

**IE2070**

**Embedded Systems**

**2nd Year, 2nd Semester**

Assignment

**IR Remote controlled Lamp Circuit**

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 : IT22231628 Name : SHARVAJEN.S

#### Contents

[1 **Introduction** 4](#_Toc8692)

[2. Components 4](#_Toc8693)

[3. Mechanism 4](#_Toc8694)

[4.Circuit Prototype 5](#_Toc8695)

[5.Source Code of Programme 6](#_Toc8696)

[Atmel Studio Implementation 11](#_Toc8697)

[Circuit Functions 13](#_Toc8698)

[References 15](#_Toc8699)

# 1 **Introduction**

This circuit allows you to control Light-Emitting Diodes (LEDs) using an infrared (IR) remote control. The microcontroller development environment used here is Atmel Studio, and the programming language is C. The microcontroller used in this guide is an Atmel AVR 328p, which is similar to the Arduino Uno. The circuit consists of an IR receiver that captures the IR signal from the remote control. The microcontroller then decodes the signal to determine which button on the remote was pressed. The microcontroller then uses this information to control the LEDs.

# 2. Components

1. Arduino Uno board 1
2. Breadboard 1
3. IR sensor 1
4. IR remote 1
5. 4 white LEDs 4
6. RGB LED 4pin 1
7. Jumper wires 1set
8. 9V Battery 1
9. Resistor(1k ohm,2k ohm) 9

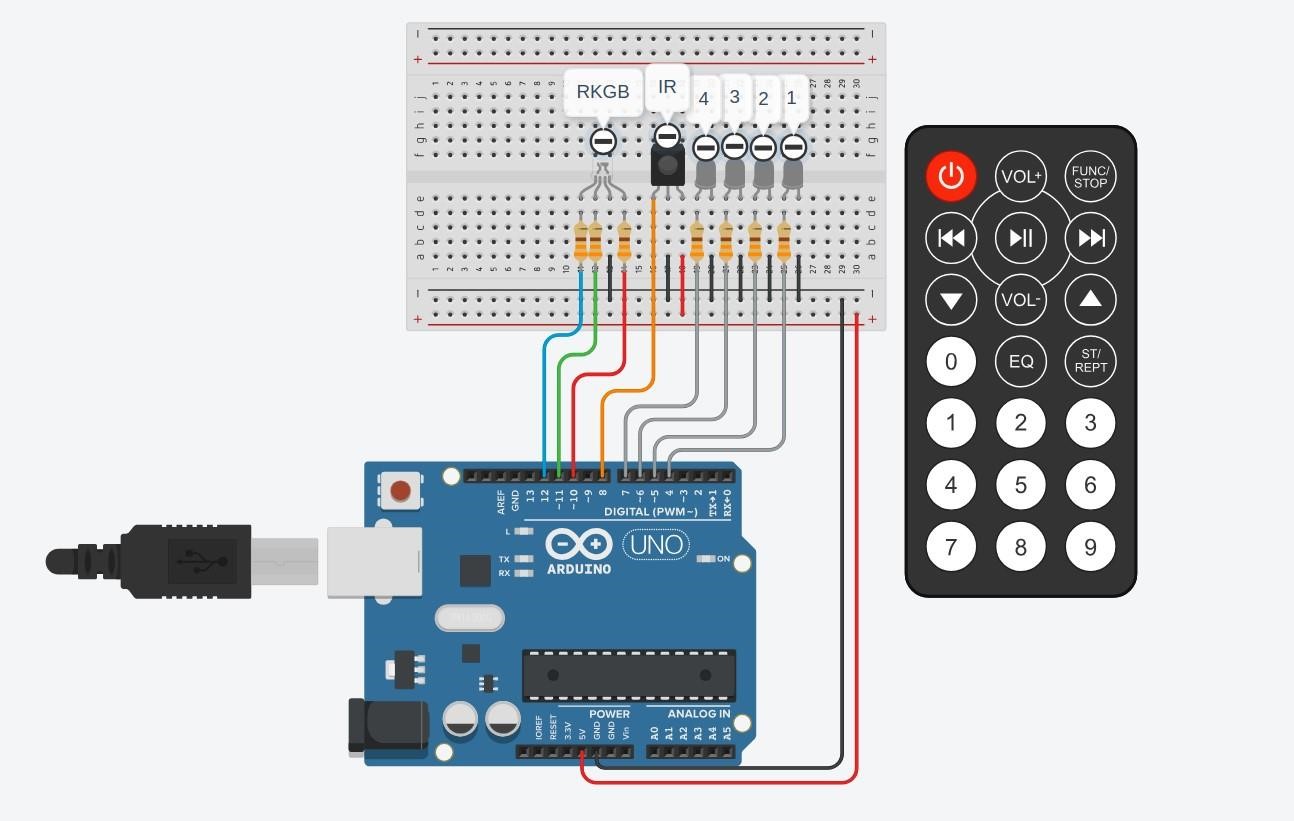
# 3. Mechanism

When the volume up button is pressed on the IR remote, the LEDs illuminate sequentially, one after another. Conversely, pressing the volume down button extinguishes the LEDs in the same manner, one by one. Once the sequence progresses to the colored LEDs, they cycle through from the first to the last when the volume up button is pressed again.

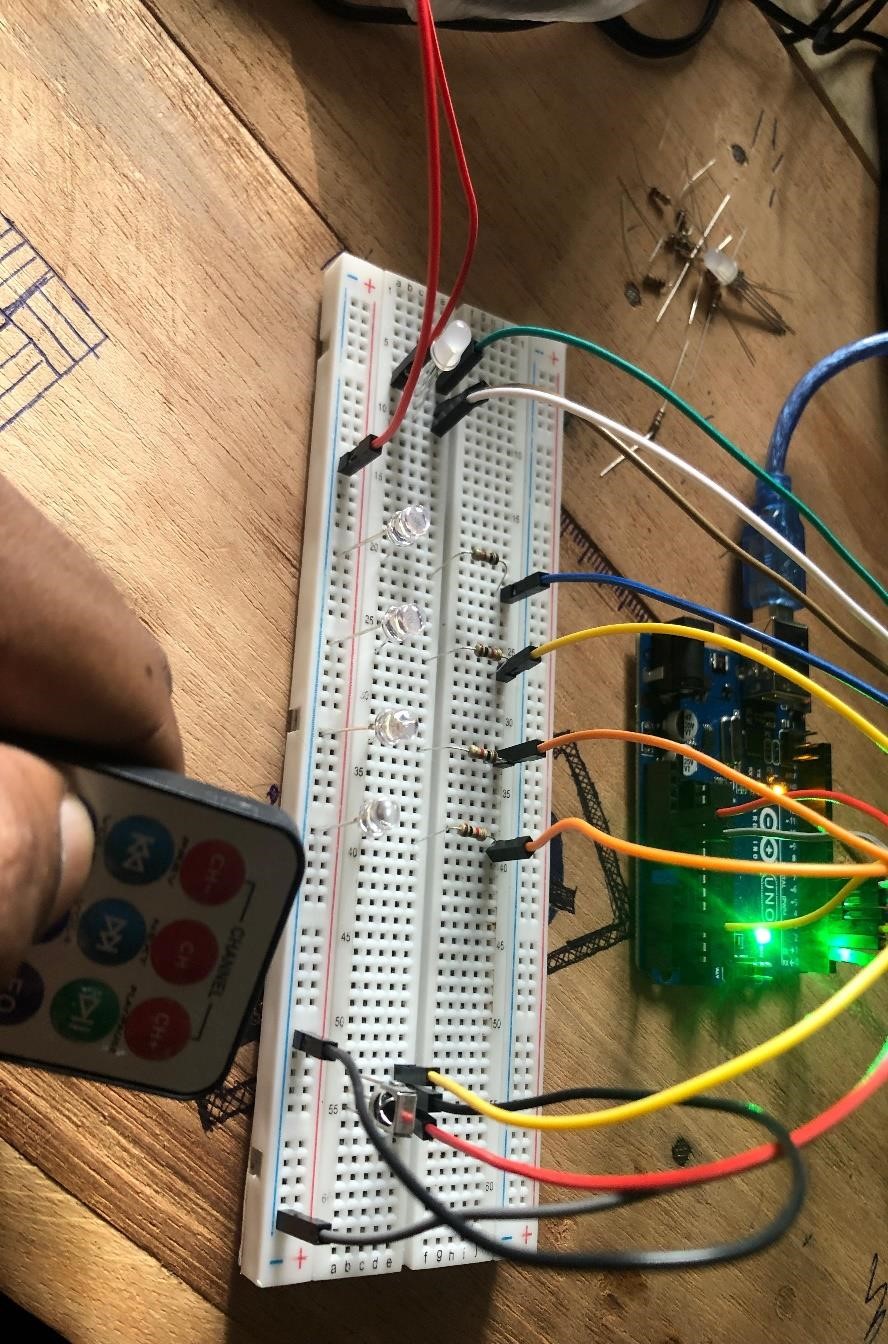
This operation relies on counters to track the number of times each switch is pressed, dictating which LED should illuminate or extinguish. For example:

* First press of volume up: Activates white LED 1.
* Second press: Activates white LED 2.
* Third press: Activates white LED 3.
* Fourth press: Activates white LED 4.
* Fifth press: Initiates the cycle through color LEDs, starting from color LED 1.
* Sixth press: Advances to color LED 2.
* Seventh press: Progresses to color LED 3.
* Eighth press: Returns to color LED 1, looping through the sequence.

# 4.Circuit Prototype



*Figure 1:circuit prototype for thinecad*



*Figure 2;real circuit*

# 5.Source Code of Program

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <IRremote.h>

// Pin definitions for LED connections

#define RED\_LED\_PIN PD2

#define GREEN\_LED\_PIN PD4

#define BLUE\_LED\_PIN PD7

#define WHITE\_LED\_1 PD3

#define WHITE\_LED\_2 PD5

#define WHITE\_LED\_3 PD6

#define WHITE\_LED\_4 PB3

// Pin definitions for IR receiver

#define IR\_RECEIVE\_PIN PB0

// IR Remote button codes

#define VOLUME\_UP\_CODE 21 // Update the correct code for Volume Up

#define VOLUME\_DOWN\_CODE 7 // Update the correct code for Volume Down

#define CHANNEL\_DOWN\_CODE 69 // Update the correct code for Channel Down

// State variables

int ledState = 0;

bool toggleColors = false;

int colorCounter = 1;

int led\_brightness[4] = {255, 255, 255, 255};

int reduced[4] = {0, 0, 0, 0};

int looped = 0;

void turnOn() {

ledState++;

if (ledState > 7) {

looped++;

ledState = 7;

toggleColors = true;

colorCounter = 1;

}

}

void turnOff() {

ledState--;

if (ledState <= 0) {

ledState = 0;

}

led\_brightness[ledState] = 255;

}

void cycleColors() {

colorCounter++;

if (colorCounter > 3) {

colorCounter = 1;

looped++;

}

}

void resetForColorToggle() {

ledState--;

if (ledState <= 0) {

ledState = 0;

}

}

void resetToWhite() {

colorCounter--;

if (colorCounter < 1 && looped > 0) {

colorCounter = 3;

looped--;

if (looped <= 0) {

toggleColors = false;

looped = 0;

}

}

}

void setAllLEDsLow() {

OCR2B = OCR0B = OCR0A = OCR2A = 0;

DDRD &= ~((1 << WHITE\_LED\_1) | (1 << WHITE\_LED\_2) | (1 << WHITE\_LED\_3));

DDRB &= ~(1 << WHITE\_LED\_4);

PORTD &= ~((1 << RED\_LED\_PIN) | (1 << GREEN\_LED\_PIN) | (1 << BLUE\_LED\_PIN));

}

void setWhiteLEDs(int count) {

if (count >= 1) {

DDRD |= \_BV(DDD3);

TCCR2A |= (\_BV(COM2B1) | \_BV(WGM21) | \_BV(WGM20));

TCCR2B |= (\_BV(CS20));

OCR2B = 255;

if (reduced[0] == 1) {

OCR2B = led\_brightness[0];

\_delay\_ms(10);

}

} else {

OCR2B = 0;

DDRD &= ~(1 << WHITE\_LED\_1);

}

if (count >= 2) {

DDRD |= \_BV(DDD5);

TCCR0A |= (\_BV(COM0B1) | \_BV(WGM01) | \_BV(WGM00));

TCCR0B |= (\_BV(CS00));

OCR0B = 255;

if (reduced[1] == 1) {

OCR0B = led\_brightness[1];

\_delay\_ms(10);

}

} else {

OCR0B = 0;

DDRD &= ~(1 << WHITE\_LED\_2);

}

if (count >= 3) {

DDRD |= \_BV(DDD6);

TCCR0A |= (\_BV(COM0A1) | \_BV(WGM01) | \_BV(WGM00));

TCCR0B |= (\_BV(CS00));

OCR0A = 255;

if (reduced[2] == 1) {

OCR0A = led\_brightness[2];

\_delay\_ms(10);

}

} else {

OCR0A = 0;

DDRD &= ~(1 << WHITE\_LED\_3);

}

if (count >= 4) {

DDRB |= \_BV(DDB3);

TCCR2A |= (\_BV(COM2A1) | \_BV(WGM21) | \_BV(WGM20));

TCCR2B |= (\_BV(CS20));

OCR2A = 255;

if (reduced[3] == 1) {

OCR2A = led\_brightness[3];

\_delay\_ms(10);

}

} else {

OCR2A = 0;

DDRB &= ~(1 << WHITE\_LED\_4);

}

PORTD &= ~((1 << RED\_LED\_PIN) | (1 << GREEN\_LED\_PIN) | (1 << BLUE\_LED\_PIN));

}

void setRGBLED(int ledPin) {

PORTD &= ~((1 << RED\_LED\_PIN) | (1 << GREEN\_LED\_PIN) | (1 << BLUE\_LED\_PIN));

PORTD |= (1 << ledPin);

}

void updateLEDs() {

if (!toggleColors) {

// Handle white LEDs and RGB LEDs for non-color toggle mode

switch (ledState) {

case 0: setAllLEDsLow(); break; // All off

case 1: setWhiteLEDs(1); break;

case 2: setWhiteLEDs(2); break;

case 3: setWhiteLEDs(3); break;

case 4: setWhiteLEDs(4); break;

case 5: setRGBLED(RED\_LED\_PIN); break; // Red LED on

case 6: setRGBLED(GREEN\_LED\_PIN); break; // Green LED on

case 7: setRGBLED(BLUE\_LED\_PIN); break; // Blue LED on

}

} else {

// Handle RGB LED for color toggle mode

switch (colorCounter) {

case 0: setWhiteLEDs(4); break; // All off

case 1: setRGBLED(RED\_LED\_PIN); break; // Red LED on

case 2: setRGBLED(GREEN\_LED\_PIN); break; // Green LED on

case 3: setRGBLED(BLUE\_LED\_PIN); break; // Blue LED on

}

}

}

void volumeUp() {

if (toggleColors) {

cycleColors();

} else {

turnOn();

}

updateLEDs();

\_delay\_ms(10); // Debounce delay

}

void volumeDown() {

if (ledState > 4 && !toggleColors) {

resetForColorToggle();

} else if (toggleColors) {

resetToWhite();

} else {

turnOff();

}

updateLEDs();

\_delay\_ms(10); // Debounce delay

}

void reduceBrightness() {

if (ledState == 0) {

return;

}

led\_brightness[ledState - 1] = 30;

reduced[ledState - 1] = 1;

updateLEDs();

}

int main(void) {

uint16\_t address = 0; // Variable to store IR address

uint16\_t command = 0; // Variable to store IR command

// Initialize white LEDs as outputs

DDRD |= ((1 << WHITE\_LED\_1) | (1 << WHITE\_LED\_2) | (1 << WHITE\_LED\_3));

DDRB |= (1 << WHITE\_LED\_4);

// Initialize RGB LEDs as outputs

DDRD |= ((1 << RED\_LED\_PIN) | (1 << GREEN\_LED\_PIN) | (1 << BLUE\_LED\_PIN));

IR\_init(); // Initialize IR remote control

while(1) {

if (IR\_codeAvailable()) { // Check if IR code is available

if (!IR\_isRepeatCode()) { // Check if IR code is not a repeat

IR\_getCode(&address, &command); // Get IR address and command

if (command == VOLUME\_UP\_CODE) {

volumeUp();

} else if (command == VOLUME\_DOWN\_CODE) {

volumeDown();

} else if (command == CHANNEL\_DOWN\_CODE) {

reduceBrightness();

}

}

}

}

}

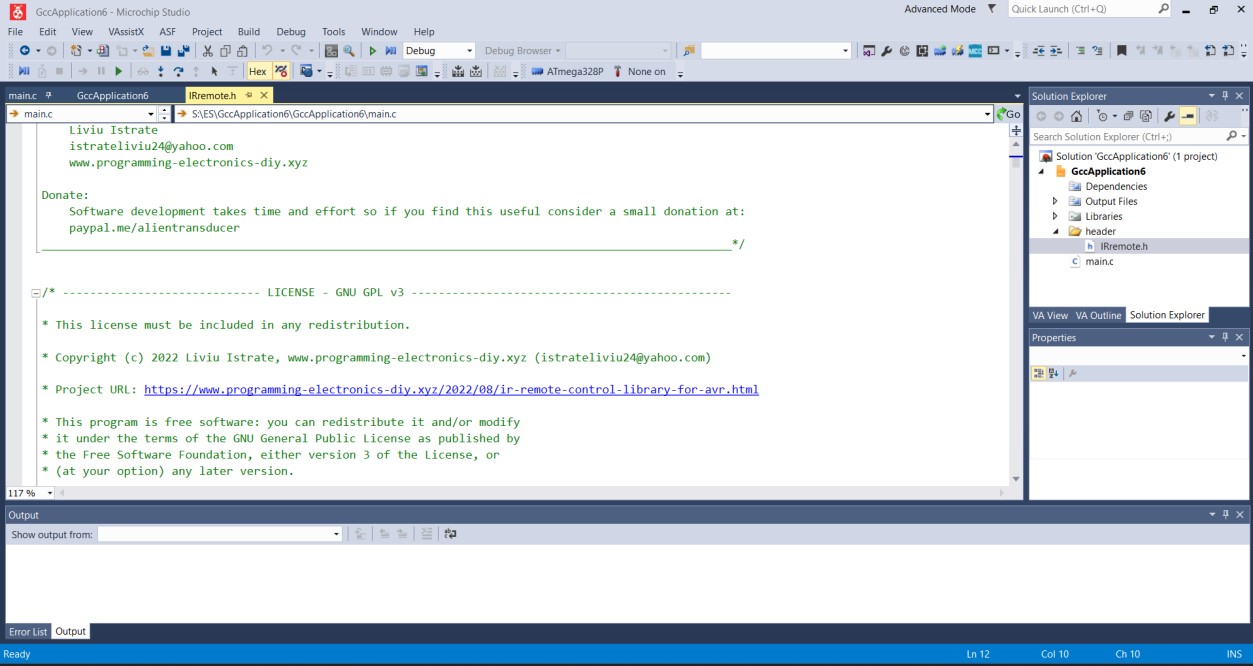
NOTE:

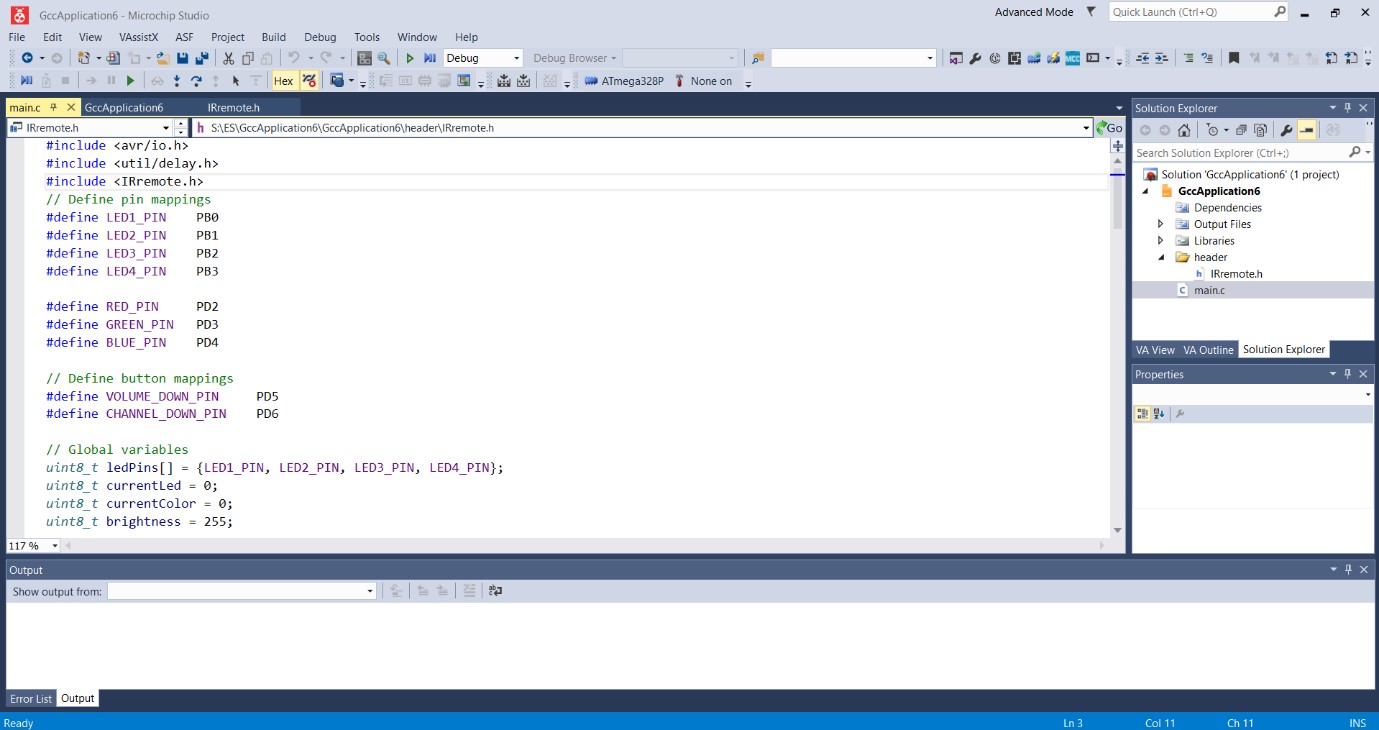
HOW TO DOWNLOAD LIBRARY

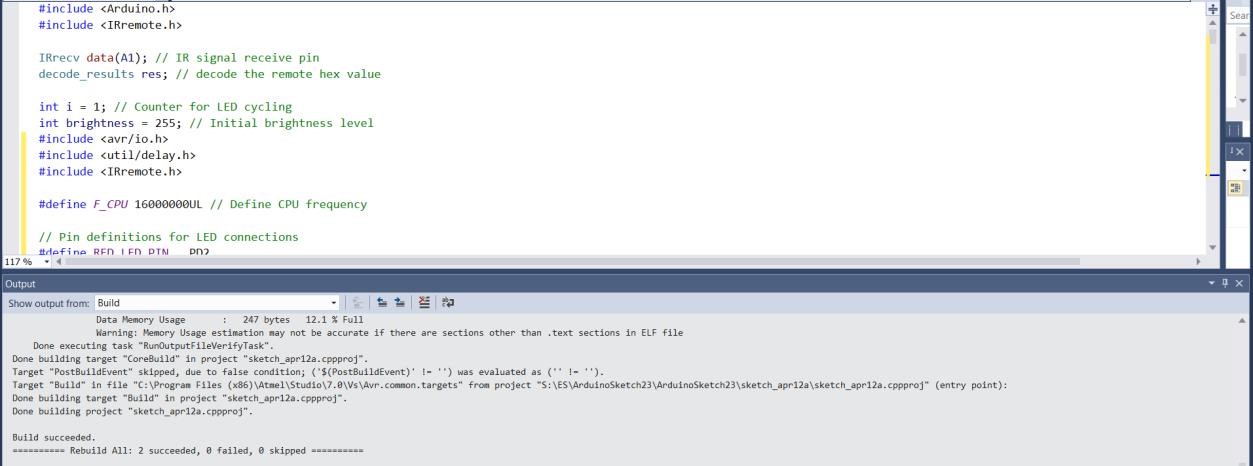
This is the library to download: https://drive.google.com/file/d/15vi9TJ4KipjbYkRRRGNbrDuCox2dz0VG/view

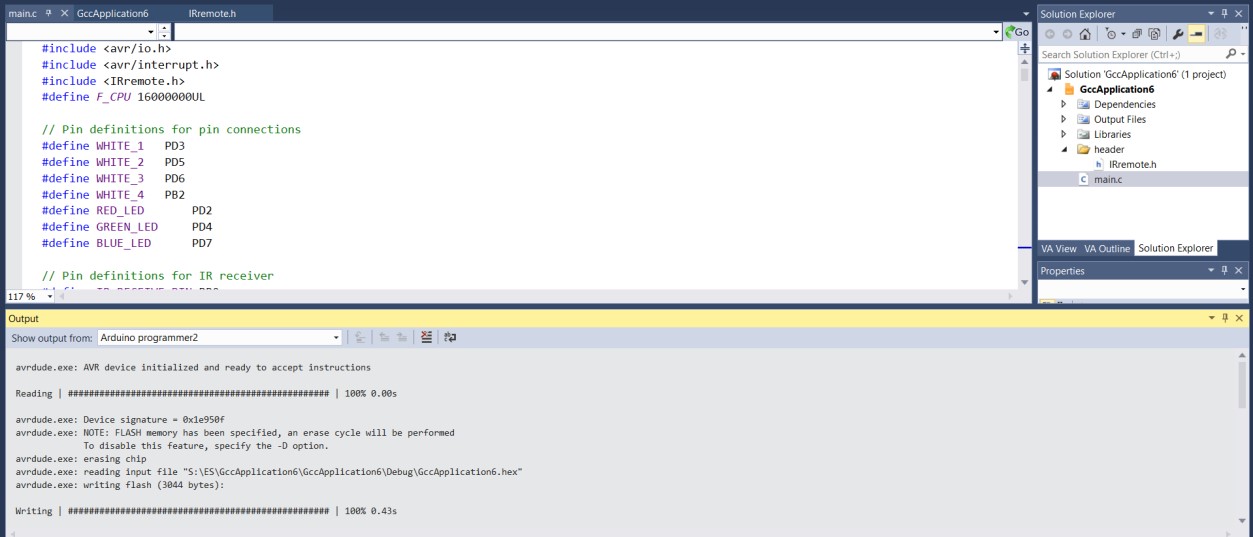
Solution Explorer -> Right-click on project name -> Add -> New folder -> Name your folder -> Copy library from your downloads -> Go to your project directory -> Paste library in your new folder -> Go to Project tab -> Click on project Properties -> Toolchain -> AVR/GNU C Compiler heading -> Directories subheading -> Click on icon with green plus -> Click on button with three dots -> Select the directory to your newly created folder -> Build project.

# 6.Atmel Studio Implementation

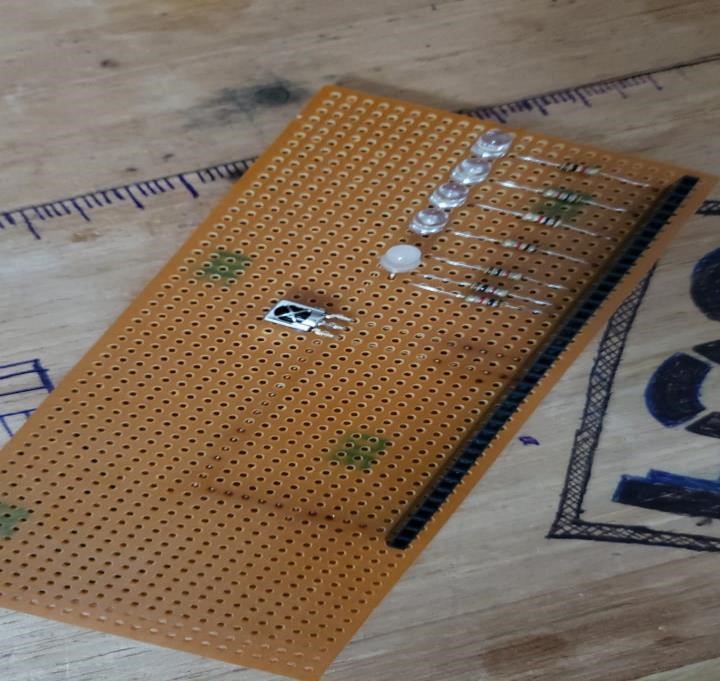


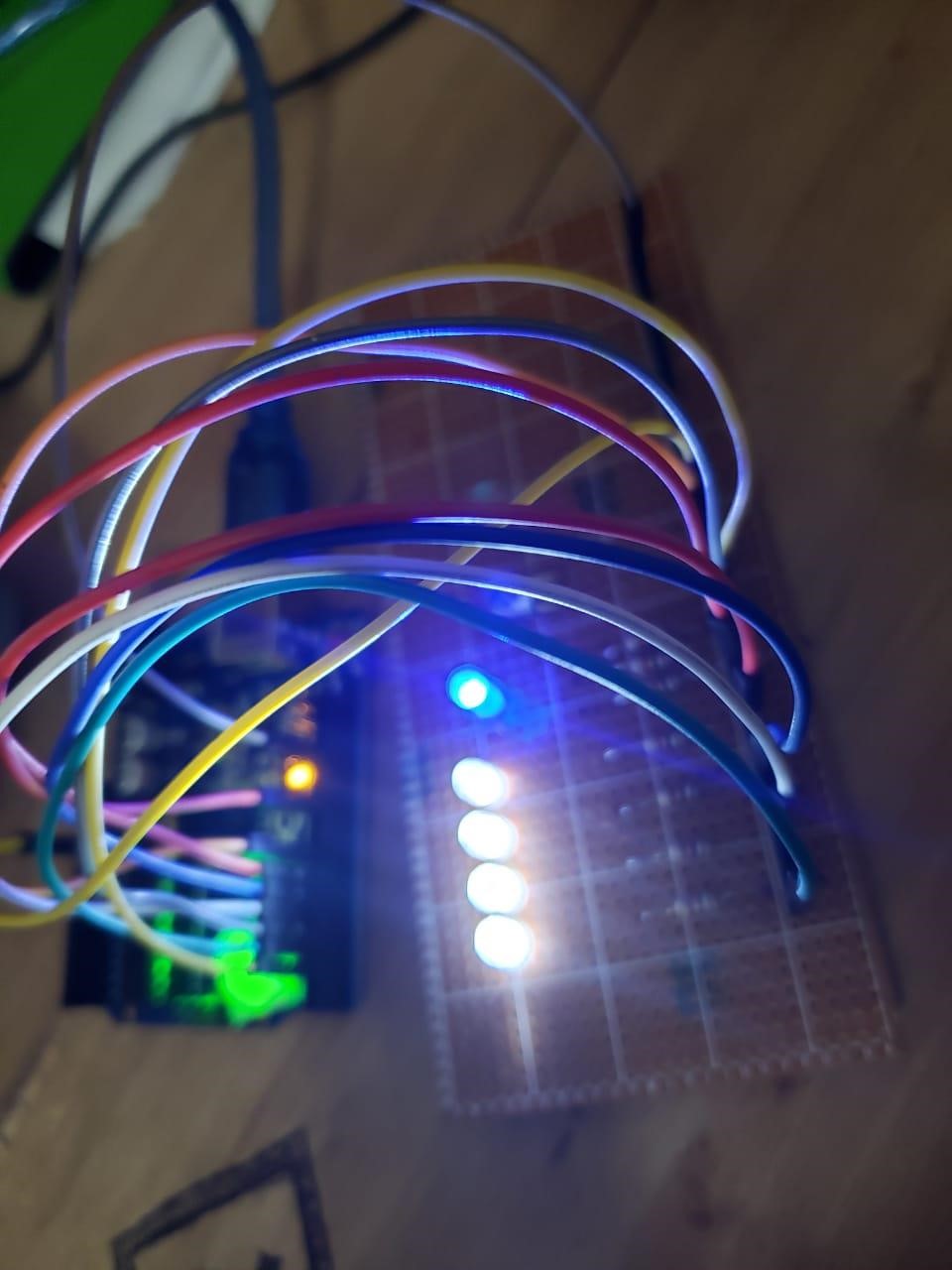


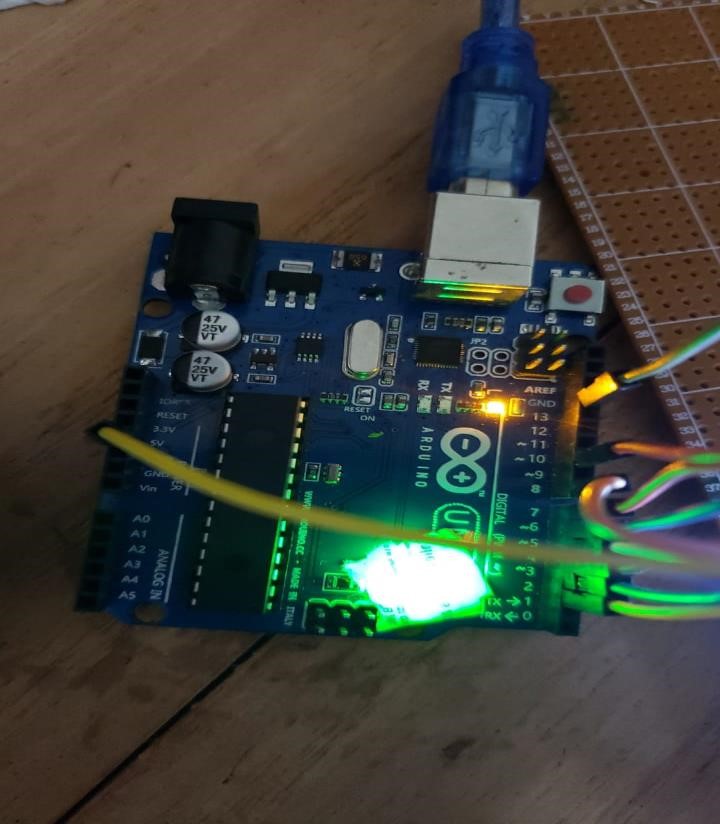




# 7.Circuit Functions







# 8.References

• [./src/libs/IRremote.h] : https://www.programming-electronics-diy.xyz/2022/08/ir- remote-control-library-for-avr.html