University of Northampton

الجـامعـة الخـليجـية
GULF UNIVERSITY

College of Engineering

Department: Electrical and Electronic Engineering

Course Title: 24/25 - Microprocessor Systems

Course Code: CSY2015-PGU-P1

Submission Date: 9-Dec-2024

UON Student ID: 23855995

Student Name: Mohamed Jaafar Yaqoob

Instructor Name: Dr. Osama Al-Rawi

PROJECT TITLE: PARKING ASSISTANT

1. YOUTUBE VIDEO:

https://youtu.be/hBc1rl63zqE?si=zo8k7QocP0Zgt0BN

2. INTRODUCTION

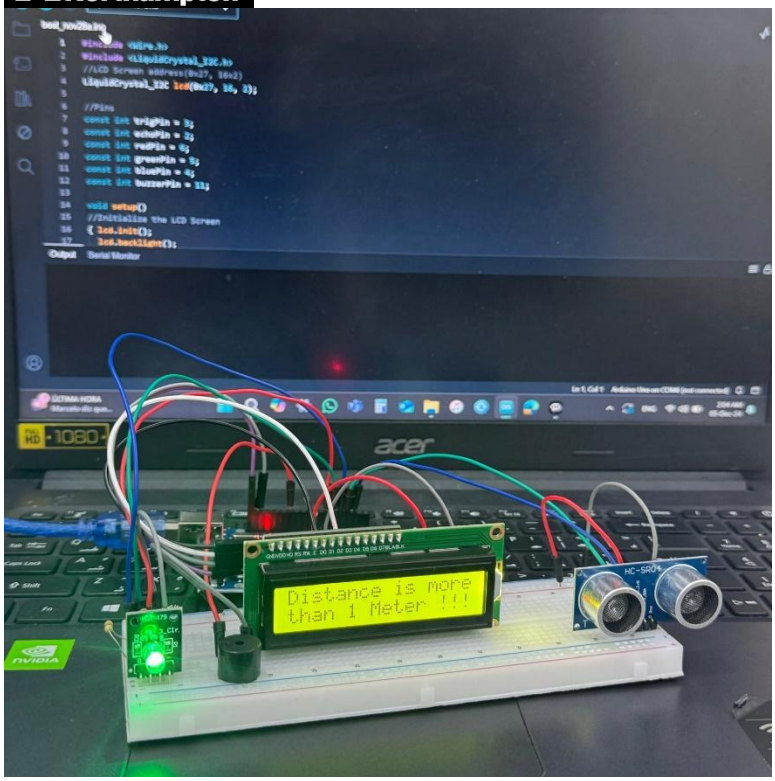
A parking sensor is a device used in vehicles to assist drivers in parking their vehicles in tight spaces safely by avoiding collisions. It can do that by alerting them from obstacles that they may not see in their rearview mirrors or through the rear window. In general these sensors are commonly mounted on the front and rear bumpers of a vehicle and use ultrasonic sensors to detect objects close to the vehicle such as walls, poles, or other vehicles that are in close proximity to the vehicle.

3. OBJECTIVES:

The purpose of the parking assistant is to enhance safety, prevent accidents, and make parking easier and more comfortable for drivers.

4. LIST OF COMPONENTS:

1. Arduino Uno (The Brain of the project)
2. Breadboard (for building the project)
3. Jumper Wires (to connect the components)
4. Ultrasonic Sensor (to detect objects)
5. LCD Screen (to display the distance)
6. RGB LED (to visibly alert)
7. Buzzer (to audibly alert)



5. PROJECT EXPLANATION:

In this project I have used Three methods to alert the driver, which are an RGB LED Light, a Buzzer, and additionally I have used an LCD Screen that displays the exact distance between the vehicle and the object. When an object is detected, the Ultrasonic sensor will send a message to Arduino Uno, so that it commands the RGB LED and Buzzer. Firstly, the RGB LED will start to change its color from green to red as the distance between the vehicle and the object decreases, and when the distance decreases to 10cm the buzzer will turn on.

6. CONCLUSION:

As a conclusion, we can understand that this project can increase drivers safety by alerting them to obstacles that may not be visible while parking. Furthermore, by providing the exact distance of objects around the vehicle, this project will help the driver to park easier than only using visual alerts, and audible beeps.

7. REFERENCES:

- https://docs.arduino.cc/
- https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/
- https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

8. CODE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
//LCD Screen address(0x27, 16x2)
LiquidCrystal_I2C lcd(0x27, 16, 2);

//Pins
const int trigPin = 3; //Ultrasonic Sensonr(Trig)
const int echoPin = 2; //Ultrasonic Sensonr(Echo)
const int redPin = 6; //RGB LED Light(Red)
const int greenPin = 5; //RGB LED Light(Green)
const int bluePin = 4; //RGB LED Light(Blue)
const int buzzerPin = 11; //Buzzer

void setup()
//Initialize the LCD Screen
{ lcd.init(); //Initialize the LCD Screen
  lcd.backlight(); //Turn on the LCD backlight
  lcd.clear(); //Clear the LCD Screen
  Serial.begin(9600);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);}

void loop() {
  //distance Calculation in cm
  long duration, distance_cm;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance_cm = duration * 0.034 / 2; //the Formula for calculating the distance in
cm

//Display the Distance on the LCD Screen
if (distance_cm >= 100 || distance_cm <= 1){
  lcd.setCursor(0, 0); //LCD Screen column & row numbers
  lcd.print("Distance is more"); //Print on the first row of the LCD Screen
  lcd.setCursor(0, 1); //LCD Screen column & row numbers
```

```
  lcd.print("than 1 Meter !!!");} //Print on the second row of the LCD Screen
else {
  lcd.setCursor(0, 0); //LCD Screen column & row numbers
  lcd.print("Distance : "); //Print on the first row of the LCD Screen
  lcd.print(distance_cm); //Print on the first row of the LCD Screen
  lcd.println(" cm "); //Print on the first row of the LCD Screen
  lcd.setCursor(0, 1); //LCD Screen column & row numbers
  lcd.print(" !!! Beware !!! ");} //Print on the second row of the LCD Screen

  //Change RGB LED color based on Distance
  int greenValue = map(distance_cm, 0, 100, 0, 255); //Changing the Green color
  int redValue = map(distance_cm, 0, 100, 255, 0); //Changing the Red color
  analogWrite(redPin, redValue);
  analogWrite(greenPin, greenValue);
  analogWrite(bluePin, 0);

  //Turn on the buzzer if Distance is 10cm
  if (distance_cm <= 10) {tone(buzzerPin, 500);} //Buzzer turn on
  else {noTone(buzzerPin);} //Buzzer turn off

  delay(500);}
```

PROJECT TITLE: SECURITY FLOODLIGHT

1. YOUTUBE VIDEO:

https://youtu.be/e78lK5-xA9s?si=oqawmZn6UM8q0RXv

2. INTRODUCTION

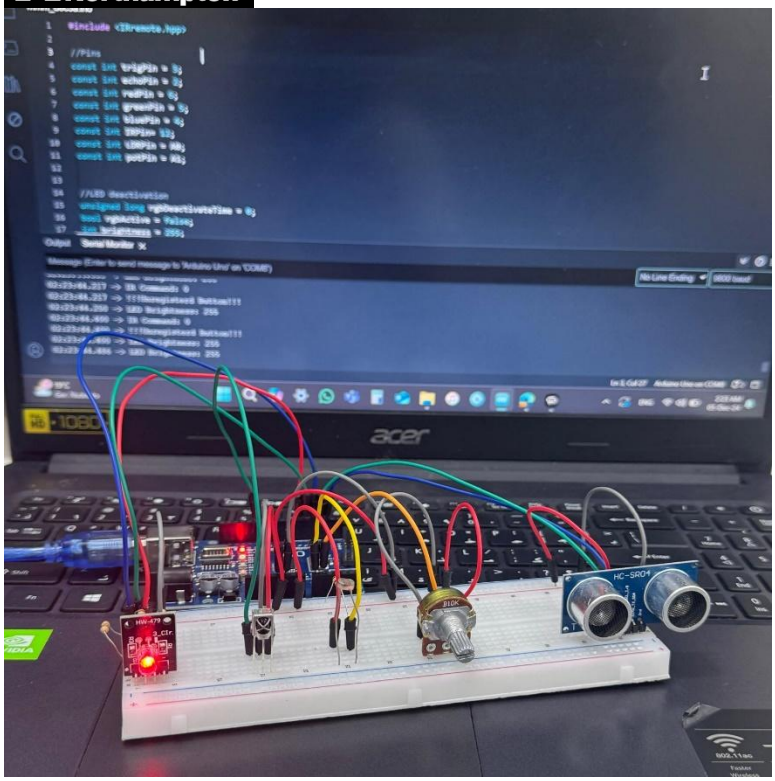
A home security floodlight is a type of outdoor lighting that is used for improving the security of a home. These floodlights are commonly installed near entry points, driveways, or other entry areas around the property. In a modern home these floodlights mostly come with advanced features such as, light level sensors, motion sensors, and the ability to be controlled remotely. Some models also include built-in cameras, two-way audio communication, and sirens for enhanced security.

3. OBJECTIVES:

The purpose of Security Floodlight project is to enhance security with a smart and customizable lighting.

4. LIST OF COMPONENTS:

1. Arduino Uno (The Brain of the project)
2. Breadboard (for building the project)
3. Jumper Wires (to connect the components)
4. Ultrasonic Sensor (to detect objects)
5. RGB LED (to display different colors)
6. IR Remote (to control the LED color)
7. IR Receiver (to receive commands from the remote)
8. LDR Sensor (to detect light level)
9. Potentiometer (to control the LED brightness)



5. PROJECT EXPLANATION:

This project is a Security Floodlight that offers customizable lighting as security alerts. In this project I have used three devices as inputs that turn the LED on, each programed to turn on the LED with a different color so that each is distinguished. Those devices are the LDR Sensor, Ultrasonic Sensor, and IR Receiver. Firstly, the LED will turn on with a green color for 2 to 4 seconds when it senses 10 as a threshold darkness. Secondly, the LED will turn on with a red color for 15 seconds when it detects motion within 1 meter. Finally, the LED can be controlled using a remote that sends commands to the IR receiver, which also allows you to choose previous or next color, and cycle through colors slowly or fastly. Additionally, you can control the brightness of the LED using a potentiometer.

6. CONCLUSION:

As a conclusion, we can observe that this project can increase home security by combining advanced sensing technologies with customizable lighting effects. It can do that by merging features such as daylight sensing, motion detection, and even remote controlling.

7. REFERENCES:

- https://docs.arduino.cc/
- https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/
- https://roboticsbackend.com/arduino-ir-remote-controller-tutorial-setup-and-map-buttons/
- https://roboticsbackend.com/arduino-potentiometer-complete-tutorial/

8. CODE:

```
#include <IRremote.hpp>

//Pins
const int trigPin = 3; //Ultrasonic Sensonr(Trig)
const int echoPin = 2; //Ultrasonic Sensonr(Echo)
const int redPin = 6; //RGB LED Light(Red)
const int greenPin = 5; //RGB LED Light(Green)
const int bluePin = 4; //RGB LED Light(Blue)
const int IRPin= 12; //IR Reciever
const int LDRPin = A0; //LDR Sensor
const int potPin = A1; //Potentiometer

//Variables
unsigned long LED_Off_TIME = 0;
bool LED = false;
unsigned long IR_Time = 0;
bool IR = false;
int currentColor = 0;

//LED RGB Color Codes
int colors[][3] = {
  {255, 0, 0}, //Red Color
  {0, 255, 0}, //Green Color
  {0, 0, 255}, //Blue Color
  {255, 255, 255}}; //White Color

// IR codes
const uint32_t IR_SLOW = 69; //Button(CH-)
const uint32_t IR_FAST = 71; //Button(CH+)
const uint32_t IR_PREV = 68; //Button(Prev)
const uint32_t IR_NEXT = 64; //Button(Next)
const uint32_t IR_RED = 12; //Button(1)
const uint32_t IR_GREEN = 24; //Button(2)
const uint32_t IR_BLUE = 94; //Button(3)
const uint32_t IR_WHITE = 8; //Button(4)

void setup() {
  //Initializing the System
  Serial.begin(9600);
  IrReceiver.begin(IRPin, ENABLE_LED_FEEDBACK);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LDRPin, INPUT);
  pinMode(potPin, INPUT);
```

```
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  setRGBColor(0, 0, 0);
  Serial.println("Security Floodlight System");} //Print on the Serial Monitor

void loop() {
  //IR commands
  if (IrReceiver.decode()) {
    handleIRCode(IrReceiver.decodedIRData.command);
    IrReceiver.resume();}
  //Deactivate IR commands
  if (IR && millis() > IR_Time) {
    IR = false;
    Serial.println("Returning to sensors control."); //Print on the Serial Monitor
    setRGBColor(0, 0, 0);}

  //Handle
  handlePotentiometer();
  handleUltrasonic();
  handleLDR();

  //LED OFF after time ends
  if (LED && millis() > LED_Off_TIME) {
    LED = false; //LED state
    setRGBColor(0, 0, 0);}
  delay(100);}

//IR remote
void handleIRCode(uint32_t command) {
  Serial.print("IR Command: ");
  Serial.println(command);
  //Activate(IR)
  IR = true;
  switch (command) {
    //Turn LED ON(Cycle colors slowly)
    case IR_SLOW:
      cycleColors(2000); //RGB LED turn on time(2s per each color)
      break;
    //Turn LED ON(Cycle colors fastly)
    case IR_FAST:
      cycleColors(500); //RGB LED turn on time(0.5s per each color)
      break;
    //Turn LED ON(Red)
    case IR_RED:
      Serial.println("LED Red Color."); //Print on the Serial Monitor
```

```
      setRGBColor(colors[0][0], colors[0][1], colors[0][2]);
      IR_Time = millis() + 3000; //RGB LED turn on time(3s)
      break;
    //Turn LED ON(Green)
    case IR_GREEN:
      Serial.println("LED Green Color."); //Print on the Serial Monitor
      setRGBColor(colors[1][0], colors[1][1], colors[1][2]);
      IR_Time = millis() + 3000; //RGB LED turn on time(3s)
      break;
    //Turn LED ON(Blue)
    case IR_BLUE:
      Serial.println("LED Blue Color."); //Print on the Serial Monitor
      setRGBColor(colors[2][0], colors[2][1], colors[2][2]);
      IR_Time = millis() + 3000; //RGB LED turn on time(3s)
      break;
    //Turn LED ON(White)
    case IR_WHITE:
      Serial.println("LED White Color."); //Print on the Serial Monitor
      setRGBColor(colors[3][0], colors[3][1], colors[3][2]);
      IR_Time = millis() + 3000; //RGB LED turn on time(3s)
      break;
    //Turn LED ON(Previous)
    case IR_PREV:
      currentColor = (currentColor - 1 + 4) % 4;
      Serial.println("Previous LED Color."); //Print on the Serial Monitor
      setRGBColor(colors[currentColor][0], colors[currentColor][1],
colors[currentColor][2]);
      IR_Time = millis() + 3000; //RGB LED turn on time(3s)
      break;
    //Turn LED ON(Next)
    case IR_NEXT:
      currentColor = (currentColor + 1) % 4;
      Serial.println("Next LED Color."); //Print on the Serial Monitor
      setRGBColor(colors[currentColor][0], colors[currentColor][1],
colors[currentColor][2]);
      IR_Time = millis() + 3000; //RGB LED turn on time(3s)
      break;
    //Other Buttons
    default:
      Serial.println("!!!Unregisterd Button!!!"); //Print on the Serial Monitor
      break;}}

//LDR(LED Green color)
void handleLDR() {
  int lightLevel = analogRead(LDRPin);
  if (lightLevel < 10 && !LED) {
```

```
    Serial.println("!!!Darkness is detected!!!"); //Print on the Serial Monitor
    LED = true; //Turn on the LED
    LED_Off_TIME = millis() + random(2000, 4000); //RGB LED turn on time(2s to 4s)
    setRGBColor(0 , 255 , 0);}} //setting the RGB LED color

//Potentiometer(LED Brightness)
void handlePotentiometer() {
  int potValue = analogRead(potPin);
  int brightness = 255;
  brightness = map(potValue, 0, 1023, 0, 255); //Potentiometer map for LED
brightness
  Serial.print("LED Brightness: "); //Print on the Serial Monitor
  Serial.println(brightness);} //Print on the Serial Monitor

//Ultrasonic(LED Red color)
void handleUltrasonic() {
long duration, distance_cm;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance_cm = duration * 0.034 / 2; //the Formula for calculating the distance in
cm
  if (distance_cm > 0 && distance_cm <= 100 && !LED) { //Setting the range to 1m
    Serial.println("!!!Motion is detected!!!"); //Print on the Serial Monitor
    LED = true;
    LED_Off_TIME = millis() + 15000; //RGB LED turn on time(15s)
    setRGBColor(255 , 0 , 0);}} //setting the RGB LED color

//Set LED RGB Color
void setRGBColor(int red, int green, int blue) {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);}

//Cycle through LED colors
void cycleColors(unsigned long delayTime) {
  for (int i = 0; i < 4; i++) {
    setRGBColor(colors[i][0], colors[i][1], colors[i][2]);
    delay(delayTime);}}
```