Course Code: CSY2015

Student Name: Sharifa Al Jowder

UON ID: 23856007

Module Tutor: Dr. Osama Al Rawi

Assignment title: Microprocessor-Based System

Student Name: Sharifa Al Jowder

UON ID: 23856007

Table of Contents

Table of Figures

Student Name: Sharifa Al Jowder

UON ID: 23856007

Part 1 - Solar panel tracking System using Arduino UNO

YouTube link - SOLAR PANEL TRACKING YOUTUBE LINK

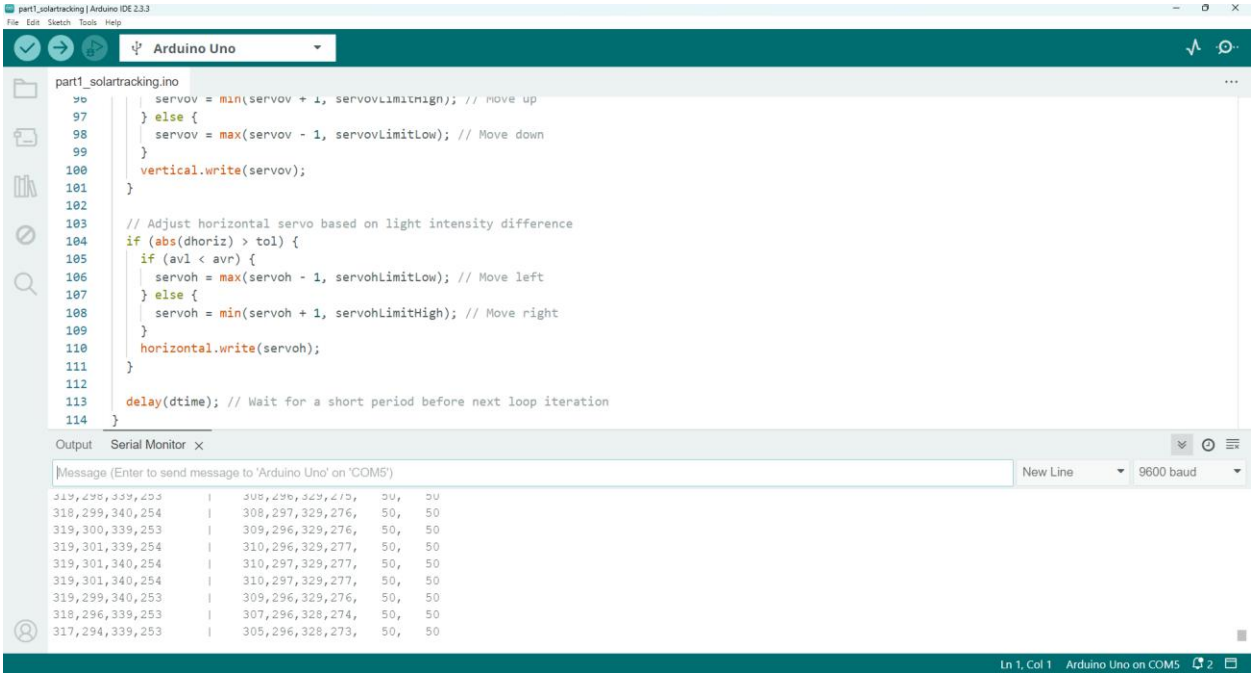

Figure 1 Snapshot of the Last edited information for the solar panel tracking system

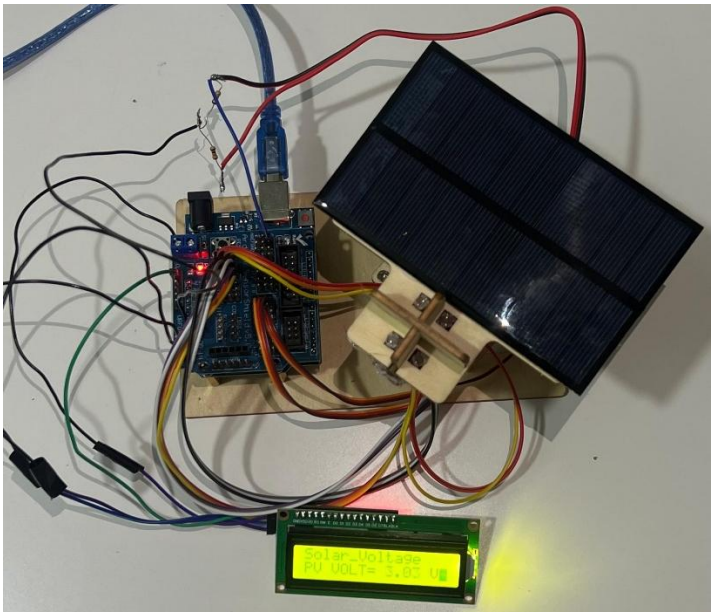

Figure 2 Image of the final design for the solar panel tracking system

Student Name: Sharifa Al Jowder

UON ID: 23856007

Introduction

Solar energy plays a vital role in renewable energy and sustainable development by reducing dependence on fossil fuels. Traditional fixed solar panels, however, lack efficiency as they cannot track the sun's movement throughout the day, leading to decreased energy output.

This project addresses the issue by creating a solar panel tracking system powered by an Arduino Uno microcontroller. Photoresistor modules detect light intensity from various angles, while servo motors adjust the panel's orientation to ensure maximum sunlight exposure. Real-time voltage readings are displayed on an I2C LCD, offering instant performance insights.

By integrating electronics, programming, and mechanical components, this energy-efficient and cost-effective solution supports SDG 7: Affordable and Clean Energy and SDG 13: Climate Action. Enhancing solar panel efficiency contributes to advancing renewable energy technology and tackling key environmental challenges.

Advantages of the Solar Panel Tracking System Using Arduino UNO

1. **Increased Efficiency in Energy Use**
   Throughout the day, the solar tracking system makes sure that the solar panels stay in line with the sun's position. In comparison to fixed solar panel systems, this optimizes energy capture and greatly increases the total power output.

2. **Economical Resolution**
   This system offers a low-cost substitute for conventional sun tracking systems without sacrificing functionality by leveraging inexpensive parts like light-dependent resistors (LDRs) and the Arduino UNO.

3. **Capability of Hybrid Energy**
   The technology can be modified to combine solar, wind, and wave energy in hybrid energy configurations. Consistent energy output is guaranteed by this integration, which makes it appropriate for a variety of environmental circumstances.

4. **Artificial Intelligence (AI) Integration**
   The technology can forecast sunshine patterns based on historical data or weather conditions when AI algorithms are added. This makes it possible to proactively move the panel, guaranteeing effective energy production even in overcast or dimly lit environments
   .
5. **Adaptability and Scalability**
   The system's modular design makes it possible to scale it to fit a range of uses, from large-scale industrial operations to small-scale home installations. Its flexible architecture allows for future additions, like more sensors or improved software.

Student Name: Sharifa Al Jowder

UON ID: 23856007

6. **Precision and Automation**
   Reliability and operating effort are increased by the servo motors and Arduino UNO, which provide precise and automated panel adjustments without the need for human interaction.

7. **Adaptability for Upcoming Enhancements**
   Because of the system's great degree of customization, AI and machine learning technologies can be included. Self-learning capabilities for optimizing solar panel movements based on real-time data analysis may be made possible by this.

8. **Value in Education and Research**
   This project is a great teaching and research tool for automation, AI integration, and renewable energy. For professionals and students interested in these sectors, it offers an opportunity for experiential learning.

9. **Monitoring in Real Time**
   An LCD makes it easier to monitor voltage in real time, giving users the ability to keep tabs on system performance. Furthermore, remote access and control may be possible with optional IoT connectivity.

10. **Eco-Friendly**
    The system encourages the use of renewable energy, lessens reliance on fossil fuels, and contributes to a sustainable and environmentally friendly energy solution by optimizing solar energy use.

This solar panel tracking system provides a creative and effective method of using renewable energy by fusing automation, artificial intelligence, and hybrid energy compatibility.

Key Aspects of the Problem

1. **Inefficient Energy Capture:** Fixed systems suffer from substantial energy losses due to their inability to track the sun effectively, limiting their overall performance.
2. **Shifting Sunlight:** The sun’s position shifts continuously throughout the day, making it challenging for stationary solar panels to remain perfectly aligned, resulting in decreased energy absorption.
3. **Sustainability Goals:** Enhancing the efficiency of solar panels is essential to strengthening renewable energy's role in addressing climate change.
4. **Adaptability Issues:** Many existing solar tracking technologies are costly and overly complicated, making them unsuitable for smaller or residential setups.
5. **Performance Monitoring:** Many systems lack real-time feedback, hindering proper maintenance and optimization and impacting long-term reliability.

Student Name: Sharifa Al Jowder

UON ID: 23856007

Issue and Resolution:

1- Limited efficiency in harnessing solar energy.
Approach: Implementing a solar tracking mechanism that continuously adjusts the panel's position to capture maximum sunlight, enhancing energy production.

2- Absence of immediate performance monitoring.
Approach: Incorporating an I2C LCD to show real-time voltage data, allowing users to track and improve the system's effectiveness.

Challenges and Resolutions:

Durability and Energy Efficiency of Servo Motors: Mitigate wear and reduce power consumption by using robust motors, implementing efficient control strategies, and activating low-power modes during periods of limited sunlight.

Enhanced Data Visualization: Improve system monitoring through the integration of touch interfaces, mobile applications, and wireless technologies such as Wi-Fi or Bluetooth for more advanced analysis and management.

Relation to SDGs	
	Enhances solar power efficiency, making renewable energy more accessible.
	Reduces reliance on fossil fuels, supporting global climate change initiatives.

Code Functionality:

Four photoresistors (LDRs), two servo motors, an LCD display, and a potentiometer are used in this code to build a sun tracking system. To identify the direction of the brightest light, the LDRs measure light intensity at various points (upper-left, upper-right, lower-left, and lower-

right). To align a solar panel or other device with the light source, the system uses these readings to compute vertical and horizontal variations in light intensity and modifies the angles of the vertical and horizontal servos. The servos' movement is limited to predetermined angle limitations to avoid over-rotation. Solar panel voltage is simulated by a potentiometer, computed, and shown on the LCD with the constant label "Solar_Voltage." For effective light tracking, the system is tuned with sensitivity and speed parameters ({tol` and `dtime`), and it has a serial output for tracking and adjusting sensor readings. This configuration is a working prototype for optimizing the capture of solar energy.

Libraries, Components (Hardware/ software) & Prototype Materials with their Usage

List of software used

Software	Purpose
LiquidCrystal_I2C	Operates the I2C LCD screen to show outputs like voltage.
Servo	Directs the servo motors to adjust the solar panel's alignment.
Arduino IDE	A versatile platform used to write, compile, and upload the program code to the microcontroller.
Serial Monitor	A built-in feature of the Arduino IDE for real-time monitoring and debugging of sensor values and system outputs.
Embedded Math Functions	Used for processing sensor data, including averaging, calculating differences, and converting potentiometer readings to voltage.

List of Hardware components used

Components	Usage
Arduino Uno Board	Connects and manages system components through input/output pins.
USB Cable for Arduino	Connects the Arduino to a computer for programming and power.
Expansion Board	Adds extra features and extends system capabilities
Solar Panel	Captures sunlight for energy production.
I2C LCD Screen	Shows voltage readings for real-time performance tracking.
Two 10k Resistors	Ensures accurate voltage measurement and calculation.

Student Name: Sharifa Al Jowder

UON ID: 23856007

4 Photoresistor Modules	Senses light intensity, compares directional light levels and facilitates efficient movement.
2 Servo Motors	Adjust the solar panel's vertical and horizontal positioning for optimal sunlight exposure.
Wires (Male to Female, Female to Female)	Create connections for power delivery and signal transmission.

List of materials used for the prototype

Prototype material	Usage
Wooden Boards	Serve as the framework and base for the prototype.
Bolts	Securely join structural components.
Nuts	Work with bolts to tighten and stabilize connections.
Screws	Fasten various parts to the wooden base.
Glue	silicone and white glue were used to securely bond the wooden boards and other components of the prototype.

Evaluation of the output:

Observation:

The system accurately follows sunlight using photoresistor modules and positions the solar panel with servo motors, displaying voltage in real-time on the I2C LCD.

Reflection:

Advantages are affordability, modular design, and enhanced solar efficiency, while drawbacks include possible inaccuracies under diffused light and long-term servo motor wear.

Analysis:

The system enhances solar energy capture by up to 30%, offers affordability for small-scale applications, and supports renewable energy objectives, though upgrading sensors and motors could improve its reliability.

Student Name: Sharifa Al Jowder

UON ID: 23856007

Part 1 Code:

```
// Include necessary libraries
#include <Servo.h>                  // Library for controlling servos
#include <LiquidCrystal_I2C.h>      // Library for I2C-based LCD

// Initialize LCD with address 0x27, 16 columns, and 2 rows
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define pin numbers and parameters
#define SERVOPINH  5 // Pin for horizontal servo
#define SERVOPINV  6 // Pin for vertical servo
#define dtime   50  // Delay parameter: smaller = faster servo movement
#define tol     50  // Sensitivity parameter: smaller = more sensitive

// Horizontal servo settings
Servo horizontal;         // Servo object for horizontal control
int servoh = 90;          // Default horizontal servo angle
int servohLimitHigh = 175; // Maximum allowed horizontal angle
int servohLimitLow = 5;    // Minimum allowed horizontal angle

// Vertical servo settings
Servo vertical;           // Servo object for vertical control
int servov = 90;          // Default vertical servo angle
int servovLimitHigh = 100; // Maximum allowed vertical angle
int servovLimitLow = 20;   // Minimum allowed vertical angle

// Pin assignments for LDR sensors
const int ldrlt = A0; // Upper left LDR sensor
const int ldrrt = A1; // Upper right LDR sensor
const int ldrld = A2; // Lower left LDR sensor
const int ldrrd = A3; // Lower right LDR sensor

// Variables for voltage measurements
int potPin = 0;
int potValue = 0;
float voltageValue = 0.;

void setup() {
  Serial.begin(9600);      // Initialize serial communication
  lcd.init();              // Initialize LCD
  lcd.backlight();         // Turn on LCD backlight
  lcd.setCursor(0, 0);     // Set cursor to the first row
  lcd.print("Solar_Voltage"); // Display initial text
  lcd.cursor();            // Show cursor on LCD
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
  lcd.blink();             // Enable blinking cursor

  // Attach servos to their respective pins
  horizontal.attach(SERVOPINH);
  vertical.attach(SERVOPINV);
  horizontal.write(servoh); // Set initial horizontal angle
  vertical.write(servov);   // Set initial vertical angle
  delay(100);               // Short delay for setup
}

void loop() {
  // Read light intensity from LDR sensors
  int lt = analogRead(ldrlt); // Upper left LDR
  int rt = analogRead(ldrrt); // Upper right LDR
  int ld = analogRead(ldrld); // Lower left LDR
  int rd = analogRead(ldrrd); // Lower right LDR

  // Calculate averages of adjacent sensors
  int avt = (lt + rt) / 2; // Average of upper sensors
  int avd = (ld + rd) / 2; // Average of lower sensors
  int avl = (lt + ld) / 2; // Average of left sensors
  int avr = (rt + rd) / 2; // Average of right sensors

  // Calculate differences in light intensity
  int dvert = avt - avd; // Vertical difference (top vs. bottom)
  int dhoriz = avl - avr; // Horizontal difference (left vs. right)

  // Print sensor values and calculations to serial monitor
  Serial.print(lt); Serial.print(",");
  Serial.print(rt); Serial.print(",");
  Serial.print(ld); Serial.print(",");
  Serial.print(rd); Serial.print("      |    ");
  Serial.print(avt); Serial.print(",");
  Serial.print(avd); Serial.print(",");
  Serial.print(avl); Serial.print(",");
  Serial.print(avr); Serial.print(",   ");
  Serial.print(dtime); Serial.print(",   ");
  Serial.println(tol);

  // Read value and calculate voltage
  potValue = analogRead(potPin);
  voltageValue = 10. * (potValue / 1023.); // voltage divider resistances of two // 10 KΩ resistors
  lcd.setCursor(0, 1); // Set cursor to second row
  lcd.print("PV VOLT=");
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
  lcd.setCursor(9, 1);
  lcd.print(voltageValue);
  lcd.setCursor(14, 1);
  lcd.print("V");

  // Adjust vertical servo based on light intensity difference
  if (abs(dvert) > tol) {
    if (avt < avd) {
      servov = min(servov + 1, servovLimitHigh); // Move up
    } else {
      servov = max(servov - 1, servovLimitLow); // Move down
    }
    vertical.write(servov);
  }

  // Adjust horizontal servo based on light intensity difference
  if (abs(dhoriz) > tol) {
    if (avl < avr) {
      servoh = max(servoh - 1, servohLimitLow); // Move left
    } else {
      servoh = min(servoh + 1, servohLimitHigh); // Move right
    }
    horizontal.write(servoh);
  }

  delay(dtime); // Wait for a short period before next loop iteration
}
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

Part 2 – RGB Floodlight

YouTube link – [RGB FLOODLIGHT YOUTUBE LINK]

Figure 3 Snapshot of the Last edited information for RBG FLOODLIGHT

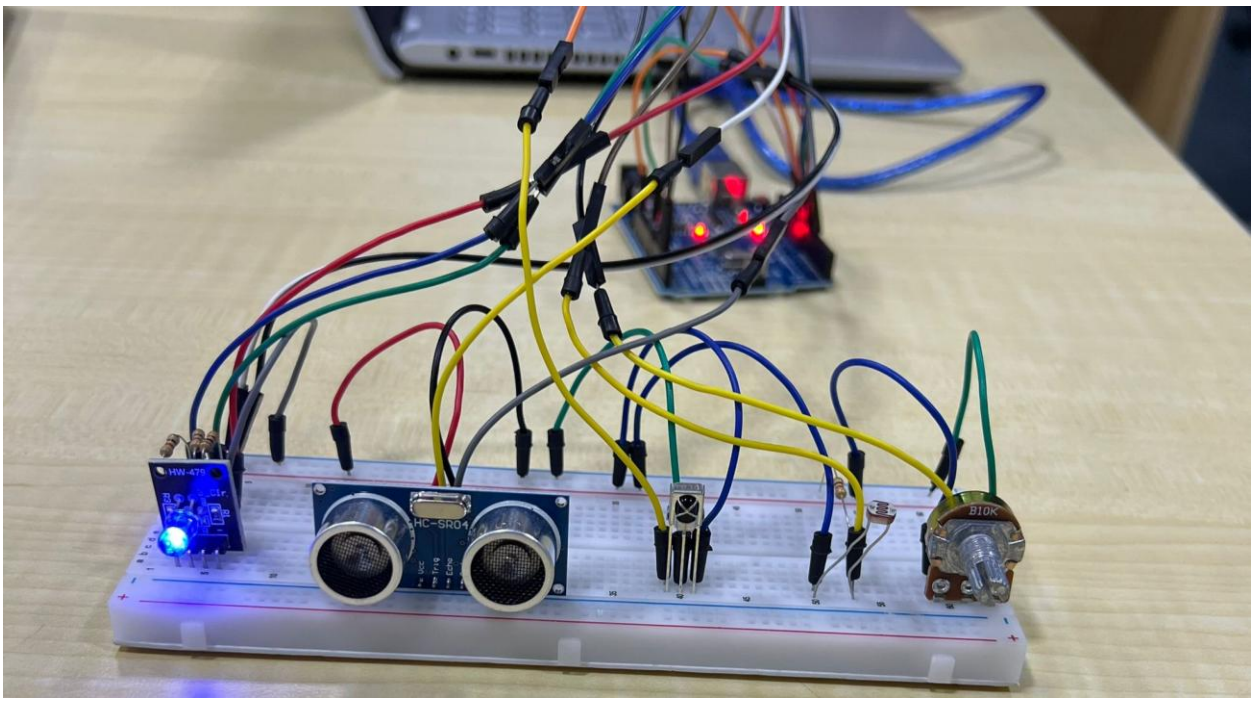

Figure 4 Image of the final design for the RGB FLOODLIGHT

Student Name: Sharifa Al Jowder

UON ID: 23856007

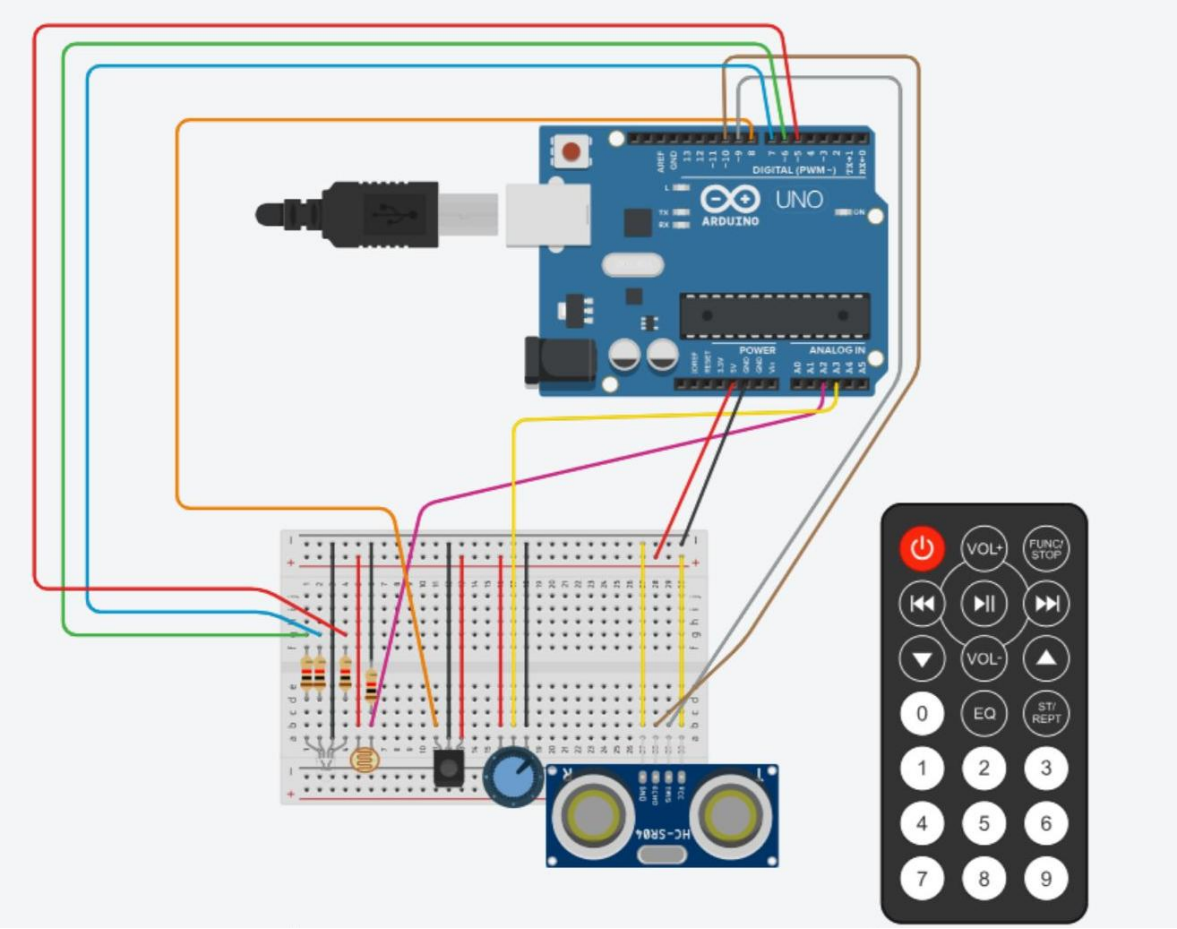

Figure 5 Project prototype using Tinkercard app for 3D design for the RGB FLOODLIGHT

Student Name: Sharifa Al Jowder

UON ID: 23856007

Introduction

The RGB Floodlight Project is a smart lighting system that improves home security and customization. It guarantees efficiency and versatility by combining sensors, RGB LED lighting and programmable features. The device can detect motion and proximity, works at night, and provides manual control and user-defined lighting effects. It uses programmable daylight thresholds to automate illumination, only turning on when user-specified circumstances are satisfied. Additional features include dimming, remote-controlled RGB effects, and adjustable motion detection.

How code works

This home security floodlight is a smart lighting system that uses a combination of components like an Arduino microcontroller, color-changing LEDs, a light sensor, and an ultrasonic sensor.

- **Automated Lighting:** The light sensor and potentiometer let you adjust how sensitive the light is to daylight, while the ultrasonic sensor detects motion and automatically turns on the light when it senses something within a set distance.
- **Color & Effects:** The RGB LED can display different colors and create cool lighting effects like flashing, fading, and smooth transitions.
- **User Control:** An infrared receiver and remote control give you manual control over the light, allowing you to dim it, change modes, and adjust the RGB colors.
- **Versatility:** This floodlight is highly adaptable with its adjustable lighting effects, brightness, and sensitivity settings.

Definitions of Features

1. **Jump Color Change:** This feature allows colors to change instantly without blending or fading, giving alarms or quick lighting changes a striking and dynamic appearance.

2. **Stroboscopic Alteration in Color:** Like strobe light, this effect creates brief, repetitive color flashes. It is frequently employed for amusement or high exposure.

3. **Gradual Color Change:** The fading effect of the colors' gradual transition is perfect for mood or ambient lighting.

4. **Smooth Color Change:** This feature guarantees a smooth, continuous change between several colors, combining hues organically to create a colorful and harmonious impression.

Student Name: Sharifa Al Jowder

UON ID: 23856007

Libraries & Components (Hardware and software) with Their Usage

List of software used

Software	Purpose
IRremote by Shirriff 4.4.1	Processes IR remote signals for RGB adjustments and mode switching.
HC-SR04	Handles ultrasonic sensor inputs for distance-based motion detection.
Arduino IDE	This platform is employed for coding, compiling, and transferring programs to the microcontroller.
Serial Monitor	Used to display debugging information like infrared commands, light levels, and sensor feedback.
Built-in Analog and Digital Pin Utilities	Manages inputs (e.g., light sensor, potentiometer) and outputs (e.g., RGB LED signals).

List of Hardware components used

Component	Usage
Wires	Establish electrical connections between the board, sensors, and LEDs.
LDR	Detects light levels to activate or deactivate the floodlight based on adjustable thresholds.
Ultrasonic Sensor	Measures distances to enable motion-triggered light activation within a defined range.
Arduino Uno	Processes sensor data and controls the lighting system.
USB Cable	Powers the board and allows code uploads.
Potentiometer	Adjusts brightness and light sensitivity thresholds.
IR Receiver	Interprets signals from the remote for light control and mode switching.
IR Remote	Sends commands for lighting adjustments, effects selection, and manual overrides.

Student Name: Sharifa Al Jowder

UON ID: 23856007

Resistors	Prevents excess current to protect components.
RGB LED	Delivers multicolor lighting with customizable brightness and effects.
Bread Board	offers a solderless platform for short-term circuit connections.

IR Remote Control Commands, buttons and Actions

IR Remote Code (Decimal)	IR Remote buttons used	Action Taken	Feedback/Sign Observed
94	3	Triggers red color for 4 seconds.	The red LED illuminates.
12	1	Activates green color for 4 seconds.	The green LED lights up.
8	4	Enables blue color for 4 seconds.	The blue LED illuminates.
24	2	Turns on white color for 4 seconds.	All LEDs light up in white color.
68	Prev	Switch to the previously set color for 4 seconds.	LED changes to the previous color in sequence.
64	Next	Moves to the next color for 4 seconds.	LED changes to the next color in the cycle.
25	100+	Initiates a slow cycling of colors.	LEDs cycle through colors at a slow pace.
13	200+	Starts fast cycling of colors.	LEDs cycle through colors rapidly.
67	Play/pause	Pauses or resumes the system's operation.	System halts or continues; LED state remains unchanged.

Student Name: Sharifa Al Jowder

UON ID: 23856007

Problems faced during the project

Numerous difficulties surfaced during the project's development, especially with the IR remote capabilities. Occasionally, the remote sent erratic signals, which caused the system to behave unexpectedly. It took more work to debug these anomalies to guarantee precise communication between the receiver and the remote. Another problem arose during the implementation of the pause function; it was challenging to switch between the active and paused states seamlessly, particularly when handling concurrent commands. Furthermore, calibrating the LDR sensitivity was difficult because it took a lot of testing in different lighting circumstances to determine the ideal light detection threshold for dependable RGB lighting activation. To complete the project, these issues required meticulous debugging and modifications.

Evaluation of the output

Observation:

This project shows how embedded systems can be used effectively in home automation, combining software and hardware to create a safe and attractive solution. Energy economy, easy-to-use settings, accurate sensor calibration, and dependable control algorithms are its main features.

Reflection:

The RGB Floodlight overcomes problems with sensor accuracy and feature management in a single codebase, showcasing the flexibility of embedded systems in solving real-world problems and improving abilities in real-time programming, sensor integration, and user interface design.

Analysis:

With RGB LEDs that can be customized, this project offers a scalable, intelligent lighting system that improves both functionality and appearance. Energy efficiency is guaranteed via automation and dimming, and strong sensor algorithms and adaptable code structures skillfully manage intricate hardware interactions.

Student Name: Sharifa Al Jowder

UON ID: 23856007

Part 2 Code:

```
#include <IRremote.hpp> // IR remote downloaded from library

// Pin Definitions
const int red_Pin_num = 5;    // Pin number for controlling the red LED
const int green_Pin_num = 6;  // Pin number for controlling the green LED
const int blue_Pin_num = 7;   // Pin number for controlling the blue LED
const int LDR_Pin_num = A2;   // Pin number for reading the LDR (Light Dependent Resistor) sensor
const int Trig_Pin_num = 9;   // Pin number for the ultrasonic sensor's trigger pin
const int Echo_Pin_num = 10;  // Pin number for the ultrasonic sensor's echo pin
const int Pot_Pin_num = A3;   // Pin number for reading the potentiometer input
const int IR_pin_num = 8;    // Pin number for receiving IR remote signals

// Variables
const int darknessThreshold = 80;  // The LDR sensitivity limit for detecting darkness.
const int minDistance = 15;  // The minimum detection range of the ultrasonic sensor in centimeters.
const unsigned long ultrasonicDuration = 8000;  //The active duration of the ultrasonic RGB sensor in
milliseconds.
const unsigned long ldrDurationMin = 2000;  // The minimum active duration of the LDR by RGB sensor
in milliseconds.
const unsigned long ldrDurationMax = 4000; // The maximum active duration of the LDR by RGB sensor in
milliseconds.

unsigned long rgbDeactivateTime = 0;  // The timer setting for disabling the RGB functionality.
bool rgbActive = false;  // The operational state of the RGB system.
bool isPaused = false; // Indicates whether the system is paused
bool irActive = false; // Tracks the active state of the IR remote
unsigned long irActiveTime = 0;  // Timer for managing RGB activation via IR control
int currentRed = 0, currentGreen = 0, currentBlue = 0;  // Stores the current RGB color values
int brightness = 255; // The standard brightness level.

// Infrared Remote Commands
const uint32_t COLOR_RED_CODE = 94;     // IR code for activating the red color
const uint32_t COLOR_GREEN_CODE = 12;   // IR code for activating the green color
const uint32_t COLOR_BLUE_CODE = 8;     // IR code for activating the blue color
const uint32_t COLOR_PREV_CODE = 68;    // IR code for switching to the previous color
const uint32_t COLOR_NEXT_CODE = 64;    // IR code for switching to the next color
const uint32_t COLOR_WHITE_CODE = 24;   // IR code for activating the white color
const uint32_t MODE_SLOW_CYCLE = 25;    // IR code for enabling slow color cycling
const uint32_t MODE_FAST_CYCLE = 13;    // IR code for enabling fast color cycling
const uint32_t PAUSE_CODE = 67;         // IR code for pausing or resuming the system

int colors[][3] = {
  {255, 0, 0},   // RGB values for Red color
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
  {0, 255, 0},   // RGB values for Green color
  {0, 0, 255},   // RGB values for Blue color
  {255, 255, 255} // RGB values for White color
};

int currentColorIndex = 0;

IRrecv irrecv(IR_pin_num);
decode_results results;

void setup() {
  Serial.begin(9600);
  IrReceiver.begin(IR_pin_num, ENABLE_LED_FEEDBACK);

  pinMode(Trig_Pin_num, OUTPUT);  // Ultrasonic sensor's trigger pin as output
  pinMode(Echo_Pin_num, INPUT);  // Ultrasonic sensor's echo pin as input
  pinMode(LDR_Pin_num, INPUT);  // LDR (light-dependent resistor) pin as input
  pinMode(Pot_Pin_num, INPUT);  //Potentiometer pin as input
  pinMode(red_Pin_num, OUTPUT);  //Red LED pin as output
  pinMode(green_Pin_num, OUTPUT);  //Green LED pin as output
  pinMode(blue_Pin_num, OUTPUT);  //Blue LED pin as output

  setRGBColor(0, 0, 0);// Set RGB to off state
  Serial.println("System Initialization Complete");
}

void loop() {
  // Process IR commands
  if (IrReceiver.decode()) {
    uint32_t command = IrReceiver.decodedIRData.command;

    if (isValidCommand(command)) {
      handleIRCode(command);
    } else {
      Serial.println("Unknown or ignored IR command.");
    }
    IrReceiver.resume();
  }

  // Terminate all operations if the system is paused
  if (isPaused) {
    delay(100); // Minimize CPU usage
    return;
  }
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
 // Manage IR timing to disable after the specified duration
  if (irActive && millis() > irActiveTime) {
    irActive = false;
    Serial.println("IR mode concluded. Returning to sensor operations.");
    setRGBColor(0, 0, 0); // Deactivate RGB lighting
  }

  handlePotentiometer();
  handleUltrasonic();
  handleLDR();

  // Check if RGB is active and the deactivation timer has expired
  if (rgbActive && millis() > rgbDeactivateTime) {
    rgbActive = false; // Deactivate the RGB
    setRGBColor(0, 0, 0);
  }

  delay(115); // Brief delay for system stability and to prevent excessive looping
}

// Function to verify and validate incoming IR commands
// Return true if the command matches any predefined IR code0
bool isValidCommand(uint32_t command) {
  return (command == COLOR_RED_CODE || command == COLOR_GREEN_CODE || command == COLOR_BLUE_CODE ||
          command == COLOR_WHITE_CODE || command == COLOR_PREV_CODE || command == COLOR_NEXT_CODE ||
          command == MODE_SLOW_CYCLE || command == MODE_FAST_CYCLE || command == PAUSE_CODE);
}

// Functionality for managing potentiometer input
void handlePotentiometer() {
  int potValue = analogRead(Pot_Pin_num); // Read the analog value from the potentiometer
  brightness = map(potValue, 0, 1023, 0, 255); // Map the potentiometer value to a brightness range
(0-255)
  Serial.print("Brightness Adjustment via Potentiometer: ");  // Log the brightness adjustment
  Serial.println(brightness);  // Output the adjusted brightness value
}

// Functionality for managing ultrasonic sensor input and operations
void handleUltrasonic() {
  float distance = getDistance();  // Get the distance measurement from the ultrasonic sensor
  // Check if the distance is within the defined range and RGB is not already active
  if (distance > 0 && distance <= minDistance && !rgbActive) {
    Serial.println("Motion detected by ultrasonic sensor. Activating RGB");   // Log motion detection
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
    rgbActive = true;  // Set the RGB active state to true
    rgbDeactivateTime = millis() + ultrasonicDuration; // Set the timer for RGB deactivation
    setRGBColor(255, 255, 255);  // Activate RGB with white color
  }
}

// Functionality for managing LDR (Light Dependent Resistor) sensor input
void handleLDR() {
  int lightLevel = analogRead(LDR_Pin_num);
  if (lightLevel < darknessThreshold && !rgbActive) {
    Serial.println("Dim conditions detected. Enabling RGB");
    rgbActive = true;
    rgbDeactivateTime = millis() + random(ldrDurationMin, ldrDurationMax);
    setRGBColor(0, 0, 255); // Blue color
  }
}

// Process input from the IR remote control
void handleIRCode(uint32_t command) {
  Serial.print("Valid IR Command Received: ");
  Serial.println(command);

// Switch between paused and active states
  if (command == PAUSE_CODE) {
    isPaused = !isPaused;
    if (isPaused) {
      Serial.println("System Paused. Holding current operations.");
    } else {
      Serial.println("System Resumed.");
    }
    return;
  }

 // Execute IR commands only when the system is not paused
  if (!isPaused) {
    irActive = true;
    switch (command) {
      case COLOR_RED_CODE:
        Serial.println("Red Color for 4 seconds.");
        setRGBColor(colors[0][0], colors[0][1], colors[0][2]);
        irActiveTime = millis() + 4000;
        break;
      case COLOR_GREEN_CODE:
        Serial.println("Green Color for 4 seconds.");
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
        setRGBColor(colors[1][0], colors[1][1], colors[1][2]);
        irActiveTime = millis() + 4000;
        break;
      case COLOR_BLUE_CODE:
        Serial.println("Blue Color for 4 seconds.");
        setRGBColor(colors[2][0], colors[2][1], colors[2][2]);
        irActiveTime = millis() + 4000;
        break;
      case COLOR_WHITE_CODE:
        Serial.println("White Color for 4 seconds.");
        setRGBColor(colors[3][0], colors[3][1], colors[3][2]);
        irActiveTime = millis() + 4000;
        break;
      case COLOR_NEXT_CODE:
        currentColorIndex = (currentColorIndex + 1) % 4;
        Serial.println("Next Color for 4 seconds.");
                        setRGBColor(colors[currentColorIndex][0],     colors[currentColorIndex][1],
colors[currentColorIndex][2]);
        irActiveTime = millis() + 4000;
        break;
      case COLOR_PREV_CODE:
        currentColorIndex = (currentColorIndex - 1 + 4) % 4;
        Serial.println("Previous Color for 4 seconds.");
                        setRGBColor(colors[currentColorIndex][0],     colors[currentColorIndex][1],
colors[currentColorIndex][2]);
        irActiveTime = millis() + 4000;
        break;
      case MODE_SLOW_CYCLE:
        cycleColors(3300);
        break;
      case MODE_FAST_CYCLE:
        cycleColors(950);
        break;
    }
  }
}

// Retrieve distance measurement from the ultrasonic sensor
float getDistance() {
  digitalWrite(Trig_Pin_num, LOW);  // Set the trigger pin to LOW to ensure a clean signal
  delayMicroseconds(3);  // Wait for a short duration before sending the trigger pulse
  digitalWrite(Trig_Pin_num, HIGH);  // Set the trigger pin to HIGH to send an ultrasonic pulse
  delayMicroseconds(8);  // Keep the pulse active for 8 microseconds
  digitalWrite(Trig_Pin_num, LOW);  // Set the trigger pin back to LOW to stop the pulse
```

Student Name: Sharifa Al Jowder

UON ID: 23856007

```
  long duration = pulseIn(Echo_Pin_num, HIGH);  // Measure the time it takes for the echo to return
  return (duration * 0.034) / 2;  // Calculate the distance in cm (time * speed of sound / 2)
}

// Set RGB color
void setRGBColor(int Red, int Green, int Blue) {
  if (!isPaused) {
    analogWrite(red_Pin_num, Red);
    analogWrite(green_Pin_num, Green);
    analogWrite(blue_Pin_num, Blue);
    currentRed = Red;
    currentGreen = Green;
    currentBlue = Blue;
  }
}

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