Arduino-Based Traffic Light System

Group 4: Vu Giang Nam, Tu Xuan Thinh, Huynh Tuan Phat, Bui Diep Anh
FPT University HCMC

I. Introduction

Ensuring safety at train crossings is of paramount importance in modern transportation infrastructure. Our project introduces an advanced train crossing traffic control system leveraging Arduino technology. This system goes beyond conventional traffic lights by incorporating features like precise timing intervals, remote control, sound alarms, and a barrier mechanism, enhancing safety and efficiency.

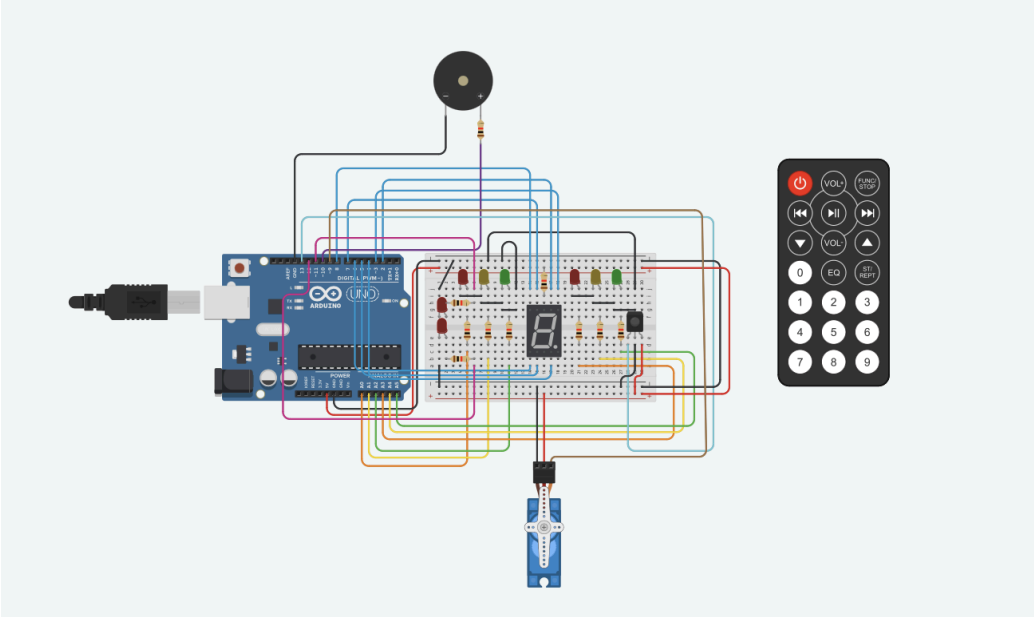
II. Main Proposal

Our project aims to develop and implement an Arduino-based train crossing traffic control system. This comprehensive endeavor encompasses system modeling, component selection, peripheral device integration, software programming, and meticulous planning via a programming flowchart.

III. Materials

- Arduino Uno
- LEDs (red, yellow and green)
- 220 ohm resistors
- Digit numbers display
- Breadboard
- Jumper wires
- Remote control
- Sound alarm
- Servo motor

IV. Circuit



The system operates through the following sequence:

- Upon activation, the system defaults to a green light. Simultaneously, the countdown timer initiates, starting from 9. The left light cluster illuminates green for 5 seconds, while the right light cluster shows red.
- As the countdown approaches 3, the right light cluster switches to a yellow light (displaying both yellow and green lights simultaneously).
- Upon reaching 1, the countdown resets to 9, and the left light cluster displays green, while the right light cluster shows red.
- Repeating this cycle, when the countdown reaches 3 again, the left light cluster illuminates yellow (displaying both yellow and green lights simultaneously).
- This cyclic process continues until the system is deactivated.

V. Method

The construction of the system involved meticulous steps:

- LEDs were connected to distinct Arduino digital pins with 220 ohm resistors to regulate current flow.
- The LCD display was interfaced with the Arduino using a breadboard, ensuring stable communication and data exchange.
- An Arduino sketch was meticulously crafted to orchestrate the LEDs, LCD, remote control, sound alarm, and barrier mechanism. This involved precise timing intervals and conditional statements to ensure seamless operation.
- Dual-directional control was implemented to ensure that when one light is green, the opposite is red, ensuring safe traffic flow.
- The LCD display accurately reflects the time remaining before the next light transition, providing valuable feedback to users.

VI. Code

The functionality of the traffic light system is achieved through a meticulously crafted Arduino sketch. The program employs timed delays to synchronize light transitions and update the countdown display on the LCD screen.

```
#include <Servo.h>
Servo myservo;

#include <IRremote.hpp>
// #include <IRremote.h> // thư viện hỗ trợ IR remote

#define IR_RECEIVE_PIN 13

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
```

```
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

#define segA 2  //connecting segment A to PIN2
#define segB 3  // connecting segment B to PIN3
#define segC 4  // connecting segment C to PIN4
#define segD 5  // connecting segment D to PIN5
#define segE 6  // connecting segment E to PIN6
#define segF 7  // connecting segment F to PIN7
#define segG 8  // connecting segment G to PIN1

const long bt0 = 3910598400;
const long bt1 = 4077715200;
const long bt2 = 3877175040;
const long bt3 = 2707357440;
const long bt4 = 4144561920;
const long bt5 = 3810328320;
const long bt6 = 2774204160;
const long bt7 = 3175284480;
const long bt8 = 2907897600;
const long bt9 = 3041591040;
const long btPlay = 3927310080;
const long btOnOff = 3125149440;

// Khai báo chân Servo
const int servoPin = 9;

// Khai báo chân buzzer
const int speakerPin = 10;//Chân được nối với loa hoặc buzzer

// danh sách các nốt nhạc
int melody[] = {
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5,
  NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5
  };

// thời gina các nốt nhạc: 4 = 1/4 nốt nhạc, 8 = 1/8nốt nhạc, ...:
int noteDurations[] = {
  2, 2, 2, 2, 2,
  2, 2, 2, 2, 2,
  2, 2, 2, 2, 2,
  2, 2, 2, 2, 2,
  2, 2, 2, 2, 2
  };

// Khai báo các chân điều khiển đèn tàu hỏa
const int trainL1 = 11;
const int trainL2 = 12;

// Khai báo các chân điều khiển cụm đèn
const int redLeft = A0;
const int yellowLeft = A1;
const int greenLeft = A2;
const int redRight = A3;
const int yellowRight = A4;
const int greenRight = A5;

// Khai báo biến đếm
int counter = 9;
int phase = 1;
int state = 1;

// Khai báo SevenSegment
// SevenSegment display(segA, segB, segC, segD, segE, segF, segG, segDP);

void setup() {
  // Khởi tạo các chân
  pinMode(redLeft, OUTPUT);
  pinMode(yellowLeft, OUTPUT);
  pinMode(greenLeft, OUTPUT);
  pinMode(redRight, OUTPUT);
  pinMode(yellowRight, OUTPUT);
  pinMode(greenRight, OUTPUT);

  for (int i = 2; i < 9; i++) {
    pinMode(i, OUTPUT);  // taking all pins from 2-8 as output
  }

  pinMode(trainL1, OUTPUT);
  pinMode(trainL2, OUTPUT);


  // Khởi tạo SevenSegment
  // display.begin();

  Serial.begin(9600);// serial
  IrReceiver.begin(IR_RECEIVE_PIN, DISABLE_LED_FEEDBACK); // start the IR receiver

  myservo.attach(servoPin);

}

void loop() {
  // Hiển thị số đếm trên màn hình LED 7-Segment
  // display.write(counter);

  if (IrReceiver.decode()) // nếu nhận được tín hiệu
  {
    translateIR();


Serial.println("---------------------------------------");

Serial.println(IrReceiver.decodedIRData.decodedRawData, HEX); // Print "old" raw data

Serial.println(IrReceiver.decodedIRData.decodedRawData, DEC); // Print "old" raw data
    IrReceiver.printIRResultShort(&Serial); // Print complete received data in one line
    IrReceiver.printIRSendUsage(&Serial);   // Print the statement required to send this data

    // Serial.println(results.value, HEX);
    delay(200);
    IrReceiver.resume(); // nhận giá trị tiếp theo
  }
}

void translateIR() {

switch(IrReceiver.decodedIRData.decodedRawData) {
    case bt1:
      phase = 1;
      while (state) {
        showTime();

        switch (phase) {
          case 1:
            digitalWrite(yellowLeft, LOW);
            digitalWrite(greenLeft, LOW);
            digitalWrite(redRight, LOW);
            digitalWrite(redLeft, HIGH);
            digitalWrite(greenRight, HIGH);
            break;
          case 7:
            digitalWrite(yellowRight, HIGH);
            break;
          case 9:
            counter = 10;
            break;
          case 10:
            digitalWrite(redLeft, LOW);
```

```
        digitalWrite(greenRight, LOW);
        digitalWrite(yellowRight, LOW);
        digitalWrite(greenLeft, HIGH);
        digitalWrite(redRight, HIGH);
        break;
       case 16:
        digitalWrite(yellowLeft, HIGH);
        break;
       case 18:
        phase = 0;
        counter = 10;
        break;
      }

      // Giảm giá trị biến đếm
      counter--;
      phase++;

      delay(1000);
     }

     break;
    case bt2:
     int noteDuration, pos = 80;
     counter = 5;
     phase = 1;
     digitalWrite(redLeft, HIGH);
     digitalWrite(redRight, HIGH);
     digitalWrite(greenRight, LOW);
     digitalWrite(greenLeft, LOW);
     for (int i = 0; i < 25; i++) {

      digitalWrite(trainL1, HIGH);
      digitalWrite(trainL2, LOW);

      showTime();

      noteDuration = 1000/noteDurations[i];
      tone(speakerPin,
melody[i],noteDuration);

      if (phase == 2) {
       myservo.write(pos += 10);
      } else if (phase == 5) {
       myservo.write(pos -= 10);
      }
      delay(500);

      digitalWrite(trainL1, LOW);
      digitalWrite(trainL2, HIGH);

      tone(speakerPin,
melody[i],noteDuration);

      if (phase == 2) {
       myservo.write(pos += 10);
      } else if (phase == 5) {
       myservo.write(pos -= 10);
      }

      delay(500);

      counter--;
      if (counter == 0) {
       counter = 5;
       phase++;
      }
     }
     noTone(speakerPin);
     digitalWrite(trainL1, LOW);
     digitalWrite(trainL2, LOW);
     digitalWrite(redLeft, LOW);
     digitalWrite(redRight, LOW);
     digitalWrite(greenRight, HIGH);
     digitalWrite(greenLeft, HIGH);
     counter = 0;
     showTime();
     break;
  }
}

void showTime() {
  switch (counter) {
    case 0:  //when count value is zero show"0"
on disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, LOW);
     digitalWrite(segE, LOW);
     digitalWrite(segF, LOW);
     digitalWrite(segG, HIGH);
     break;
    case 1:  // when count value is 1 show"1" on
disp
     digitalWrite(segA, HIGH);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, HIGH);
     digitalWrite(segE, HIGH);
     digitalWrite(segF, HIGH);
     digitalWrite(segG, HIGH);
     break;
    case 2:  // when count value is 2 show"2" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, LOW);
     digitalWrite(segC, HIGH);
     digitalWrite(segD, LOW);
     digitalWrite(segE, LOW);
     digitalWrite(segF, HIGH);
     digitalWrite(segG, LOW);
     break;
    case 3:  // when count value is 3 show"3" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, LOW);
     digitalWrite(segE, HIGH);
     digitalWrite(segF, HIGH);
     digitalWrite(segG, LOW);
     break;
    case 4:  // when count value is 4 show"4" on
disp
     digitalWrite(segA, HIGH);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, HIGH);
     digitalWrite(segE, HIGH);
     digitalWrite(segF, LOW);
     digitalWrite(segG, LOW);
     break;
    case 5:  // when count value is 5 show"5" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, HIGH);
     digitalWrite(segC, LOW);
     digitalWrite(segD, LOW);
     digitalWrite(segE, HIGH);
     digitalWrite(segF, LOW);
     digitalWrite(segG, LOW);
     break;
    case 6:  // when count value is 6 show"6" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, HIGH);
     digitalWrite(segC, LOW);
     digitalWrite(segD, LOW);
     digitalWrite(segE, LOW);
     digitalWrite(segF, LOW);
     digitalWrite(segG, LOW);
     break;
    case 7:  // when count value is 7 show"7" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, HIGH);
     digitalWrite(segE, HIGH);
     digitalWrite(segF, HIGH);
     digitalWrite(segG, HIGH);
     break;
    case 8:  // when count value is 8 show"8" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, LOW);
     digitalWrite(segE, LOW);
     digitalWrite(segF, LOW);
     digitalWrite(segG, LOW);
     break;
    case 9:  // when count value is 9 show"9" on
disp
     digitalWrite(segA, LOW);
     digitalWrite(segB, LOW);
     digitalWrite(segC, LOW);
     digitalWrite(segD, LOW);
     digitalWrite(segE, HIGH);
     digitalWrite(segF, LOW);
     digitalWrite(segG, LOW);
     break;
    break;
  }
}
```

VII. Results

The system functions flawlessly, emulating a real train crossing traffic control system. The lights transition seamlessly between red, yellow, and green, adhering to predefined timing sequences. The LCD effectively displays the countdown timer for each phase, enhancing user comprehension and interaction.

VIII. Discussion

Our advanced train crossing traffic control system offers a scalable and robust solution for enhancing safety at train crossings. Its integration of remote control, sound alarms, and a barrier mechanism elevates safety standards, making it suitable for deployment in diverse environments, from educational demonstrations to real-world applications.

IX. Conclusion

In conclusion, our project signifies a significant leap forward in train crossing safety measures. By leveraging Arduino technology and integrating advanced features, we've created a system that provides not only efficient traffic control but also enhanced safety and convenience. Future iterations could further expand the system's capabilities, potentially integrating pedestrian crossing lights or implementing wireless communication for remote monitoring.