

TaskHandle\_t TaskAquaH; //Task handler for Task of LDR

void setup()

{

Serial.begin(9600);

pinMode(ledaqua, OUTPUT);

myservo.attach(23);

lcd.begin(16, 2);

lcd.clear();

//Each Task's pvTaskCode, pcName, usStackDepth, pvParameters, uxPriority, pxCreatedTask

xTaskCreate(TaskSensor, "Sensor", 100, NULL, 1, NULL);

xTaskCreate(TaskAqua, "LDR", 100, NULL, 2, &TaskAquaH);

xTaskCreate(TaskRed, "Red", 100, NULL, 0, &TaskRedH);

xTaskCreate(TaskGreen, "Green", 100, NULL, 0, &TaskGreenH);

}

void loop()

{

}

void TaskAqua(void \*pvParameters) //Task with the highest priority is run

{

while(1)

{

//Serial.println("Loop");

val = analogRead(ldr); //analog read value of ldr value is passed to val.

Serial.print("LDR"); //show the word "LDR" at LCD screen

Serial.println(val); //show the word val at LCD screen

if(val <500){ //if the received light intensity of ldr is less than 500

digitalWrite(44, HIGH); //turn on the buzzer

vTaskDelay( 400 / portTICK\_PERIOD\_MS ); //delay 400ms

//delay(400);

}

else{

digitalWrite(44, LOW); //turn on the buzzer

vTaskDelay( 500 / portTICK\_PERIOD\_MS ); //delay 500ms

//delay(300);

}

//vTaskDelete(NULL);

}

}

void TaskSensor(void \*pvParameters)

{

pinMode(trigPin1, OUTPUT); //trigPin1 of front sensor as the output

pinMode(echoPin1, INPUT); //echoPin1 of front sensor as the input

pinMode(buzzer1, OUTPUT); //buzzer as the output

pinMode(trigPin2, OUTPUT); //trigPin2 of reverse sensor as the output

pinMode(echoPin2, INPUT); //echoPin2 of reverse sensor as the input

while(1)

{

digitalWrite(trigPin2, LOW); //\_\_\_\_REVERSE SENSOR\_\_\_//

delayMicroseconds(2); //trigPin2 low for 2 ms

digitalWrite(trigPin2, HIGH);

delayMicroseconds(10); //trigPin2 high for 10 ms

digitalWrite(trigPin2, LOW);

duration = pulseIn(echoPin2, HIGH); //store the time in duration variable

distance = duration \* 0.034 / 2; //formula to calculate the distance between object

Serial.print("DisBack: "); //print out the results to the Serial Monitor

Serial.print(distance);

Serial.println(" cm");

lcd.setCursor(0,2);

lcd.print("DisBack: ");

lcd.print(distance);

lcd.print(" cm ");

delay (500);

if(distance <20){

tone(buzzer1,1000); //buzzer will turn on when distance <20cm

}

else{

noTone(buzzer1); //buzzer will turn off if distance >=20cm

}

delay(1000);

digitalWrite(trigPin1, LOW); //\_\_\_\_FRONT SENSOR\_\_\_\_//

delayMicroseconds(2); //trigPin1 low for 2 ms

digitalWrite(trigPin1, HIGH);

delayMicroseconds(10); //trigPin1 high for 10 ms

digitalWrite(trigPin1, LOW);

duration = pulseIn(echoPin1, HIGH); //store the time in duration variable

distance = duration \* 0.034 / 2; //formula to calculate the distance between object

Serial.print("DisFront: "); //print out the results to the Serial Monitor

Serial.print(distance);

Serial.println(" cm");

lcd.setCursor(0,0);

lcd.print("DisFront: ");

lcd.print(distance);

lcd.print(" cm ");

delay (500);

if(distance <=10){

vTaskResume(TaskRedH); //TaskRed resumes when distance <=10cm(red led on)

vTaskSuspend(TaskGreenH); //TaskGreen suspends (green led off)

for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 90 degrees in steps of 1 degree

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15 ms for the servo to reach the position

}

for (pos = 90; pos >= 0; pos -= 1) { // goes from 90 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15 ms for the servo to reach the position

}

//vTaskResume(TaskAquanH);

}

else{

vTaskResume(TaskGreenH); //Task Green resumes when distance >10 cm(green led on)

vTaskSuspend(TaskRedH); //Task Red suspends (red led off)

}

vTaskDelay( 500 / portTICK\_PERIOD\_MS );

}

}

void TaskRed(void \*pvParameters)

{

while(1)

{

digitalWrite(ledred, HIGH); //red led will turn on

digitalWrite(ledgreen, LOW); //green led will turn off

tone(buzzer1, 500); //buzzer turn on

vTaskDelay( 500 / portTICK\_PERIOD\_MS );

}

}

void TaskGreen(void \*pvParameters)

{

while(1)

{

digitalWrite(ledred, LOW); //red led turn off

digitalWrite(ledgreen, HIGH); //green led turn on

noTone(buzzer1); //buzzer turn off

vTaskDelay( 500 / portTICK\_PERIOD\_MS );

}

}