University of Northampton

Name: Abdelrahman Nassar

Student ID: 23856013

Assignment Title: Microprocessor-Based System

MODULE: CSY2015

Module Tutor: Dr. Osama Alrawi

2024/2025

Lis of Contents

1. Speed Radar (Part 1)

A video demonstration of the system application is available on
https://youtu.be/XrcUuoNhQuk

1.1. Introduction

This project represents a real-world problem that checking vehicles speed. Since over speed is the main cause of cars accidents, it helps monitor vehicles' speed and provide visual and audio alarms for different speed ranges. The project goes with **SDG 3** because it potentially can reduce vehicles accidents by enhancing road safety which leads to saving lives. The data collected for the project can be integrated into AI to predict the zones with high risks of over speed accidents.

1.2. Objective

The objective of this project is to provide a microcontroller-based system using Arduino to monitor vehicles' speed. We use Arduino to measure the speed when a vehicle passes through two IR sensors, then it shows the speed of the vehicle and provides a visual and audio indication whether the vehicle is in safe speed or over speed. As the project is combatable with IoT, it could also be integrated into smart traffic management systems in order to contribute to more sustainable urban environments.

1.3. List of Components

1. Arduino Uno.
2. I2C 16x2 Liquid Crystal Display.
3. Two IR Obstacle Avoidance sensors.
4. RGB LED.
5. Buzzer.
6. One 200 Ω Resistor.
7. Breadboard.
8. Connecting wires.
9. USB Power Source.

1.4. List of Software

1. Arduino IDE.
2. Serial Monitor.

1.5.Theory

The project calculates the speed of a vehicle by dividing the distance between the two IR Sensors over the time required to pass from the first to the second IR Sensor, the calucaltion can be measured using this equation:

$$Speed = \frac{Distance}{Time}$$

Speed Categories are:

- Safe speed (≤ 70 km/h): Green, No Buzzer.
- Warning (71–100 km/h) Yellow, No Buzzer.
- Over speed (> 100 km/h): Red, Buzzer On.

1.6.Procedure

1.6.1. Circuit Setup:

Connect IR sensors to pins 2 and 3.

Connect RGB LED to pins 4, 5, and 6.

Connect the buzzer to pin 12.

Connect the LCD to the I2C pins (A4 for SDA, A5 for SCL).

All components are connected to the power source and grounded correctly.

1.6.2. Coding:

Initialize all components using setup function.

Reading IR sensors in the loop function in continuous way.

Calculate time and speed when a vehicle passes through both IR sensors.

Send alerts using RGB LED and buzzer based on speed category.

Display the speed and alert on the LCD.

1.7.Code:

```
/*
****************************** Speed Detection System ************************
-- Author: Abdelrahman Nassar
-- Date: 08/12/2024
```

```
-- Purpose: To measure the speed of a passing vehicle using two IR sensors and calculate its speed.
--          Display the speed on an LCD and indicate safe, warning, or overspeeding conditions using an RGB LED and a buzzer.
-- Requirements:
   - Two IR sensors positioned at a fixed distance.
   - An RGB LED for status indication.
   - A buzzer for alerting overspeeding.
   - An LCD display for speed readout.
   - Assumes sensors are mounted securely with a 10 cm gap (0.10 meters).
-- Modifications: None
********************************************************************************
*/

// Include necessary libraries
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Define the LCD screen (I2C address, columns, rows)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Fixed distance between the two sensors (in meters)
float distance = 0.10; // 10 cm = 0.10 meters

// Define IR sensors and buzzer pins
int ir_s1 = 2; // IR sensor 1 pin
int ir_s2 = 3; // IR sensor 2 pin
int buzzer = 12; // Buzzer pin

// Variables for time measurement
int timer1 = 0, timer2 = 0;
float Time;
float speed;

// Flags to track sensor states
int flag1 = 0, flag2 = 0;

// Define RGB LED pins
#define redPin 6
#define greenPin 5
#define bluePin 4
```

```
/*
************************************************************
********** Function: setup **********
-- Date: 08/12/2024
-- Purpose: Initializes all components including pins for IR sensors, buzzer, RGB LED, and LCD.
-- Called by: System startup.
-- Modifications: None
************************************************************
************************************************************
*/
void setup() {
  // Initialize pins
  pinMode(ir_s1, INPUT);
  pinMode(ir_s2, INPUT);
  pinMode(buzzer, OUTPUT);

  // Initialize RGB LED pins
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
}

/*
************************************************************
********** Function: loop **********
-- Date: 08/12/2024
-- Purpose: Continuously monitors IR sensors to detect a passing vehicle.
        Calculates the speed, updates the LCD display, controls RGB LED and buzzer based on speed conditions.
-- Called by: Main program execution.
-- Modifications: None
************************************************************
************************************************************
*/
void loop() {
  // Detect at Sensor 1
  if (digitalRead(ir_s1) == LOW && flag1 == 0) {
    timer1 = millis(); // Record the timestamp
```

```
  flag1 = 1;         // Mark Sensor 1 as triggered
}

// Detect at Sensor 2
if (digitalRead(ir_s2) == LOW && flag2 == 0) {
  timer2 = millis(); // Record the timestamp
  flag2 = 1;         // Mark Sensor 2 as triggered
}

// Calculate speed when both sensors have detected
if (flag1 == 1 && flag2 == 1) {
  // Calculate time difference in seconds
  Time = abs(timer1 - timer2) / 1000.0;

  // Calculate speed in km/h
  speed = (Time > 0) ? (distance / Time) * 3.6: 0;

  // Display speed or waiting message
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Speed: ");
  lcd.print(speed, 1);
  lcd.print(" Km/Hr");

  lcd.setCursor(0, 1);
  if (speed <= 70) { // Safe Speed
    lcd.print("  Safe Speed    ");
    setColor(0, 255, 0); // Green (Safe speed)
    digitalWrite(buzzer, LOW); // Turn off buzzer
  } else if (speed > 70 && speed <= 100) { // Warning Speed
    lcd.print("  Ok Speed      ");
    setColor(255, 255, 0); // Yellow (Warning)
    digitalWrite(buzzer, LOW); // Turn off buzzer
  } else { // Overspeeding
    lcd.print("  Over Speeding ");
    setColor(255, 0, 0); // Red (Over speed)
    digitalWrite(buzzer, HIGH); // Turn on buzzer
  }

  delay(3000); // Wait before resetting

  // Reset for the next car detection
```

```
    speed = 0;
    flag1 = 0;
    flag2 = 0;
  } else if (speed == 0) { // No detection
    lcd.setCursor(0, 1);
    lcd.print("No car detected ");
  }
}

/*
**********************************************************
********** Function: setColor **********
-- Date: 08/12/2024
-- Purpose: Sets the RGB LED color based on the intensity of red, green, and blue components (0-255).
-- Parameters:
    - redIntensity: Intensity of the red component.
    - greenIntensity: Intensity of the green component.
    - blueIntensity: Intensity of the blue component.
-- Called by: loop function.
-- Modifications: None
**********************************************************
**********************************************************
*/
void setColor(int redIntensity, int greenIntensity, int blueIntensity) {
  // Set accurate color mix by adjusting red, green, and blue intensities
  analogWrite(redPin, redIntensity);   // Red intensity (0-255)
  analogWrite(greenPin, greenIntensity); // Green intensity (0-255)
  analogWrite(bluePin, blueIntensity);  // Blue intensity (0-255)
}
```

1.8.Results

Waiting for a vehicle to pass through IR sensors: LCD displays "No car detected".

When a vehicle passes:

- Speed is calculated and displayed on the LCD.
- RGB color changes based on speed:
  - Green when the vehicle is over speed (< 70 km/h).
  - Yellow when the vehicle is in warning speed (70> speed < 100 km/h).
  - Red when the vehicle is over speed (> 100 km/h).
  - Buzzer sounds when the vehicle is over speed (> 100 km/h).

1.9. Discussion

The project calculates the vehicle's speed in a smooth way and gives the correct response (visual and audio). The project assumed that the vehicles always pass the IR sensors perpendicularly and only one care ate time. The project can be enhanced by adding log data to store the measured speeds. Also, an IR remote can be added to change the speeds categories.

1.10. Conclusion

The project is a low cost and effective way to monitor vehicle speeds and provide alarms based on the vehicle speeds categories. It is effective and can be trusted and can be considered as a base prototype to be enhanced in real-world applications.

2. RGB Floodlight (Part 2)

A video demonstration of the system application is available on
https://youtu.be/pV9g5ZEiGKI

2.1. Introduction

This project represents a design for a smart RGB floodlight system. It senses the ambient light as well as any motion integrated with a feature that can be controlled by a user by an IR remote control to customize lighting effects and set the sensitivity of the ambient light as well as the detected motion distance. The project goes with **SDG 7** as well as **SDG 11** since it offers adjustable energy usage and enhances home security by using smart technology.

2.2. Objective

The objective of this project is to provide a microcontroller-based system using Arduino to build a prototype of home security floodlight system. The system works based on light intensity and motion detection. Also, the user should be able to control lighting effects, colors, brightness, and operational settings.

2.3. List of Components

1. Arduino Uno.
2. RGB LED.
3. Light-dependent resistor (LDR).
4. Ultrasonic sensor (HC-SR04).
5. Infrared receiver.
6. IR remote control.
7. One 10 kΩ Resistor.
8. Three 200 Ω Resistor.
9. Connecting wires.
10. Breadboard.
11. USB Power Source.

2.4. List of Software

1. Arduino IDE.
2. Serial Monitor.

2.5. Theory

LDR senses the intensity of ambient light which will provide input to the microcontroller to set day and night status. Ultrasonic sensor detected any nearby objects then providing a signal for RGB LED to turn on for 15 seconds. RGB LED produce light in different colors. The IR remote control gives the user the ability to control operational settings and adjust brightness as well as selecting lighting effects.

2.6. Procedure

2.6.1. Circuit Setup:

Connect the RGB LED to pins on the Arduino with three 200 Ω Resistors.

Connect the LDR to pins with 10 kΩ Resistor.

Connect the ultrasonic sensor's Trig and Echo pins.

Connect the IR receiver to pin on the Arduino.

All components are connected to the power source and grounded correctly.

2.6.2. Coding:

Initialize all components using setup function.

Reading LDR sensor in the loop function in continuous way to detect Day/night.

Reading motion detection using an ultrasonic sensor in continuous way.

Remote-controlled lighting effects and manual operational settings.

2.7.Code

```
/*--
**********RGB Floodlight System**********
-- Author: Abdelrahman Nassar
-- Date: 08/12/2024
- Purpose: This program implements a smart lighting system using sensors and an infrared remote control.
- Features:
  - Detects ambient light and adjusts LED brightness and color.
  - Uses ultrasonic sensor to detect proximity and activates lighting effects.
  - Infrared remote control for manual operation and settings adjustment.
- Requirements:
  - Components: LDR, Ultrasonic Sensor, RGB LEDs, Infrared Receiver, Remote Control.
  - Ensure the connections match the defined pins.
- Output:
  - RGB LED control for various lighting effects.
  - Serial output for system logs and current status.
- Assumptions:
  - The program assumes the sensors are correctly calibrated.
  - Serial monitor at 9600 baud for debugging.
********************************************************************************
--*/
#include <IRremote.h> // Remote control library
// Define pins
#define LDR_PIN A0   #define TRIG_PIN 11    // Light sensor , // Ultrasonic sensor (Trig)
   #define ECHO_PIN 12       #define RED_PIN 5       // Ultrasonic sensor (Echo) ,  // Red LED
    #define GREEN_PIN 6  #define BLUE_PIN 10       // Green LED  , // Blue LED

            #define IR_PIN 2             // Infrared receiver
// Control variables
int lightThreshold = 5;      // Default light threshold
```

```
      int distanceThreshold = 15;   // Default minimum distance
         int motionTime = 15000;        // Light duration when motion is detected (in milliseconds)
          bool manualMode = false, ON_OFF = true ,night = true, detectM = true;
     int brightness = 255;          // Default brightness level
           int currentEffect = 0, color = -1;       // Current effect (0: no effect)
/*--
**********************************************************
- Function: setup
- Date: 08/12/2024
- Purpose: Initialize the system components and serial communication.
- Called by: System startup
**********************************************************
*/

void setup() {
 pinMode(LDR_PIN, INPUT);    pinMode(TRIG_PIN, OUTPUT);    pinMode(ECHO_PIN, INPUT);

          pinMode(RED_PIN, OUTPUT);  pinMode(GREEN_PIN, OUTPUT);  pinMode(BLUE_PIN, OUTPUT);

                IrReceiver.begin(IR_PIN); // Start the infrared receiver

                                    Serial.begin(9600);
}


/*--
**********************************************************
- Function: readUltrasonicDistance
- Date: 08/12/2024
- Purpose: Read distance using the ultrasonic sensor.
- Returns: Distance in centimeters.
- Called by: loop
**********************************************************
*/
long readUltrasonicDistance() {
   digitalWrite(TRIG_PIN, LOW);   delayMicroseconds(2);
       digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
            digitalWrite(TRIG_PIN, LOW);
               return pulseIn(ECHO_PIN, HIGH) * 0.034 / 2; // Convert time to distance (cm)
}
```

```
/*--
**********************************************************
- Function: setColor
- Date: 08/12/2024
- Purpose: Control the RGB LED color and brightness.
- Parameters:
  - r: Red intensity (0-255).
  - g: Green intensity (0-255).
  - b: Blue intensity (0-255).
- Called by: loop, jumpEffect, strobeEffect, gradualEffect, smoothEffect
**********************************************************
*/
void setColor(int r, int g, int b) {
  analogWrite(RED_PIN, (r * brightness) / 255);
         analogWrite(GREEN_PIN, (g * brightness) / 255);
               analogWrite(BLUE_PIN, (b * brightness) / 255);
}

/*--
**********************************************************
- Function: jumpEffect
- Date: 08/12/2024
- Purpose: Cycle through predefined colors with a delay between transitions.
- Called by: loop
**********************************************************
*/
void jumpEffect() {
        int colors[3][3] = {{255, 0, 0}, {0, 255, 0}, {0, 0, 255}};
             for (int i = 0; i < 3; i++) {
    s              etColor(colors[i][0], colors[i][1], colors[i][2]);
                        delay(300);
  }
**********************************************************
- Function: strobeEffect
- Date: 08/12/2024
- Purpose: Create a strobe effect by toggling white light on and off.
- Called by: loop
**********************************************************
*/
}
// Stroboscopic effect
void strobeEffect() {
```

```
    for (int i = 0; i < 5; i++) {
       setColor(255, 255, 255);
            delay(100);
                setColor(0, 0, 0);
                      delay(100);
 }
}
/*--
************************************************************
- Function: gradualEffect
- Date: 08/12/2024
- Purpose: Transition colors gradually between red, green, and blue.
- Called by: loop
************************************************************
*/
void gradualEffect() {
    for (int i = 0; i <= 255; i++) {
        setColor(i, 255 - i, 0);
              delay(10);
 }
 for (int i = 0; i <= 255; i++) {
      setColor(0, i, 255 - i);
            delay(10);
 }

          setColor(0, 0, 0);

}

/*--
************************************************************
- Function: smoothEffect
- Date: 08/12/2024
- Purpose: Create a smooth RGB transition effect.
- Called by: loop
************************************************************
*/
                    void smoothEffect() {
              for (int i = 0; i <= 255; i++) {
        setColor(i, 255 - i, i / 2);
     delay(20);
 }
```

```
}
/*--
************************************************************
- Function: loop
- Date: 08/12/2024
- Purpose: Main program logic for light control and interaction.
- Called by: System runtime
************************************************************
*/
  void loop() {
       // Read light level
          int lightLevel = analogRead(LDR_PIN);
  long distance = readUltrasonicDistance();
            if (lightLevel < lightThreshold && night){
                      setColor(255, 0, 0);
           if (night){
     Serial.println("night- Activated");}
           delay (2550);
               setColor(0, 0, 0);
       night =false;}
          if (lightThreshold < lightLevel ){
   if (!night){
          Serial.println("Day- Deactivated");}night =true;}

  // If manual mode is enabled
      if (manualMode && ON_OFF) {           // Turn off the light
           setColor(255, 255, 255); // Turn on the light in white color
  }
    else {
     // Check ambient light, motion, and distance
   if (distance > 0 && distance <= distanceThreshold && !night) {
                      setColor(0, 255, 0); // Turn on the light in white color
               Serial.println("Object motion detected and light ON");
     delay(motionTime);
                  setColor(0, 0, 0);       // Turn off the light
   }
  }

  /*
************************************************************
** Project: Remote-Controlled Multi-Function Lighting System
** Description:
```

```
  - This program integrates sensors, LEDs, and a remote control to create
    a dynamic lighting system. The system reacts to ambient light, proximity,
    and remote inputs to adjust lighting color, brightness, and effects.
** Features:
  - Supports automatic and manual modes.
  - Light intensity and effects based on environmental conditions.
  - Infrared remote control for manual adjustments.
** Requirements:
  - Arduino board.
  - IR receiver, Ultrasonic sensor, LDR, RGB LEDs.
** Instructions:
  1. Compile and upload the code to an Arduino.
  2. Connect the components as per the pin configurations.
  3. Operate using the remote or sensors as required.
** Outputs:
  - Dynamic RGB LED colors and effects.
  - Serial monitor logs for debugging.
** Assumptions:
  - LDR is used to detect light levels (night/day).
  - Ultrasonic sensor detects proximity for triggering lights.
  - IR remote uses predefined codes.
************************************************************
*/
  if (IrReceiver.decode()) {
          switch (IrReceiver.decodedIRData.decodedRawData) {
              case 0xE916FF00: // White
    color = 1;
          currentEffect = 0;
     Serial.println("White Selected");
              break;
    case 0xF30CFF00: // Red
             color = 2;
    currentEffect = 0;
     Serial.println("Red Selected");
              break;
    case 0xE718FF00: // Green
            color = 3;
    currentEffect = 0;
     Serial.println("Green Selected");
     break;
           case 0xA15EFF00: // Blue
    color = 4;   currentEffect = 0;
```

```
  Serial.println("Blue Selected");
          break;
case 0xF708FF00: // Jump effect
           color = -1;
  currentEffect = 1;
  Serial.println("Jump Effect");
           break;
case 0xE31CFF00: // Strobe effect
  color = -1;
  currentEffect = 2;   Serial.println("Strobe Effect");

           break;
case 0xA55AFF00: // Gradual effect
                color = -1;
  currentEffect = 3;
           Serial.println("Gradual Effect");
  break;
case 0xBD42FF00: // Smooth effect
  color = -1;   currentEffect = 4;     Serial.println("Smooth Effect");

  break;
case 0xEA15FF00: // Increase brightness
  brightness += 25;  if (brightness > 255) brightness = 255;

  Serial.print("Brightness: ");
        Serial.println(brightness);
   break;
case 0xF807FF00: // Decrease brightness
                    brightness -= 25;
  if (brightness < 0) brightness = 0;
              Serial.print("Brightness: ");
  Serial.println(brightness);
            break;
case 0xB946FF00: // Manual mode toggle button
      manualMode = !manualMode;
             ON_OFF = true;
               if (manualMode == false){color = 0 ; currentEffect = 0;}
  //Serial.println(manualMode ? "Light ON" : "Light OFF");
            if (manualMode) {
        Serial.println("Manual Mode Light ON");
                color = 1;// Turn on the White light
                    currentEffect = 0;
```

```
} else {
 Serial.println("Automatic Mode Light ON");
          setColor(0, 0, 0); // Turn off the light
 }

  break;
case 0xB847FF00 : // Increase light threshold
                        lightThreshold += 5;
  if (lightThreshold > 1023) lightThreshold = 1023;
                  Serial.print("Light Threshold: ");
  Serial.println(lightThreshold);
  break;
case 0xBA45FF00 : // Decrease light threshold
       lightThreshold -= 5;
  if (lightThreshold < 0) lightThreshold = 0;
               Serial.print("Light Threshold: ");
  Serial.println(lightThreshold);
          break;
case 0xBF40FF00: // Increase distance
               distanceThreshold += 5;
          if (distanceThreshold > 200) distanceThreshold = 200;
                           Serial.print("Distance Threshold: ");
  Serial.println(distanceThreshold);
              break;
case 0xBB44FF00: // Decrease distance
          distanceThreshold -= 5;
            if (distanceThreshold < 10) distanceThreshold = 10;
  Serial.print("Distance Threshold: ");
              Serial.println(distanceThreshold);
  break;
     case 0xF20DFF00: // Increase Lighting duration
                                motionTime += 1000;
                Serial.print("Lighting duration: ");
  Serial.print(motionTime/1000);
               Serial.println(" s ");


                        break;
case 0xE619FF00: // Decrease Lighting duration
                    motionTime -= 1000;
  if (motionTime < 1000) lightThreshold = 1000;
  Serial.print("Lighting duration: ");
```

```
        Serial.print(motionTime/1000);
                        Serial.println(" s ");
                    break;
    case 0xF609FF00: // Reset to default settings
      lightThreshold = 500;
          distanceThreshold = 100;
       currentEffect = 0;
            motionTime = 15000;
      Serial.println("Settings Reset to Default");
            break;
  }
          IrReceiver.resume(); // Resume signal reception
}
if (0 < color || currentEffect != 0) {
                    manualMode = true;
  ON_OFF = false;
}

// Execute the current effect
switch (currentEffect) {
       case 1: jumpEffect(); break;
            case 2: strobeEffect(); break;
  case 3: gradualEffect(); break;
          case 4: smoothEffect(); break;
}
switch (color) {
          case 0:  setColor(0, 0, 0); break;
  case 1:  setColor(255, 255, 255); break;
              case 2:  setColor(255, 0, 0); break;
  case 3:  setColor(0, 255, 0); break;
              case 4:  setColor(0, 0, 255); break;
 }
}
```

2.8.Results

- The floodlight activates in low-light conditions by turning ON the RGB for 2 seconds and starts responding to motion and starts detecting object by turning on the RGB for 15 seconds.
- The floodlight deactivated in high-light conditions.

- The users are able to adjust settings using the IR remote control including brightness, selecting different colors for RGB, effect modes, and activation thresholds for both LDR and Ultrasonic sensors.
- Lighting effects perform as intended, and manual mode allows full user control.
- The RGB turns on and off based on motion and light intensity.
- User inputs via the IR remote are accurately decoded and executed.

2.9. Discussion

The project effectively senses the ambient light intensity and active/deactivate the system. Ultrasonic sensor is able to detect motions within the preset distance threshold and send signals to RGB to turn on for the preset time duration. Users are fully capable of changing the RGB colors, selecting different colors effects set the threshold of LDR light intensity, change the RGB time duration when motion detection, and change the threshold of Ultrasonic distance.
There are some limitations of the project such as when a motion is detected, and the light turn on for 15 seconds (adjustable) the system freezes and cannot respond to the IR remote until the duration of the timer ends.

2.10. Conclusion

The project is a low cost and effective way to be used as a smart RGB floodlight system. The prototype is suitable for home security applications that can offer customizable features for users.

References

- Arduino website, https://www.arduino.cc/.
- TheBackShed.com - Forum. TheBackShed. https://www.thebackshed.com/forum/ViewTopic.php?TID=7170.
- Arduino - Infrared Obstacle Avoidance Sensor | Arduino getting started. https://arduinogetstarted.com/tutorials/arduino-infrared-obstacle-avoidance-sensor.