IE2060
Embedded Systems
2nd Year, 2nd Semester

Individual Assignment

IR remote controlled LED lamp

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of Science Special Honors Degree in Information Technology

04/05/2024

Declaration

I certify that this report does not incorporate without acknowledgement, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in text.

Registration Number : IT22360250

Name : Sahan W A G

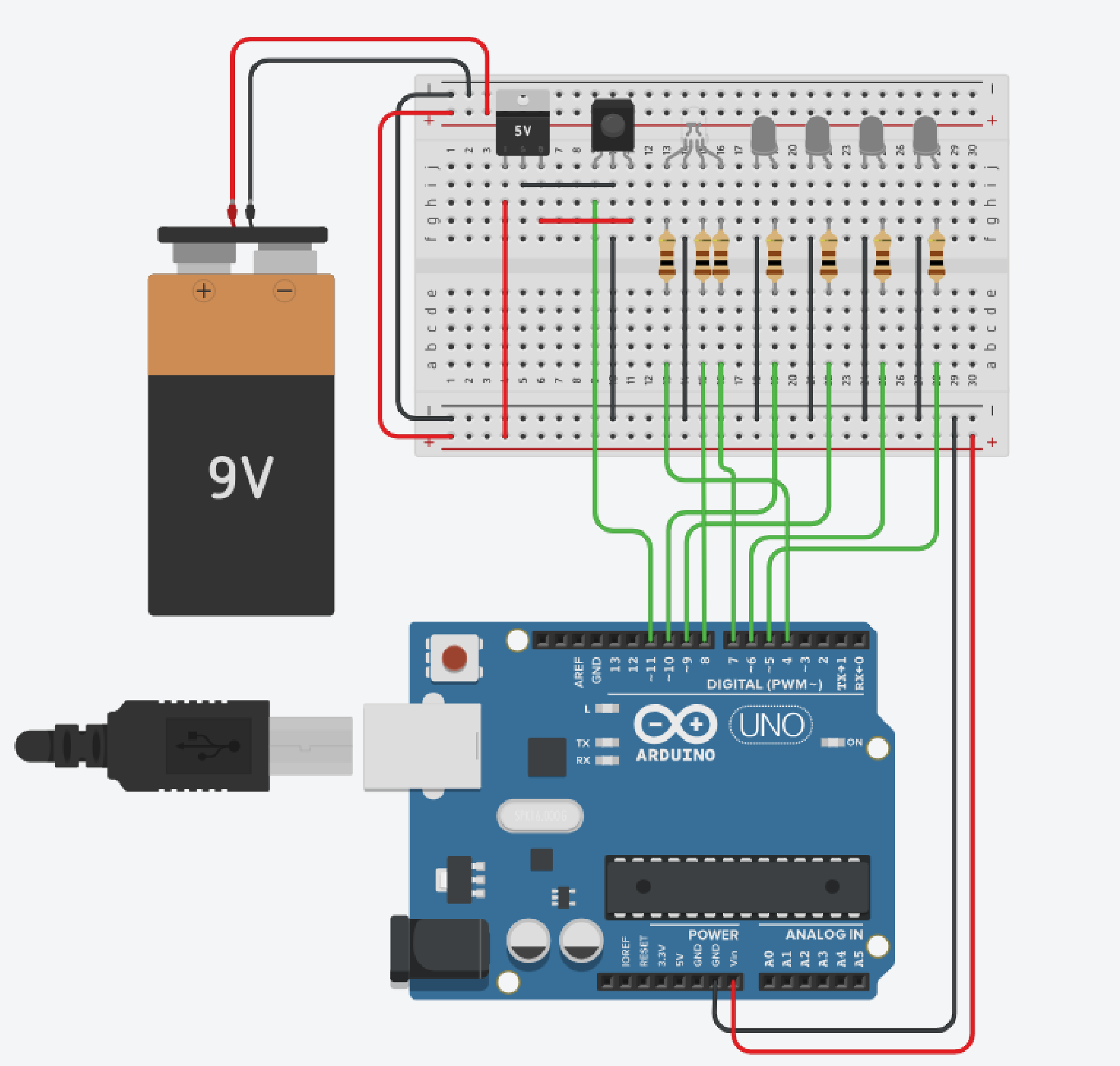
Introduction

The aim of the project is to design and develop remote controlled LED lamp using Arduino UNO (Atmega328p) microcontroller, IR(infrared) remote and receiver, LEDs. LED lamp's LED on, off, brightness level up, down can control using IR remote.

Design methodology

Circuit design

In the beginning of the designing process, Tinkercad used to design the circuit. PWM pin 5, 6, 9, 10 on the Arduino Uno board were allocated for while LEDs and pin 4, ,7 8 were allocated for multicolor LED. Pin 12 were allocated for IR receiver. Source Volts, LED Volts and current were considered when choosing the resistors for the LEDs (R = V/I). 5V voltage regulator used to maintain a constant 5V output for the IR receiver.



Components used:

- 1 x Arduino Uno board
- 1 x IR remote
- 1 x IR receiver
- 4 x White LED
- 1 x Multicolor LED
- 1 x 9V battery
- 7 x 100ohm resistors
- 1 x dot board
- Jumper wires
- Female headers and male headers

Component testing

Each components wired using jumper wires on the breadboard. LEDs were individually tested to ensure proper functionality. IR remote and IR receiver were tested to ensure each button of the remote gets unique values.

IR remote values

Simple code and “IRremote” library were used to get the values of the IR remote

```
#include <IRremote.h>

const int IR_PIN = 24;     // Define the IR receiver pin

IRrecv irrecv(IR_PIN);    // Define the IR receiver object

void setup() {

  Serial.begin(9600);

  irrecv.enableIRIn();

}

void loop() {

  if (irrecv.decode()) {

    // Print the HEX value of the button press

    Serial.println(irrecv.decodedIRData.decodedRawData, HEX);

    irrecv.resume();    // Reset the IR receiver for the next signal

  }

}
```

Button	HEX value
UP Arrow	E718FF00
Down Arrow	AD52FF00
Right Arrow	A55AFF00
Left Arrow	F708FF00

Code Development

Atmel Studio 7 were used to develop the code. Developing the code gets few steps,

1. Coding the sequence of LED on and off when pressing ‘up’ and ‘down’ arrow buttons in remote.
2. Program the code to change the LEDs brightness.
3. Develop the code for toggling the red, green and blue when repeatedly pressing ‘up’ button.

Final circuit

After finalizing the breadboard circuit, components were solder to dot board

Code Implementation

```
#include <IRremote.h>


const int IR_PIN = 12;                         // define IR receiver pin

int Brightnesslvl = 150;                       // default brightness level = 150

int values[8] = {0,1,2,3,4,5,6,7};             // LED array

int i = 0;

int cycle = 0;


#define LED1 5                                 //pin 5 : 1st LED   PWM

#define LED2 6                                 //pin 6 : 2nd LED   PWM

#define LED3 9                                 //pin 9 : 3rd LED   PWM

#define LED4 10                                //pin 10 : 4th LED  PWM

#define REDLED 4                               //pin 4 : red LED   non-PWM
```

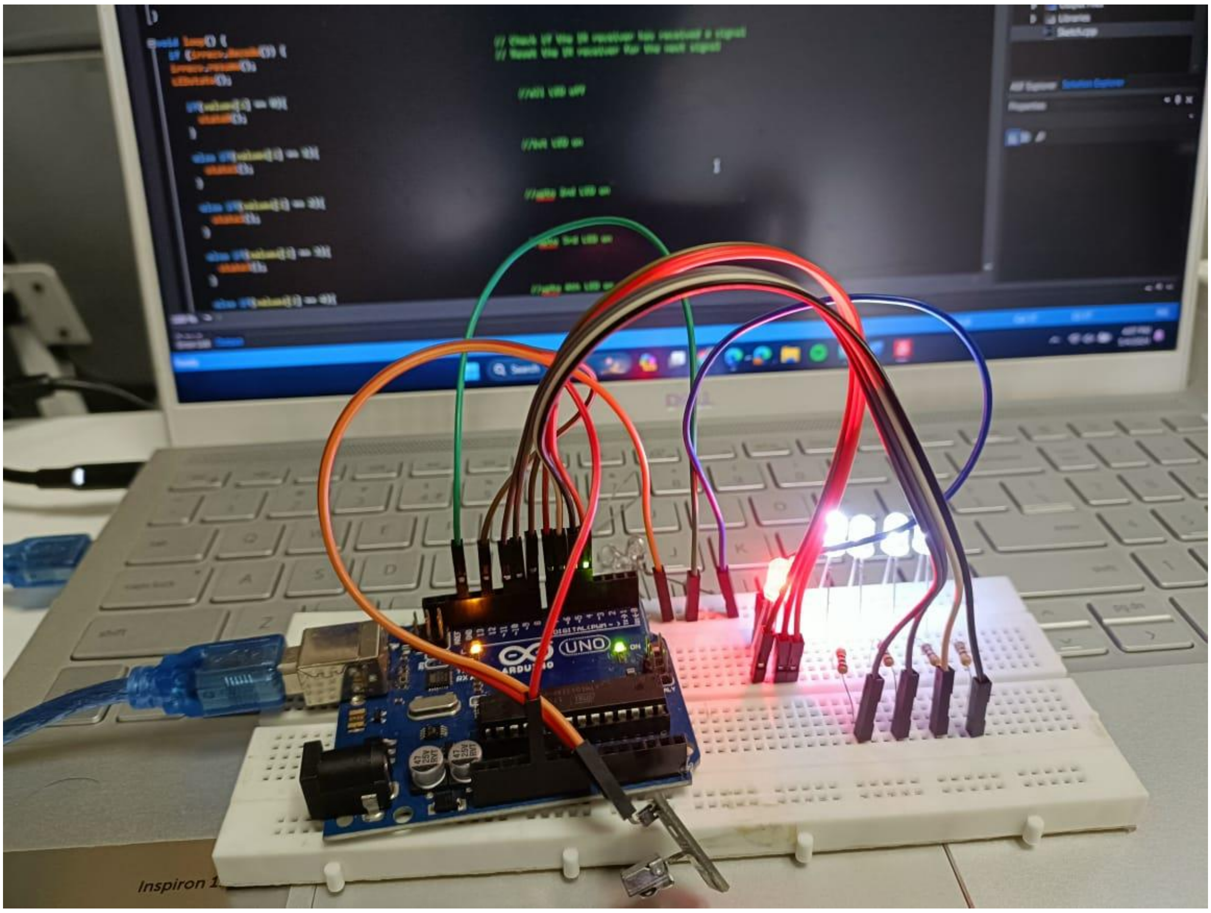
```
#define GREENLED 7                          //pin 7 : green LED non-PWM
#define BLUELED 8                           //pin 8 : blue LED  non-PWM

IRrecv irrecv(IR_PIN);                      // Define IR receiver object

void setup() {
  Serial.begin(9600);                       // Serial monitor
  irrecv.enableIRIn();

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(REDLED, OUTPUT);
  pinMode(GREENLED, OUTPUT);
  pinMode(BLUELED, OUTPUT);

}

void loop() {
  if (irrecv.decode()) {                    // Check if the IR receiver has received a
signal
  irrecv.resume();                          // Reset the IR receiver for the next signal
  LEDstate();

    if(values[i] == 0){                     //all LED off
      state0();
```

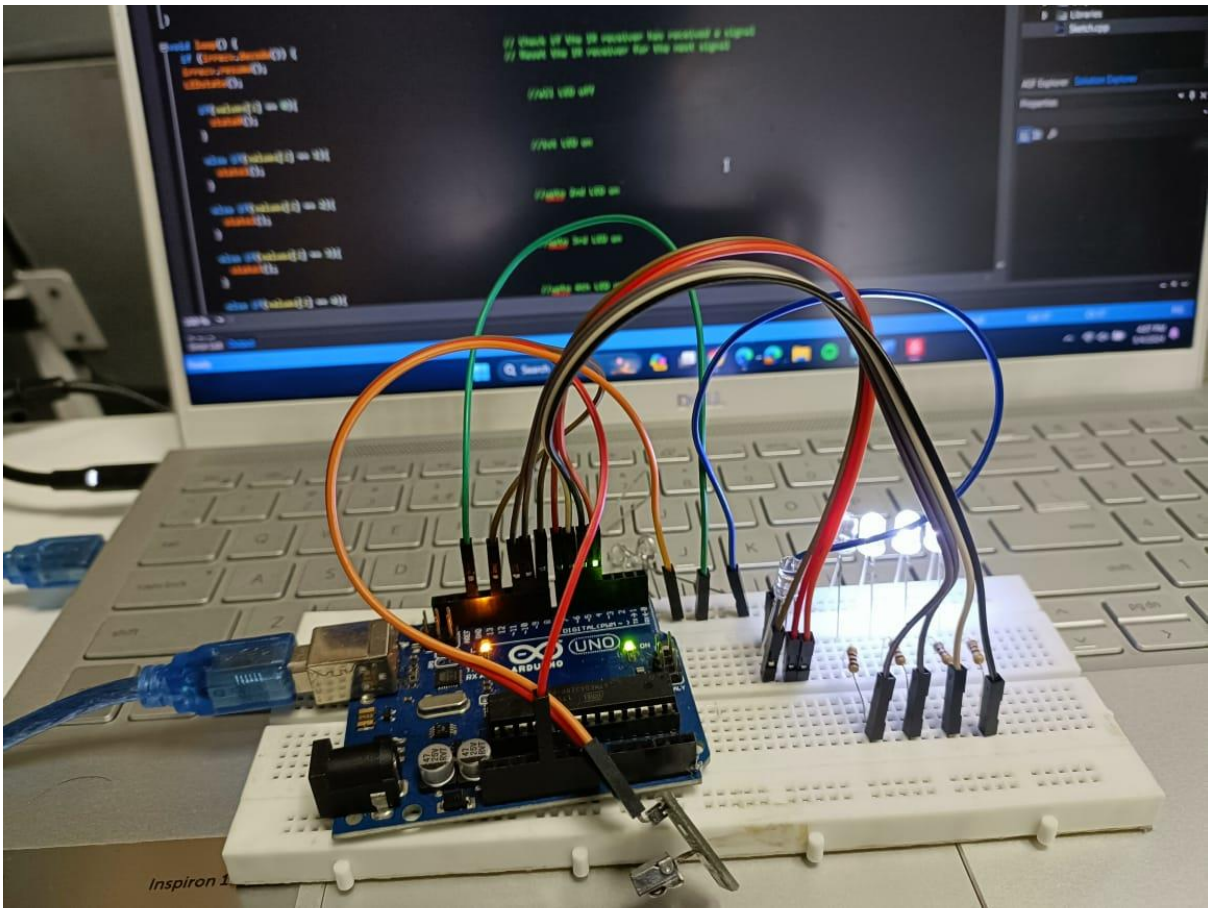
```
}

else if(values[i] == 1){                              //1st LED on
  state1();
}

else if(values[i] == 2){                              //up to 2nd LED on
  state2();
}

else if(values[i] == 3){                              //up to 3rd LED on
  state3();
}

else if(values[i] == 4){                              //up to 4th LED on
  state4();
}

else if(values[i] == 5){                              //4 white LED and red LED on
  state5();
}

else if(values[i] == 6){                              //4 white LED and green LED on
  state6();
}

else if(values[i] == 7){                              //4 white LED and blue LED on
```

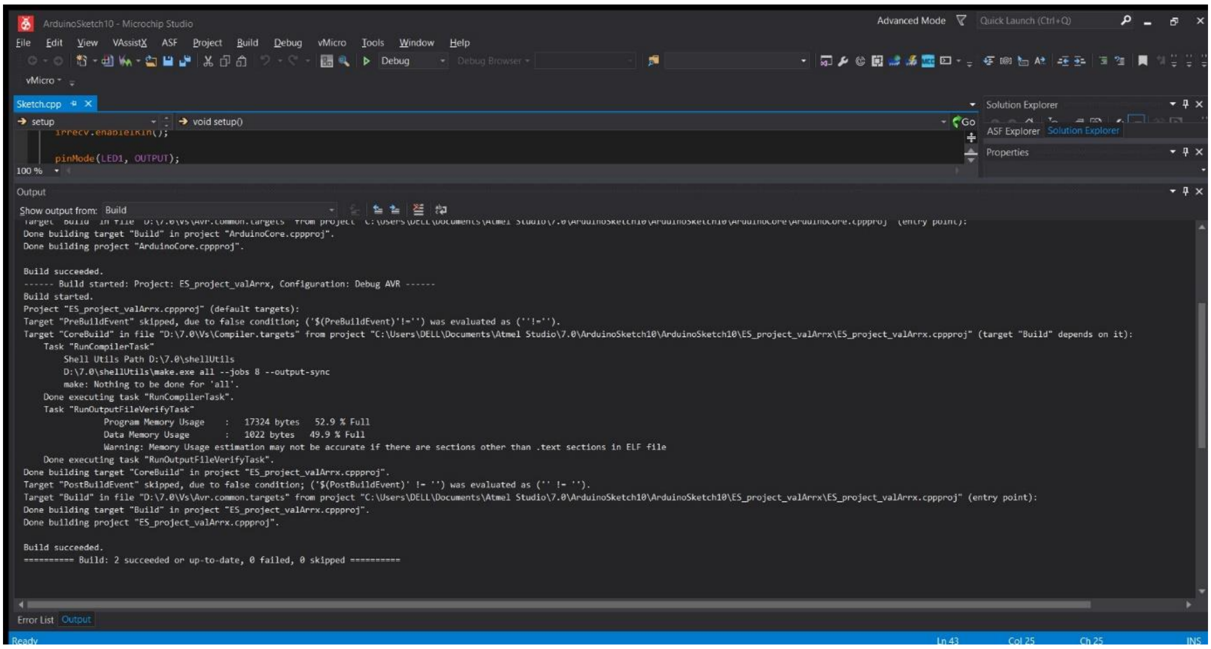
```
   state7();
  }
 }
}

void LEDstate(){
 if(irrecv.decodedIRData.decodedRawData == 0xE718FF00){        // check if receiver receive UP key value , 0xE718FF00 = UP
  if(i == 7){                         // check if LED in state 7
   i = 4;
   values[i];
   i ++;                              // cycle state5, state6, state7
   cycle ++;                          // count RGB cycles
   Serial.print("RGB cycles: ");
   Serial.println(cycle);
  }
  else{
   values[i];                         // if it is not in state 7, increase array without loop
   i ++;
  }
 }
 else if(irrecv.decodedIRData.decodedRawData == 0xAD52FF00){   // check if receiver receive DOWN key value , 0xAD52FF00 = DOWN
  if(i >= 0){                         // i always has to be 0 or higher
   if(cycle != 0){                    // check if RGB cycle not equals to zero
   if(i == 5){                        // cycle != 0 and i == 5
    i = 8;
    i --;
    values[i];

    cycle --;

    Serial.print("RGB cycles: ");

    Serial.println(cycle);

   }

   else{                    // cycle != 0 and i != 5

    values[i];

    i --;

   }

  }else{                    // RGB cycles are 0

   values[i];

   i --;

  }

  }

  else{

   i = 0;

  }

 }

 else if(irrecv.decodedIRData.decodedRawData == 0xA55AFF00){   // check if receiver
receive RIGHT key value , 0xA55AFF00 = RIGHT

  if(Brightnesslvl < 250){

   Brightnesslvl = Brightnesslvl + 50;

   Serial.print("Brightness: ");

   Serial.println(Brightnesslvl);

  }

  else if(Brightnesslvl >= 250){       // once brightness lvl is 250 or get higher, brightness lvl
set to 250

   Brightnesslvl = 250;
```

```
    }
  }
  else if(irrecv.decodedIRData.decodedRawData == 0xF708FF00){   // check if receiver receive LEFT key value , 0xF708FF00 = LEFT
    if(Brightnesslvl > 50){
      Brightnesslvl = Brightnesslvl - 50;
      Serial.print("Brightness: ");
      Serial.println(Brightnesslvl);
    }
    else if(Brightnesslvl <= 50){          // once brightness lvl is 50 or get lower, brightness lvl set to 50
      Brightnesslvl = 50;
    }
  }
}

void state0(){
  analogWrite(LED1, 0);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: -");
}
void state1(){
  analogWrite(LED1, Brightnesslvl);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: W");
}
void state2(){
  analogWrite(LED1, Brightnesslvl);
  analogWrite(LED2, Brightnesslvl);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: W  W");
}
void state3(){
  analogWrite(LED1, Brightnesslvl);
  analogWrite(LED2, Brightnesslvl);
  analogWrite(LED3, Brightnesslvl);
  digitalWrite(LED4, LOW);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: W  W  W");
```

```
}
void state4() {
  analogWrite(LED1, Brightnesslvl);
  analogWrite(LED2, Brightnesslvl);
  analogWrite(LED3, Brightnesslvl);
  analogWrite(LED4, Brightnesslvl);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: W  W  W  W");
}
void state5(){
  analogWrite(LED1, Brightnesslvl);
  analogWrite(LED2, Brightnesslvl);
  analogWrite(LED3, Brightnesslvl);
  analogWrite(LED4, Brightnesslvl);
  digitalWrite(REDLED, HIGH);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: W  W  W  W  R");
}
void state6(){
  analogWrite(LED1, Brightnesslvl);
  analogWrite(LED2, Brightnesslvl);
  analogWrite(LED3, Brightnesslvl);
  analogWrite(LED4, Brightnesslvl);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, HIGH);
  digitalWrite(BLUELED, LOW);
  Serial.println("HIGH LEDs: W  W  W  W  G");
}
void state7(){
  analogWrite(LED1, Brightnesslvl);
  analogWrite(LED2, Brightnesslvl);
  analogWrite(LED3, Brightnesslvl);
  analogWrite(LED4, Brightnesslvl);
  digitalWrite(REDLED, LOW);
  digitalWrite(GREENLED, LOW);
  digitalWrite(BLUELED, HIGH);
  Serial.println("HIGH LEDs: W  W  W  W  B");
}
```


UNO
ARDUINO
Inspiron 1


UNO
ARDUINO
Inspiron 1



```
    irrecv.enableIRIn();

    pinMode(LED1, OUTPUT);
```

```
Target "Build" in file "D:\7.0\Vs\Avr.common.targets" from project "C:\Users\DELL\Documents\Atmel Studio\7.0\ArduinoSketch10\ArduinoSketch10\ArduinoCore\ArduinoCore.cppproj" (entry point):
Done building target "Build" in project "ArduinoCore.cppproj".
Done building project "ArduinoCore.cppproj".

Build succeeded.
------ Build started: Project: ES_project_valArrx, Configuration: Debug AVR ------
Build started.
Project "ES_project_valArrx.cppproj" (default targets):
Target "PreBuildEvent" skipped, due to false condition; ('$(PreBuildEvent)'!='') was evaluated as (''!='').
Target "CoreBuild" in file "D:\7.0\Vs\Compiler.targets" from project "C:\Users\DELL\Documents\Atmel Studio\7.0\ArduinoSketch10\ArduinoSketch10\ES_project_valArrx\ES_project_valArrx.cppproj" (target "Build" depends on it):
    Task "RunCompilerTask"
        Shell Utils Path D:\7.0\shellUtils
        D:\7.0\shellUtils\make.exe all --jobs 8 --output-sync
        make: Nothing to be done for 'all'.
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
            Program Memory Usage    :   17324 bytes   52.9 % Full
            Data Memory Usage       :   1022 bytes   49.9 % Full
            Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "ES_project_valArrx.cppproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "D:\7.0\Vs\Avr.common.targets" from project "C:\Users\DELL\Documents\Atmel Studio\7.0\ArduinoSketch10\ArduinoSketch10\ES_project_valArrx\ES_project_valArrx.cppproj" (entry point):
Done building target "Build" in project "ES_project_valArrx.cppproj".
Done building project "ES_project_valArrx.cppproj".

Build succeeded.
========== Build: 2 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Testing and validation

Testing and validation ensure that remote controlled LED lamp operate reliability and usability. Also check the lamp meets the requirements of the project.

Functionality testing

- Each LED turned on when the UP-ARROW button was pressed.
- Multicolor LED toggle through the red, green, blue when UP-ARROW button repeatedly pressed
- Brightness level of the four LEDs ware reduce when LEFT-ARROW button pressed
- Brightness level of the four LEDs were increased when RIGHT-ARROW button pressed

The default brightness level is 150 in the lamp. There are 5 brightness levels implemented (50, 100, 150, 200 and 250).

Circuit stability

1. A 5V voltage regulator was used when powering up the IR receiver. It prevents corrupt remote values upon unstable voltage.
2. 100ohms resisters were used to keep durability of the LEDs.
3. LEDs and IR receiver were connected to female headers. So, if LEDs or IR receiver gets failure, it can easily be replaced.

Discussion

This project successfully demonstrates the design and implementation of a remote-controlled LED lamp using an Arduino microcontroller, infrared remote control, and various LEDs. The design process was methodical, beginning with circuit simulation on Tinkercad to ensure proper component selection and connectivity. Thorough component testing was conducted, verifying the functionality of individual LEDs and capturing the unique hex values associated with each button on the IR remote.

The code development process was well-structured, addressing key functionalities such as LED sequencing, brightness control, and color toggling for the multi-color LED. The implementation leveraged pulse-width modulation (PWM) for brightness control and efficient use of Arduino pins. The final circuit was neatly soldered onto a dot board, enhancing durability and compactness.

Extensive testing and validation were carried out to ensure the lamp met the project requirements. All functionalities, including LED activation, brightness adjustment, and color cycling, were thoroughly tested and found to operate reliably. The implementation of multiple brightness levels (50, 100, 150, 200, and 250) adds versatility to the lamp's usability.

Notably, the project emphasizes circuit stability and component protection. The inclusion of a 5V voltage regulator for the IR receiver ensures stable operation, preventing corrupt remote values due to voltage fluctuations. The use of appropriate resistors for the LEDs enhances their durability and longevity. The modular design, with LEDs and the IR receiver connected via female headers, allows for easy component replacement in case of failure.

Overall, this project showcases a well-designed and implemented remote-controlled LED lamp, demonstrating proficiency in microcontroller programming, electronic circuit design, and systematic testing and validation procedures.

Reference

[1] agarwalkrishna3009, “Decode IR Remote Control Signals of any Remote Using Arduino”, 19-mar-2022 [Online]. Available: https://projecthub.arduino.cc/agarwalkrishna3009/decode-ir-remote-control-signals-of-any-remote-using-arduino-9b8e30

[2] shirriff, z3to, ArminJo, “IRremote” [Online]. Available: https://www.arduino.cc/reference/en/libraries/irremote/