VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MICROPROCESSORS - MICROCONTROLERS

Report

Traffic Light

Advisor(s):	Nguyễn Thiên Ân	
Student(s):	Nguyễn Huy Tài	ID 2110513
	Bùi Quốc Bảo	ID 2152411
	Nguyễn Minh Đăng	ID 2252154

HO CHI MINH CITY, DECEMBER 2024

Contents

1 Introduction **5**
1.1 Set up 5
1.2 Simulation 6

2 Implementation **8**
2.1 Timer Interrupt 8
2.1.1 Code 8
2.2 Button 9
2.2.1 Code 10
2.3 LCD Display 11
2.4 FSM Traffic Light 11
2.4.1 Code 11
2.5 FSM MODE of Traffic Light 16
2.5.1 Code 16

3 Result **19**

4 Source code **21**

List of Figures

1.1 Setup PIN on ioc 5
1.2 Set up timmer 5
1.3 Set up timmer 5
1.4 Set up timmer 5
1.5 Setup pull-up 6
1.6 STM32F103RB (extension) 6
1.8 Button controlled by 2 bits 7
1.7 Traffic Light controlled by 2 bits 7
1.9 LCD 8
2.1 Bounce 9
3.1 The system is in a normal mode 19
3.2 Mode 2: Red - before modifying 20
3.3 Mode 2: Red - after modifying 20
3.4 Mode 3: Yellow, which functions are similar mode 2 20

3.5 Mode 4: Yellow, which functions are similar mode 2 21
3.6 The result 21

List of Tables

Listings

1 Introduction

Traffic lights are necessary in our lives, which controls traffic in all over the world. Therefore, my team decides to implement & simulation how the traffic light works and how we can control it using the buttons.

1.1 Set up

We use STM32CubeIDE software to implement these tasks.

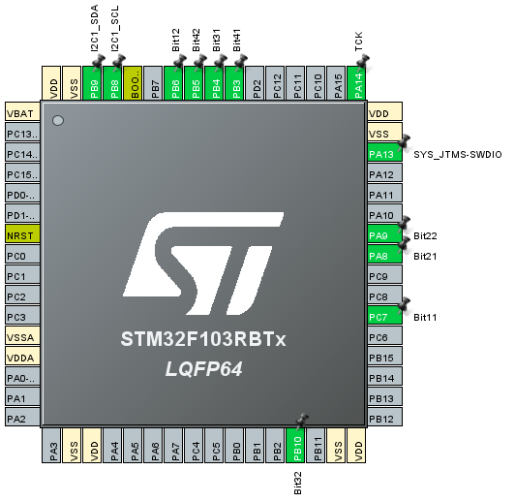


Figure 1.1: Setup PIN on ioc

First of all, we must set up the timer interrupt as follows:

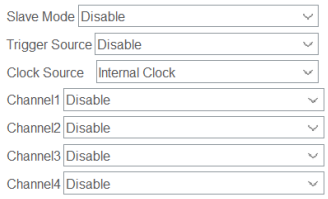


Figure 1.2: Set up timmer

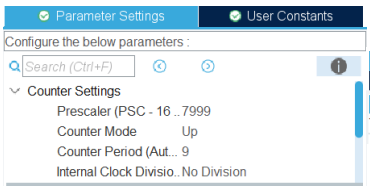


Figure 1.3: Set up timmer

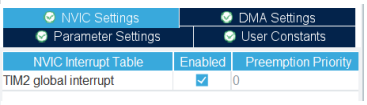
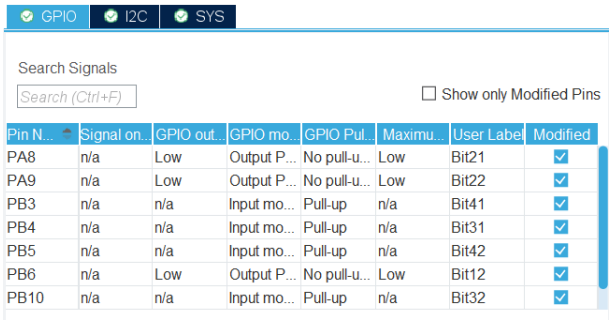


Figure 1.4: Set up timmer

Secondly, we setup the buttons:



GPIO | I2C | SYS

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin N...	Signal on...	GPIO out...	GPIO mo...	GPIO Pul...	Maximu...	User Label	Modified
PA8	n/a	Low	Output P...	No pull-u...	Low	Bit21	☑
PA9	n/a	Low	Output P...	No pull-u...	Low	Bit22	☑
PB3	n/a	n/a	Input mo...	Pull-up	n/a	Bit41	☑
PB4	n/a	n/a	Input mo...	Pull-up	n/a	Bit31	☑
PB5	n/a	n/a	Input mo...	Pull-up	n/a	Bit42	☑
PB6	n/a	Low	Output P...	No pull-u...	Low	Bit12	☑
PB10	n/a	n/a	Input mo...	Pull-up	n/a	Bit32	☑

Figure 1.5: Setup pull-up

We need to set up a pull-up for 3 buttons on PB3, PB4, PB5 pins to avoid high-impedence or floating status.

- When the switch is open, it has no impact on the port pin. An internal resistor on the port “pulls up" the pin to the supply voltage of the MCU (typically 3.3V for STM32F103RB). If we read the pin, we will see the value ‘1’.
- When the switch is closed (pressed), the pin voltage will be 0V. If we read the pin, we will see the value ‘0’.

1.2 Simulation

We use some devices in the following to simulate a system of traffic lights for the crossroad.

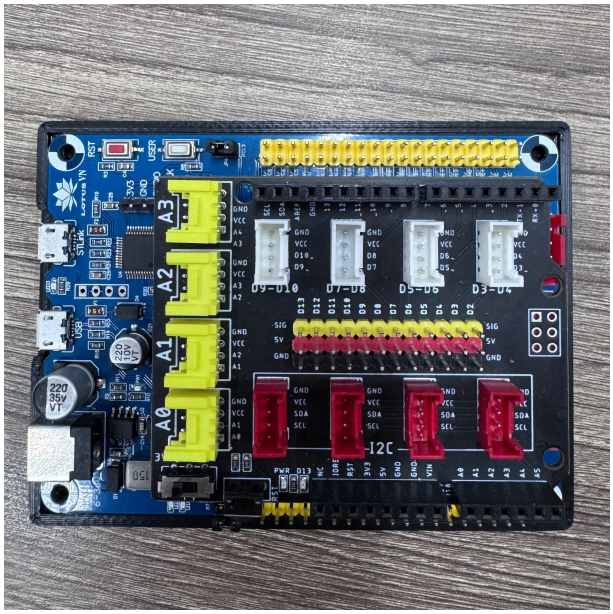


Figure 1.6: STM32F103RB (extension)

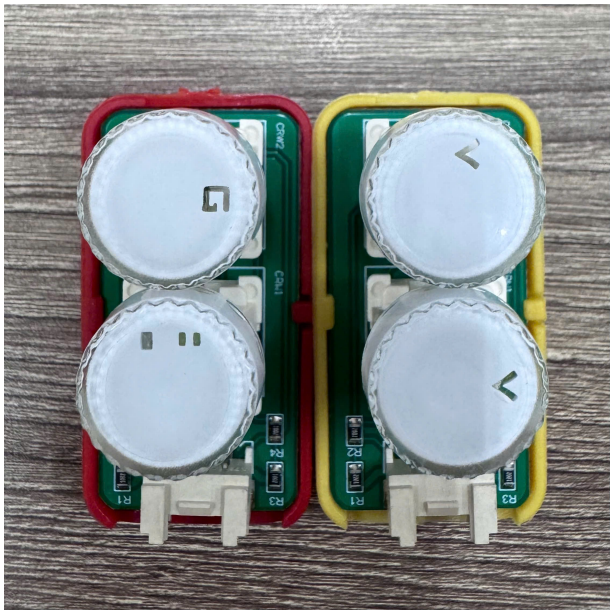


Figure 1.8: Button controlled by 2 bits

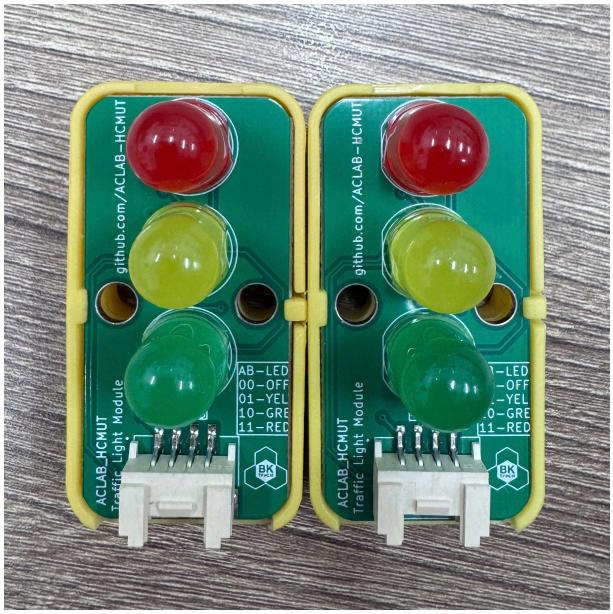


Figure 1.7: Traffic Light controlled by 2 bits

Figure 1.9: LCD

2 Implementation

2.1 Timer Interrupt

2.1.1 Code

```
void timerRun()
{
  counterTime[0] --;
  counterTime[1] --;
  counterTime[2] --;
  counterTime[4] --;
  if (counterTime[0] == 0)
  {
    flag[0] = 1;
  }

  if (counterTime[1] == 0)
  {
    flag[1] = 1;
  }
```

```

  if (counterTime[2] == 0)
  {
    flag[2] = 1;
  }

  if (counterTime[4] == 0)
  {
    flag[4] = 1;
  }
}
```

And we setup the function into a call back loop in the main.c.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  button_reading();
  timerRun();
}
```

2.2 Button

In practice, all mechanical switch contacts bounce (turn on, turn off, hold,... for a short time) after the switch closed or opened as follows figure:

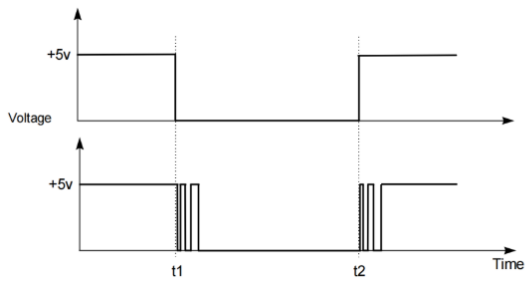


Figure 2.1: Bounce

As far as the MCU concerns, each “bounce" is equivalent to one press and release of an “ideal” switch. Therefore, there are some problems:

- We may read 'AAAAA' rather than 'A' for once times pressed in a short time.
- It is difficult to count the number of times that button is pressed.
- When a switch is pressed and then released after some time, the bouncing effect may cause it to register as though the switch was pressed again at the moment of release.

We know some milestone of time in following:

- Bounce time: most buttons seem stop bouncing within 10ms.
- Button pressed time: the shortest time that user can press and release between 50 and 100ms.
- Response time: the user can notices if the response is 100ms and not if it under 50ms.

2.2.1 Code

Here the code to read input and debounce the buttons:

```
void button_reading(void) {
    for (char i = 0; i < NO_OF_BUTTONS; ++i) {
        debounceButtonBuffer1[i] = debounceButtonBuffer2[i];
        debounceButtonBuffer2[i] = HAL_GPIO_ReadPin(port[i], pinsButtons[i]);

        if (debounceButtonBuffer1[i] == debounceButtonBuffer2[i]) {
            buttonBuffer[i] = debounceButtonBuffer1[i];

            if (buttonBuffer[i] == BUTTON_IS_PRESSED) {
                // If a button is pressed, we start counting
                if (counterForButtonPress1s[i] < DURATION_FOR_AUTO_INCREASING) {
                    counterForButtonPress1s[i]++;
                } else {
                    // The flag is turned on when 1 second has passed since the button is pressed.
```

```
                    flagForButtonPress1s[i] = 1;
                    // todo
                }
            } else {
                counterForButtonPress1s[i] = 0;
                flagForButtonPress1s[i] = 0;
            }
        }
    }
}
```

And we setup the function into a call back loop in the main.c

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  button_reading();
  timerRun();
}
```

2.3 LCD Display

2.4 FSM Traffic Light

We change the led when its duration run out.
The default of duration of each leds:

- Red led: 10s
- Yellow led: 3s
- Green led: 7s

After changed the led, we have had update the buffer to display the number on LCD.

2.4.1 Code

```c
void fsm_traffic_light()
{
    switch (state)
    {
        case 0:
            HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 1);
            HAL_GPIO_WritePin(Bit12_GPIO_Port, Bit12_Pin, 1);
            HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 1);
            HAL_GPIO_WritePin(Bit22_GPIO_Port, Bit22_Pin, 0);
            updateDigitBuffer(counterLightBuffer[0] - countTemp, counterLightBuffer[2] - countTemp, 0, 0);
            if (countTemp == counterLightBuffer[2])
            {
                state = 1;
                countTemp = 0;
            }
            break;

        case 1:
            HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 1);
            HAL_GPIO_WritePin(Bit12_GPIO_Port, Bit12_Pin, 1);
            HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 0);
            HAL_GPIO_WritePin(Bit22_GPIO_Port, Bit22_Pin, 1);
            updateDigitBuffer(counterLightBuffer[1] - countTemp, counterLightBuffer[1] - countTemp, 0, 0);

            if (countTemp == counterLightBuffer[1])
            {
                state = 2;
                countTemp = 0;
            }
            break;

        case 2:
            HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 1);
```

```
            HAL_GPIO_WritePin(Bit12_GPIO_Port, Bit12_Pin, 0);
            HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 1);
            HAL_GPIO_WritePin(Bit22_GPIO_Port, Bit22_Pin, 1);

            updateDigitBuffer(counterLightBuffer[2] - countTemp, counterLightBuffer[0] - countTemp, 0, 0);

            if (countTemp == counterLightBuffer[2])
            {
                state = 3;
                countTemp = 0;
            }
            break;

        case 3:
            HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 0);
            HAL_GPIO_WritePin(Bit12_GPIO_Port, Bit12_Pin, 1);
            HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 1);
            HAL_GPIO_WritePin(Bit22_GPIO_Port, Bit22_Pin, 1);
            updateDigitBuffer(counterLightBuffer[1] - countTemp, counterLightBuffer[1] - countTemp, 0, 0);

            if (countTemp == counterLightBuffer[1])
            {
                state = 0;
                countTemp = 0;
            }
            break;

        default:
            state = 0;
            break;
    }

    if (flag[0] == 1)
```

```
    {
        countTemp ++;
        setTimer1(100);
        lcd_clear_display();
    }

    return;
}

/*
*
*
*/

void updateDigitBuffer(int duration1, int duration2, int mode
   , enum Mode *mode_case)

 (mode == 0)

lcd_goto_XY(0, 0);
lcd_send_string("Lane 1: ");
lcd_send_num(duration1);
lcd_goto_XY(1, 0);
lcd_send_string("Lane 2: ");
lcd_send_num(duration2);


 (mode == 1)

lcd_goto_XY(0, 0);
lcd_send_string("Mode: ");
lcd_send_num(duration2);
switch(duration2)
{
  case 2:
```

```
    lcd_send_string(" - RED");
    break;

  case 3:
    lcd_send_string(" - YELLOW");
    break;

  case 4:
    lcd_send_string(" - GREEN");
    break;

  default:
    break;
}

lcd_goto_XY(1, 0);
lcd_send_string("MOD: ");
lcd_send_num(duration1);
lcd_goto_XY(1, 8);
lcd_send_string("SET: ");
if (mode_case == 1)
{
  lcd_send_num(counterLightBuffer[0]);
}
else if (mode_case == 2)
{
  lcd_send_num(counterLightBuffer[1]);
}
else if (mode_case == 3)
{
  lcd_send_num(counterLightBuffer[2]);
}
```

2.5 FSM MODE of Traffic Light

We have 4 modes:

- Mode 1: we can change the duration of the red led on 2 road by pressing the button 2 and save the value by pressing button 3.
- Mode 2: we can change the duration of the yellow led on 2 road by pressing the button 2 and save the value by pressing button 3.
- Mode 3: we can change the duration of the green led on 2 road by pressing the button 2 and save the value by pressing button 3.

Note: we modify the duration of each leds but we must guarantee the duration of red led had to equal the total of duration of yellow led and green led.

2.5.1 Code

```
void fsm_for_input_processing(void) {
    for (int i = 0; i < NO_OF_BUTTONS; i++)
    {
        detect_signal_button(&buttonStates[i], i);  // Pass button state and pin number
    }
    switch(mode) {
        case MODE1:
            if (counterLightBuffer[0] != counterLightBuffer[1]+counterLightBuffer[2])
            {
                counterLightBuffer[0] = counterLightBuffer[1]+counterLightBuffer[2];
            }
            if (previous_mode == 4)
            {
                lcd_clear_display();
            }

            fsm_traffic_light();
```

```
            if (flag[1] == 1)
            {
            setTimer_blinkly(50);
            }
            previous_mode = 1;
            setTimer_blinkly(50);
            setCounter[0] = counterLightBuffer[0];
            setCounter[1] = counterLightBuffer[1];
            setCounter[2] = counterLightBuffer[2];
            break;

        case MODE2:
            if (previous_mode == 1)
            {
                lcd_clear_display();
                HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 1);
                HAL_GPIO_WritePin(Bit12_GPIO_Port, Bit12_Pin, 1);
                HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 1);
                HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit22_Pin, 1);
            }
            if (flag[1] == 1)
            {
                HAL_GPIO_TogglePin(Bit11_GPIO_Port, Bit11_Pin);
                HAL_GPIO_TogglePin(Bit12_GPIO_Port, Bit12_Pin);
                HAL_GPIO_TogglePin(Bit21_GPIO_Port, Bit21_Pin);
                HAL_GPIO_TogglePin(Bit22_GPIO_Port, Bit22_Pin);
```

```
                setTimer_blinkly(50);
            }
            updateDigitBuffer(setCounter[0], 02, 1, 1);

            previous_mode = 2;
            break;

        case MODE3:
            if (previous_mode == 2)
            {
                lcd_clear_display();
            }
            HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 0);
            HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 0);
            if (flag[1] == 1)
            {
                HAL_GPIO_TogglePin(Bit12_GPIO_Port, Bit12_Pin);
                HAL_GPIO_TogglePin(Bit22_GPIO_Port, Bit22_Pin);
                setTimer_blinkly(50);
            }
            updateDigitBuffer(setCounter[1], 03, 1, 2);
            previous_mode = 3;
            break;

        case MODE4:
            if (previous_mode == 3)
            {
                    lcd_clear_display();
                    HAL_GPIO_WritePin(Bit22_GPIO_Port, Bit22_Pin, 0);
                    HAL_GPIO_WritePin(Bit12_GPIO_Port, Bit12_Pin, 0);
```

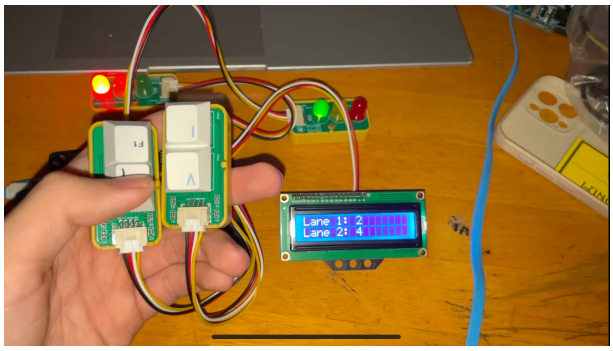


Figure 3.1: The system is in a normal mode

```
                    HAL_GPIO_WritePin(Bit11_GPIO_Port, Bit11_Pin, 0);
                    HAL_GPIO_WritePin(Bit21_GPIO_Port, Bit21_Pin, 0);
            }
            if (flag[1] == 1)
            {
                HAL_GPIO_TogglePin(Bit21_GPIO_Port, Bit21_Pin);
                HAL_GPIO_TogglePin(Bit11_GPIO_Port, Bit11_Pin);
                setTimer_blinkly(50);
            }
            reset_state();

            updateDigitBuffer(setCounter[2], 04, 1, 3);
            previous_mode = 4;
            break;
    }
```

3 Result

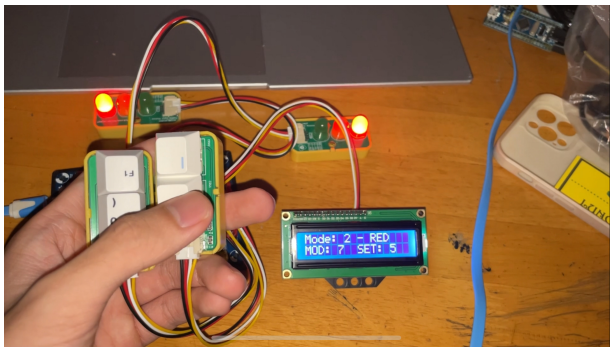


Figure 3.2: Mode 2: Red - before modifying

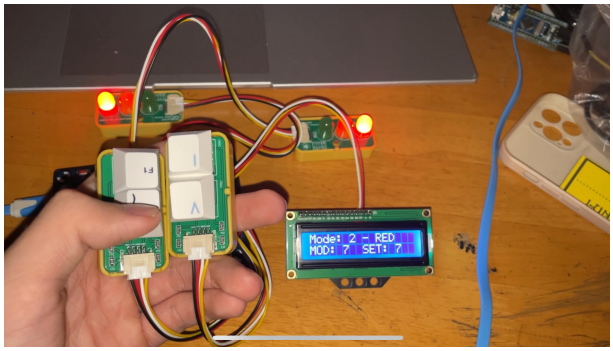


Figure 3.3: Mode 2: Red - after modifying

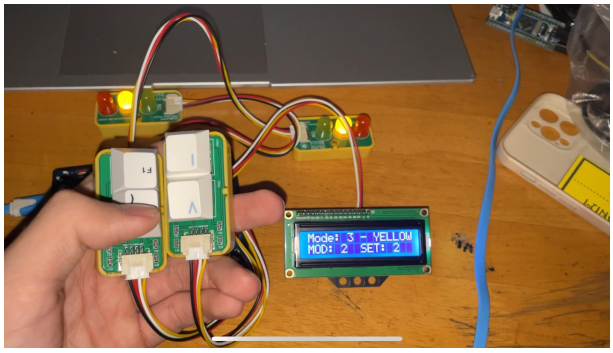


Figure 3.4: Mode 3: Yellow, which functions are similar mode 2

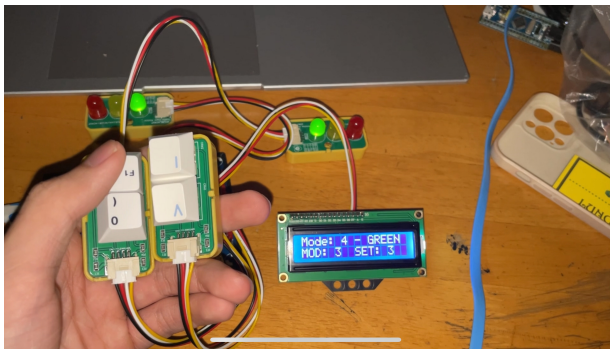


Figure 3.5: Mode 4: Yellow, which functions are similar mode 2

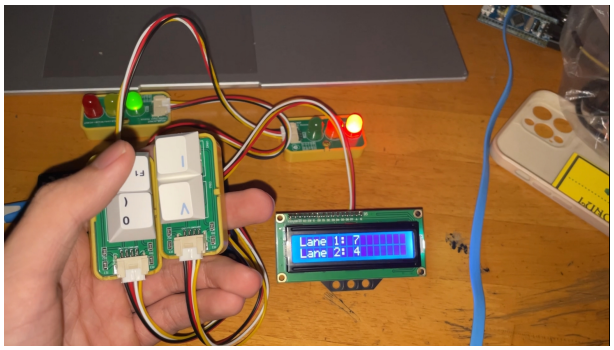


Figure 3.6: The result

4 Source code

You can view the source code on GitHub: My GitHub Project.

You can view the our result on Youtube: My video.