Introduction to robotics HW2
Traffic light prototype

Mantvydas Skruodys
2311158

October 6, 2025

Contents

1 Problem

We are creating a small prototype for a traffic light.

A pedestrian can press the button - this starts a countdown until the person can walk.

After the time is done - the person is shown for how long the light is going to be green.

The lights mean this:

- The red light means that the pedestrian has to wait.
- The red and yellow means that it will soon turn green, but you still have to wait.
- The green light means that the person can walk.

The number of total button clicks is counted and saved.

2 Design

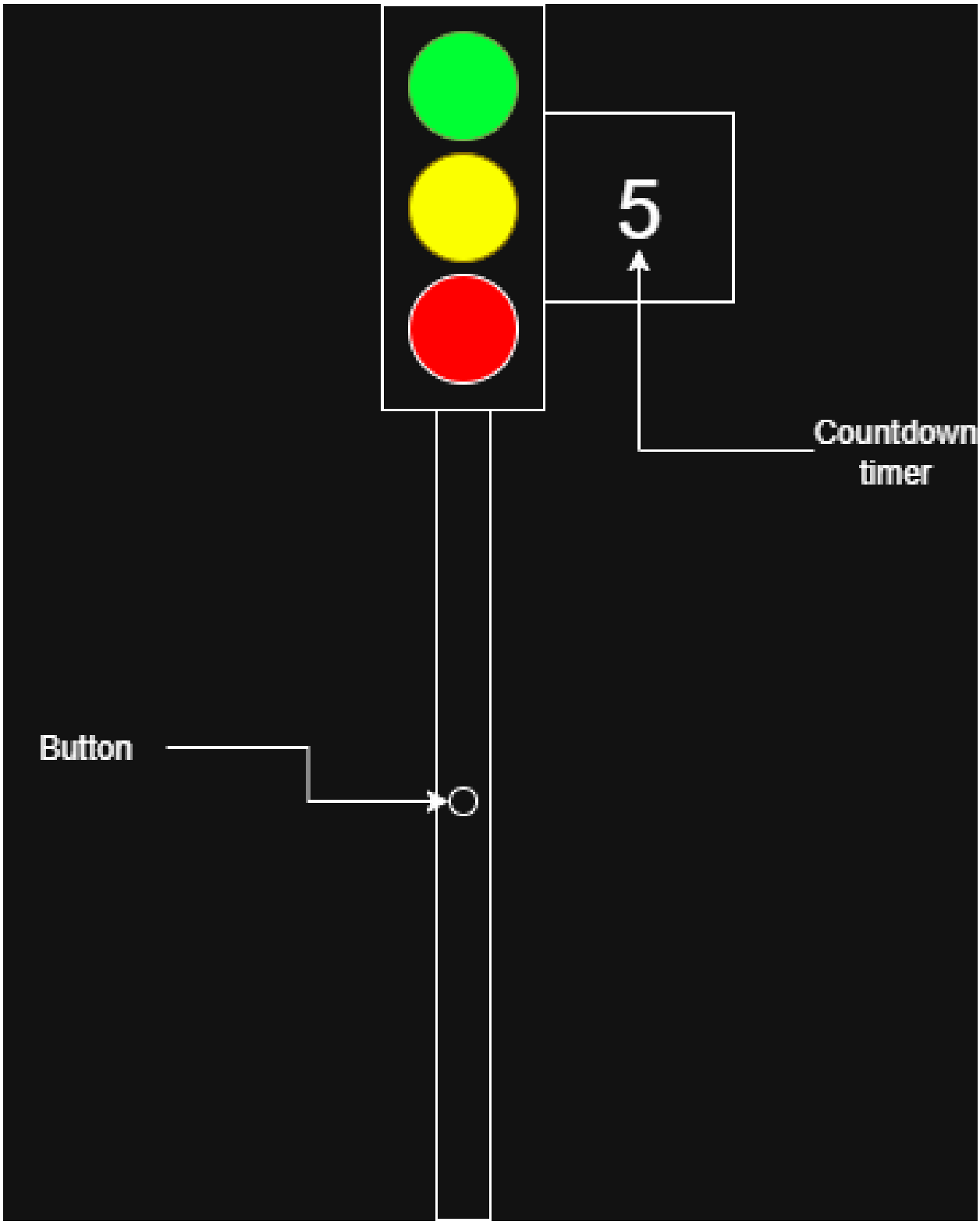


Figure 1: The instalation of the trafic light

3 Parts list

- 1x Arduino UNO
- 3x resistor(220Ω)

- 1x red LED
- 1x green LED
- 1x yellow LED
- 1x LCD screeen
- 1x I2C LCD Adapter
- wires

4 Wiring

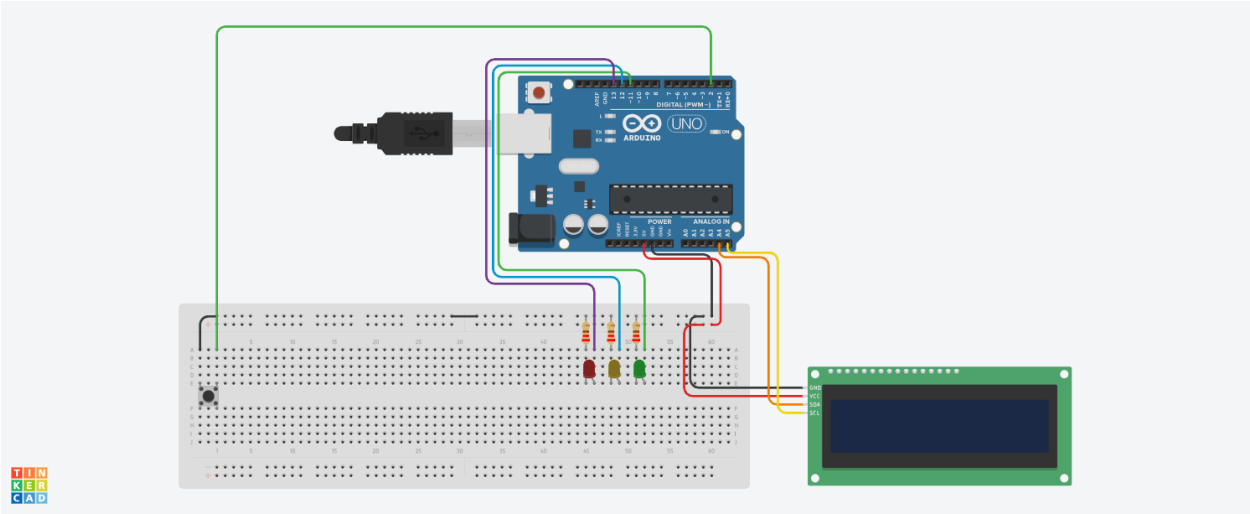
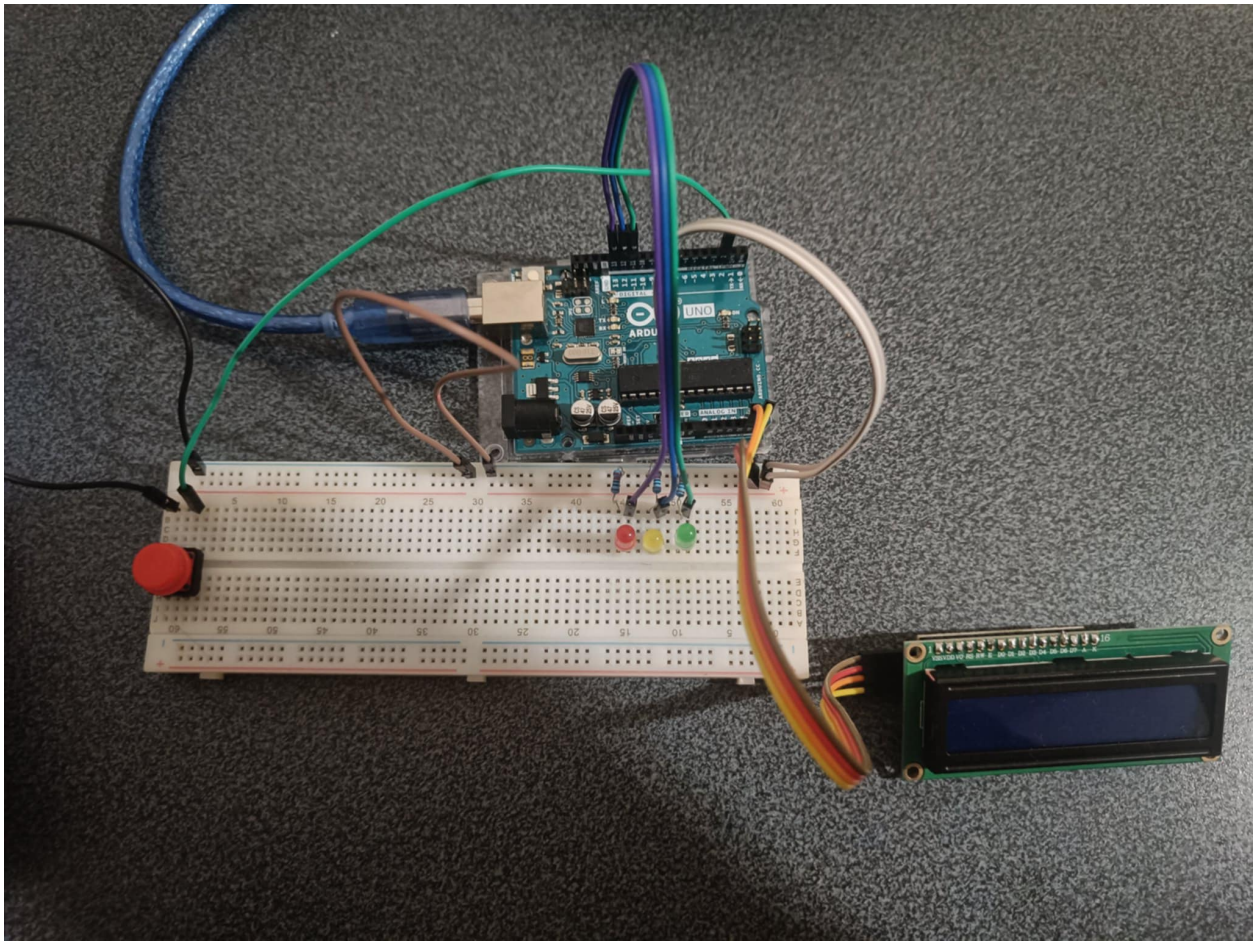


Figure 2: Arduino wiring

5 Photo of the robot



6 Main ideas for coding

The initial state of the system is when the button has not been pressed, the red light is on, and the display shows "Wait" and the press counter.

When the pedestrian button is pressed, we set the flag pressed = true and start a countdown from the configured total time (totalTime).

Each button press is stored permanently in EEPROM (pressCounter is saved).

The countdown has two phases:

- Red/Yellow phase: the red LED stays on, and the yellow LED turns on partway through this waiting period. The LCD shows "Wait" and the seconds remaining until walking is allowed.
- Green phase: the green LED turns on, the red/yellow LEDs turn off, and the LCD shows "Walk" with the seconds remaining to cross.

The countdown is driven by Timer1 configured in CTC mode, generating an interrupt every 1 second. Each interrupt decreases countdownSeconds.

When the countdown reaches zero, the system resets to its initial state: red LED on, LCD cleared, waiting for the next button press.

7 Arduino code

```
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const int redPin = 13;
const int yellowPin = 12;
const int greenPin = 11;
const int interruptPin = 2;

volatile bool pressed = false;
volatile bool tickFlag = false;
volatile int countdownSeconds = 0;

volatile bool buttonPressed = false;

int totalTime = 10;
int walkTime = totalTime / 2;

int VERSION = 1;

int pressCounter = 0;

void setup() {
  if (EEPROM.read(0) == VERSION) {
    totalTime = EEPROM.read(1);
    pressCounter = EEPROM.read(2);
  } else {
    EEPROM.update(0, VERSION);
    EEPROM.update(1, totalTime);
    EEPROM.update(2, pressCounter);
  }

  walkTime = totalTime / 2;

  lcd.init();
  lcd.backlight();

  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);

  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), toggleLights, FALL

  cli();//stop interrupts

  //set timer1 interrupt at 1Hz
```

```
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  // set compare match register for 1hz increments
  OCR1A = 15624;// = (16*10^6) / (1*1024) - 1 (must be <65536)
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS10 and CS12 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  sei();//allow interrupts
}

void loop() {
  if (buttonPressed) {
    buttonPressed = false;

    if (!pressed && millis() > 2000) {
      pressed = true;
      countdownSeconds = totalTime;

      pressCounter++;
      EEPROM.update(2, pressCounter);
    }
  }

  if (!pressed) {
    digitalWrite(redPin, HIGH);
    digitalWrite(yellowPin, LOW);
    digitalWrite(greenPin, LOW);

    lcd.setCursor(0, 0);
    lcd.print("Wait");

    lcd.setCursor(7, 0);
    lcd.print(pressCounter);
  } else {
    if (tickFlag) {
      tickFlag = false;

      if (countdownSeconds > walkTime) {
        digitalWrite(redPin, HIGH);
        if (countdownSeconds <= totalTime - walkTime / 2) {
          digitalWrite(yellowPin, HIGH);
        }
        digitalWrite(greenPin, LOW);

        lcd.setCursor(0, 0);
```

```
        lcd.print("Wait");

        lcd.setCursor(7, 0);
        lcd.print(pressCounter);

        lcd.setCursor(0, 1);
        lcd.print(countdownSeconds - walkTime);
        lcd.print("          ");
      }
      else if (countdownSeconds > 0) {
        digitalWrite(redPin, LOW);
        digitalWrite(yellowPin, LOW);
        digitalWrite(greenPin, HIGH);

        lcd.setCursor(0, 0);
        lcd.print("Walk");
        lcd.setCursor(0, 1);
        lcd.print(countdownSeconds);
        lcd.print("          ");
      }
      else {
        pressed = false;
        lcd.clear();
      }
    }
  }
}

void toggleLights() {
  buttonPressed = true;
}

ISR(TIMER1_COMPA_vect) {
  if (pressed && countdownSeconds > 0) {
    countdownSeconds--;
    tickFlag = true;
  }
}
```

8 Future improvements

- Add the lights for the cars
- add a button "on the other side of the road"

9 Git repository

`https://github.com/Humantvis/Arduino-trafficLight`

10 Demo Video

https://drive.google.com/file/d/1qA9mkeBKlTggDvnrXUO229TrsHAMfr8c/view?usp=drive_link