JOINT INSTITUTE
交大密西根学院

Eyesight Test Device

VE373 Project Report

Group 1: Bosen Cheng, Yaxin Chen

Introduction

This project aims to build an eyesight test device prototype, which measures the eyesight in a traditional way. Namely, the device uses a LCD screen to display the symbol “E” instead of an actual chart, and uses a LiDAR and servo to detect the hand position and determine whether the answer is correct.

The motivation for this project is to implement an embedded system that can solve a real-world problem. Nowadays, an embedded system can solve a problem in an efficient way with a relatively low cost. This project explores the implementation of a basic level embedded system that can help us familiar with the usage of embedded programming skills such as UART, LiDAR, PWM and ADC.

Design

The device consists of three main parts: display, processor, and sensor. The display is an LCD screen communicate with the processor PIC32 directly. The PIC32 sends the data that contains the testing symbol “E” with different directions and the corresponding eyesight degree.

The eyesight test device’s main challenge is to build the hand position detection system using a combination of a time-of-flight (TOF) sensor and a servo. The sensor uses a servo to allow the LiDAR to scan a 2D plane, where user shows his/her gesture. The servo is controlled by PIC32 through PWM. With the PWM signal sent periodically, the servo rotates periodically so that the LiDAR can scan the plane within one period. Besides, we also use the slide rheostat from Lab 6, and implement the voltage meter using ADC. It controls the delay between LCD display and the start of LiDAR scanning. The LiDAR communication with the PIC32 through UART. Notice that there will be certain distance between the LCD screen and the LiDAR (user’s hand). The top-level diagram is shown in Figure 1.

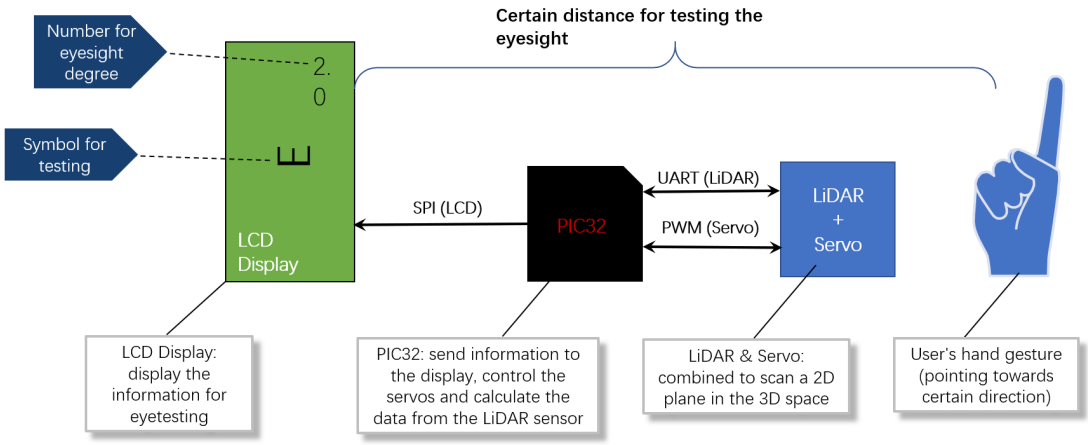

Figure 1. Top-level Design.

Software

Main Function

The main function follows the flow chart in Figure 2. At the starting stage, we initialize all the interrupts (PWM, UART, ADC, SPI) and timers and set up all the components. Then we signal the LCD screen to display the testing symbol “E”. There will be certain delay for user to decide his/her answer to the current problem. The delay time is controlled by a slide rheostat using ADC. Then we signal the servo motor and the LiDAR at the same time, the servo starts to rotate, and the LiDAR starts to scan. This scanner rotates from 0 degree to 120 degrees and back to 0 degree. Meanwhile, the data is collected in the main program. We calculate the position with the data. The logic to determine the hand position here is as following:

1. The angle of the PWM is divided into 3 three directions (Left, Middle, Right)
2. Initialize an array Pos = [0,0,0,0] to store the information of four position (left, up, down, right)
3. For the collected data:
4. If the data corresponds to the Left direction and the distance value is small
5. Pos[0] ++
6. If the data corresponds to the Right direction and the distance value is small
7. Pos[3] ++
8. If the data corresponds to the Middle direction and the distance value is small
9. Pos[2] ++
10. If the data corresponds to the Middle direction and the distance value is large
11. Pos[1] ++
12. End for
13. Check the largest value int the Pos array and output its corresponding position

After we get the calculated position, we compare it with the direction of the symbol displayed. If they match, we decrease the size of the “E” and repeat the procedure. Otherwise, the program will be stopped and stay with the same size of “E”, when user is ready, he/she can click the button on the PIC32 to repeat the exactly the same procedure.

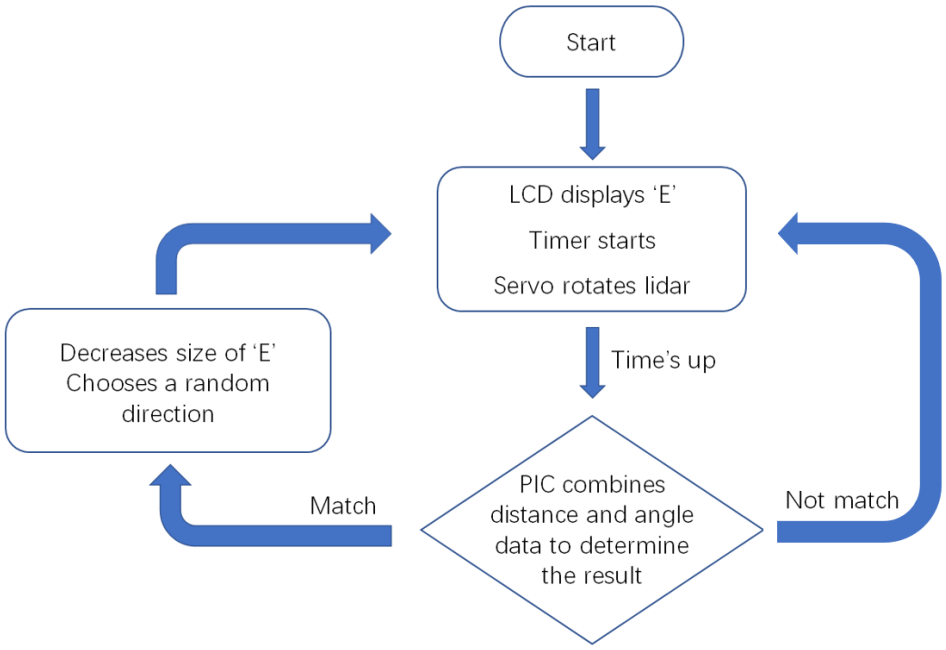

Figure 2. Program Flow Chart.

Hardware

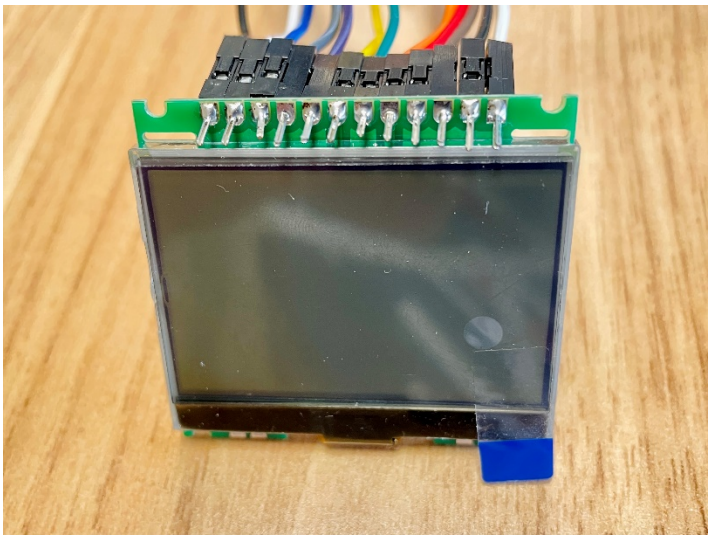

Figure 3. LCD Display

The LCD screen we used is JLX12964G-086-P. We send the symbol “E” with four different directions and three different sizes through SPI connection from the PIC32 processor. To customize the graph for

displaying (“E” in our case), we need to specify a matrix (array of char) to determine the binary value for each pixel on the screen.

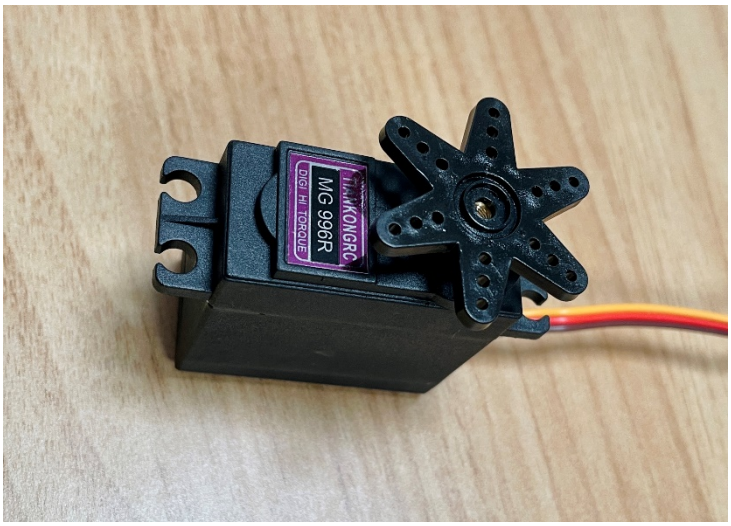

Figure 4. Servo Motor

The servo motor we use in our project is MG996R. The voltage it uses is 5V. As for the PWM signal input, it takes signal with frequency 50Hz (period 20ms). The duty cycle varies from 2.5% (0.5ms) to 12.5% (2.5ms). The corresponding angle varies from 0 degree to 180 degrees.

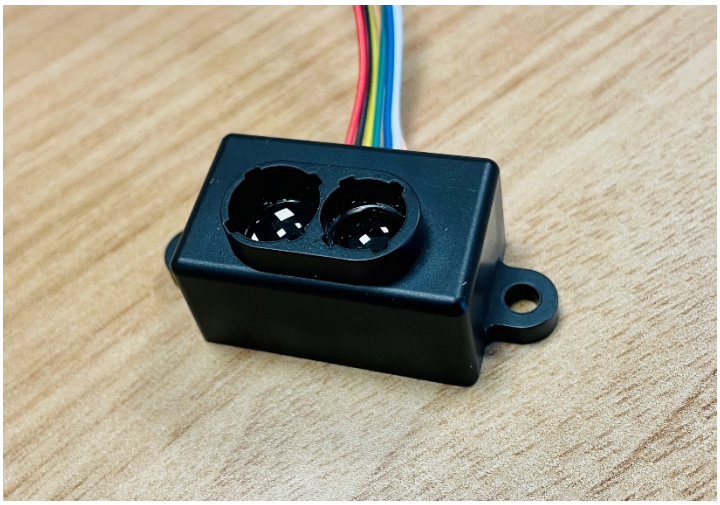

Figure 5. LiDAR

The TOF LiDAR we use in our project is MVR2EA. The sample rate is 50Hz. The PIC32 communicate with the LiDAR through UART. We need to send instruction to start the LiDAR and stop it, when the LiDAR is running, the data will be automatically sent back to the PIC32 board.

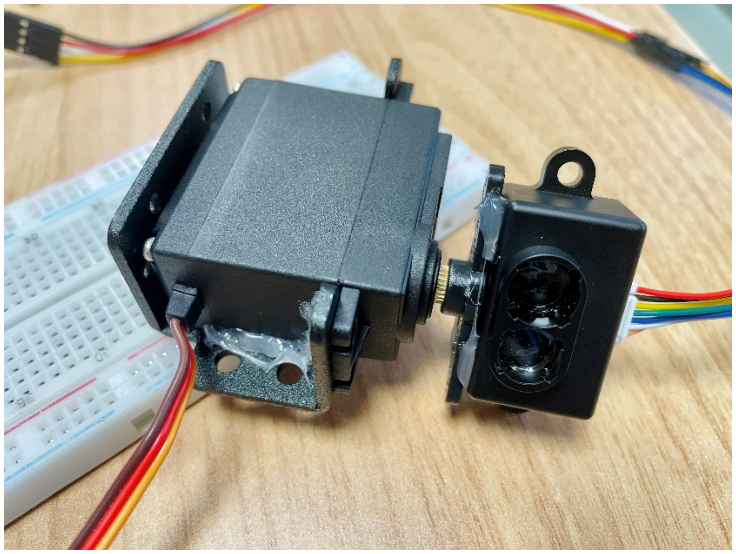

Figure 6. Combination of the Servo and LiDAR

We bind the LiDAR and the servo together and make it a 2D scanner. The LiDAR can one line and with send the data back to the processor with an interrupt. When the interrupt is generated, the program will check the corresponding PWM signal value and find out the angle to determine the hand position. Specifically, for each measurement, the servo rotates for 0 degree to 120 degrees with around 10-degree steps.

Figure 7. Slide Rheostat

For accuracy of the hand gesture detection, we use a slide rheostat with ADC to control the delay time for user to determine his/her answer (hand position in our case). By rotating the slide rheostat, user can customize the time needed for to decide the hand position.

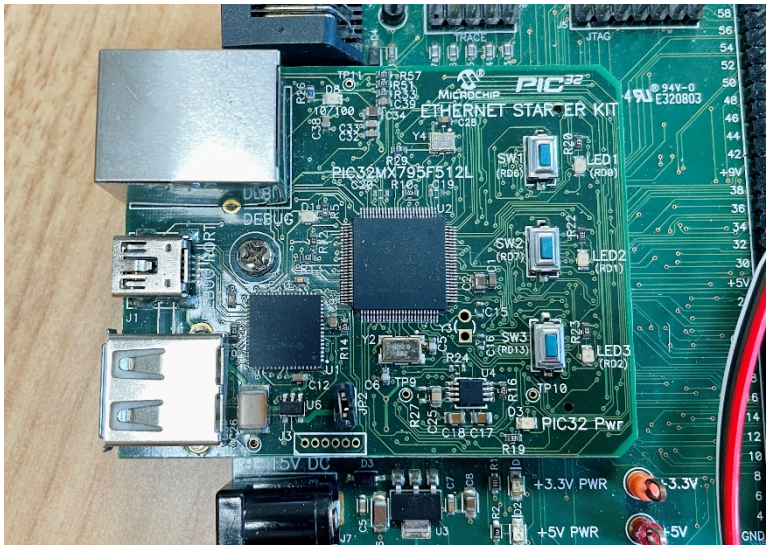

Figure 8. PIC32 Microprocessor

The core processor we used for this project is the PIC32MX795F512L. It is an MIPS architecture, 32-bit microprocessor offered by the course. All the controlling and logical calculation are done with this unit.

Results and Conclusion

The resulting system is shown in the following figures.

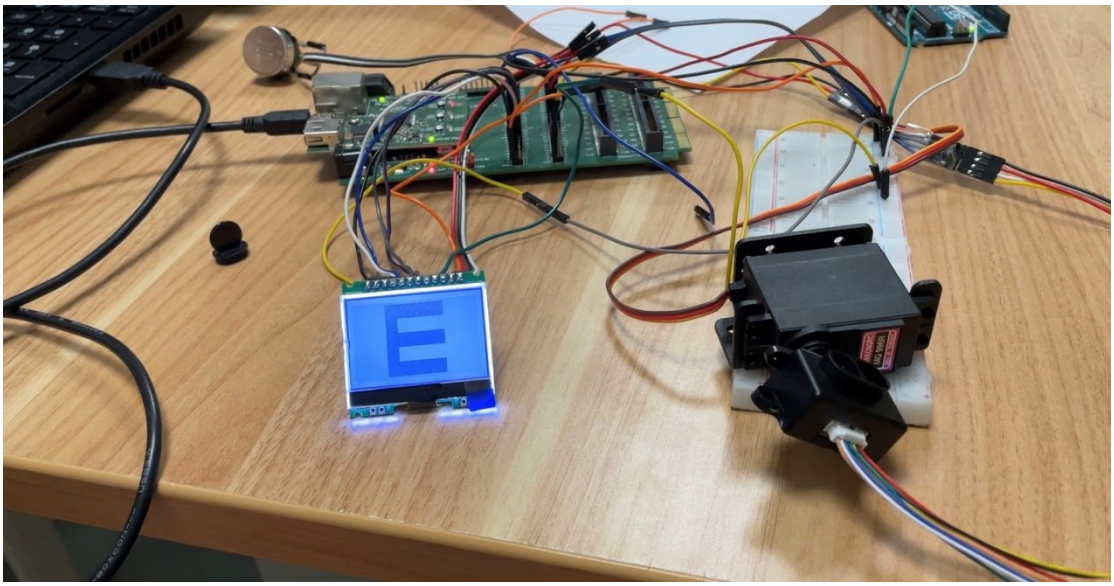

Figure 9. Eyesight Test Device

Finally, we successfully completed our automated vision test project. I don't want to say we'll do anything different next time, because every minute spent on this project was a great learning experience and we're always making progress, no matter how small it may seem. We spent a week debugging

various functions for PWM and LCD screens, and then an even greater effort to configure the radar for UART communications. As mentioned earlier, many extensions were possible in this project, but if we dealt with the different aspects separately, we would each learn less, and debugging each problem would be nearly impossible. Besides, as we are a group of two, we actually spent more effort to make the device works like our expectation.

Reference

[1] PIC32MX Datasheet

[2] PIC32 IO Expansion Schematics

[3] PIC32MX Family Reference Manual Section 14. Timers

[4] PIC32MX Family Reference Manual Section 16. Output Compare

[5] PIC32MX Family Reference Manual Section 14. 10-bit Analog-to-Digital Converter (ADC)

[6] PIC32MX Family Reference Manual Section 21. UART

[7] JlX12864G-086-PN LCD Display Instruction Manual

[8] MVR2EA Ranging Module Instruction Manual

[9] MVR2EA Datasheet

[10] VE373 Microprocessor Based System Design Lab 6. Voltmeter – ADC Application

Appendix

Bill of Materials

Name	Quantity	Price
LCD Display (JLX12964G-086-P))	1	20.3 RMB
TOF LiDAR (MVR2EA)	1	198 RMB
Servo Motor (MG996R)	1	13.4 RMB
PIC32 Microprocessor	1	\

Program

```c
/*
 * File:   adc.h
 * Author: 12591
 *
 * Created on August 1, 2021, 8:49 PM
 */

#ifndef ADC_H
#define ADC_H

void ADC_init();
int read_adc_value();

#endif  /* ADC_H */
#include <plib.h>
#include <stdio.h>
#include "util.h"

static unsigned int raw_value = 0;
// static float        cvt_value = 0.0;

void ADC_init()
{
    // ADC module
    IPC6bits.AD1IP = 3;
    IPC6bits.AD1IS = 1;
    IFS1bits.AD1IF = 0;
    IEC1bits.AD1IE = 1;
    AD1CON1 = 0;
    AD1CON1bits.FORM = 4; // 32-bit integer
    AD1CON1bits.SSRC = 7; // auto convert
    AD1CON1bits.ASAM = 1; // auto start
    AD1CON2 = 0;
    AD1CON2bits.VCFG = 0; // AV_DD/AV_SS
    AD1CON2bits.SMPI = 0; // interrupt for each sample
    AD1CON2bits.BUFM = 0; // one 16-word buffer
    AD1CON2bits.ALTS = 0; // always use sample A input mux
    AD1CON3 = 0;
    AD1CON3bits.ADRC = 0; // clock from PBCLK

    AD1CON3bits.SAMC = 4; // 4 T_AD
    // Sample Time = Acquisition Time (SAMC) + Conversion Time (12T_AD) = 1/500Hz = 2ms
    // 4 T_AD + 12 T_AD = 2ms => T_AD = 1/8ms
    // T_PB = 2.5*e-4 ms (1/4MHz)
```

```
    // TAD = 2*TPB*(ADCS + 1) (Reference manual Sec17)
    // ADCS = (T_AD/(2*T_PB))-1 = 249
    AD1CON3bits.ADCS = 249;
    AD1PCFG = 0;
    AD1PCFGbits.PCFG = 0xFFFE;
    AD1CHS = 0;
    AD1CHSbits.CH0SA = 0; // AN0 --> Channel 0 pos input

    AD1CON1bits.ON = 1;
}

int read_adc_value() {
    return raw_value;
}

#pragma interrupt ADC_ISR ipl3 vector 27
void ADC_ISR(void){
    raw_value = ADC1BUF0; // 0 - 1024
    // cvt_value = (double)raw_value / 1024 * 3.3;
    // LATDbits.LATD0 = 1;
    IFS1bits.AD1IF = 0;
}

/*
 * File:   lcd.h
 * Author: 12591
 *
 * Created on July 31, 2021, 6:39 PM
 */

#ifndef LCD_H
#define LCD_H

#include <xc.h>

#define sclk PORTEbits.RE0
#define sid PORTEbits.RE1 // sda
#define rs PORTEbits.RE2
#define reset PORTEbits.RE3
#define cs1 PORTEbits.RE4


#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long
```

```
int read_lcd_dir();

//void initial_lcd();
//void disp_rand(char size);
//void clear_screen();
//void display();


#endif  /* LCD_H */


#include "lcd.h"

// 64*60
char e_12[4][480] = {
    {
        // r
        0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc
, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc
, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
```

```
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc
, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
        0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
    },
    {
        // d
        0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc
, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
    },
    {
        // l
```

```
        0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc
, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
        0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
    },
    {
        // u
        0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0x
fc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc
, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
```

```
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc,
0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x
ff, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f,
0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x
3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f, 0x3f
    }
};
// 48*40
char e_8[4][240] = {
    {
        0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0x
f0, 0xf0, 0xf0, 0xf0, 0xf0,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f,
0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x
0f, 0x0f, 0x0f, 0x0f, 0x0f,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0x
f0, 0xf0, 0xf0, 0xf0, 0xf0,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f,
0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x
0f, 0x0f, 0x0f, 0x0f, 0x0f,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0x
f0, 0xf0, 0xf0, 0xf0, 0xf0,
```

```
        0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f
    },
    {
        0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f
    },
    {
        0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
        0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f
    },
```

```
    {
        0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xf0, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f, 0x0f
    }
};
// 32 * 24
char e_4[4][80] = {
    {
        0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0,
        0xff, 0xff, 0xff, 0xff, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3,
        0xff, 0xff, 0xff, 0xff, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3,
        0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03
    },
    {
        0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0,
        0xff, 0xff, 0xff, 0xff, 0x03, 0x03, 0x03, 0x03, 0xff, 0xff, 0xff, 0xff, 0x03, 0x03, 0x03, 0x03, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
        0x03, 0x03, 0x03, 0x03, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x03, 0x03, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x03, 0x03
    },
    {
```

```
        0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0,
0xc0, 0xc0, 0xc0,
        0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xff,
0xff, 0xff, 0xff,
        0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xc3, 0xff,
0xff, 0xff, 0xff,
        0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03
    },
    {
        0xc0, 0xc0, 0xc0, 0xc0, 0x00, 0x00, 0x00, 0x00, 0xc0, 0xc0, 0xc0, 0xc0, 0x00, 0x00, 0x00, 0x00, 0xc0,
0xc0, 0xc0, 0xc0,
        0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0xff,
0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xc0, 0xc0, 0xc0, 0xc0, 0xff, 0xff, 0xff, 0xff, 0xc0, 0xc0, 0xc0, 0xc0, 0xff,
0xff, 0xff, 0xff,
        0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03
    }
};

void delay(int i) {
    int j, k;
    for (j = 0; j < i; j++)
        for (k = 0; k < 110; k++);
}

void transfer_command(int data1) {
    char i;
    cs1 = 0;
    rs = 0;
    for (i = 0; i < 8; i++) {
        sclk = 0;
        if (data1 & 0x80) sid = 1;
        else sid = 0;
        sclk = 1;
        data1 <<= 1;
    }
    cs1 = 1;
}

void transfer_data(int data1) {
    char i;
    cs1 = 0;
```

```
    rs = 1;
    for (i = 0; i < 8; i++) {
        sclk = 0;
        if (data1 & 0x80) sid = 1;
        else sid = 0;
        sclk = 1;
        data1 <<= 1;
    }
    cs1 = 1;
}

void initial_lcd() {
    cs1 = 0;
    reset = 0;
    delay(500);
    reset = 1;
    delay(200);
    transfer_command(0xe2);
    delay(50);
    transfer_command(0x2c);
    delay(50);
    transfer_command(0x2e);
    delay(50);
    transfer_command(0x2f);
    delay(50);
    transfer_command(0x23);
    transfer_command(0x81);
    transfer_command(0x28);
    transfer_command(0xa2);
    transfer_command(0xc8);
    transfer_command(0xa0);
    transfer_command(0x40);
    transfer_command(0xaf);
}

void lcd_address(uchar page, uchar column) {
    transfer_command(0xb0 + page);
    transfer_command(((column >> 4)&0x0f) + 0x10);
    transfer_command(column & 0x0f);
}

void clear_screen() {
    uchar i, j;
    for (i = 0; i < 9; i++) {
```

```
        lcd_address(i, 0);
        for (j = 0; j < 132; j++) {
            transfer_data(0x00);
        }
    }
}

void clear_screen_64() {
    uchar i, j;
    for (i = 0; i < 9; i++) {
        lcd_address(i, 33);
        for (j = 0; j < 64; j++) {
            transfer_data(0x00);
        }
    }
}

void disp(uchar page, uchar column, uchar *dp, int width, int height) {
    uchar i, j, k, *data_temp, data1;
    data_temp = dp;
    for (j = 0; j < width/8; j++) {
        lcd_address(page + j, column);
        for (i = 0; i < height; i++) {
            transfer_data(*data_temp);
            data_temp++;
        }
    }
}

static unsigned int next = 1;
int dir = 0;

int read_lcd_dir() {
    return dir;
}

// range: 0 1 2 3
int myrand(void) {
    next = next * 1103515245 + 12345;
    dir = (unsigned int)(next/65536) % 32768 / 8192;
    return dir;
}

void mysrand(unsigned int seed) {
```

```
    next = seed;
}

void disp_rand(char size) {
    clear_screen_64();
    int dir = myrand();
    if (size == 0x00) {
        disp(0, 33, e_12[dir], 64, 60);
    }
    else if (size == 0x01) {
        disp(1, 43, e_8[dir], 48, 40);
    }
    else {
        disp(2, 53, e_4[dir], 32, 20);
    }
}

void disp_fix(char size, int dir) {
    clear_screen_64();
    if (size == 0x00) {
        disp(0, 33, e_12[dir], 64, 60);
    }
    else if (size == 0x01) {
        disp(1, 43, e_8[dir], 48, 40);
    }
    else {
        disp(2, 53, e_4[dir], 32, 20);
    }
}

/*
void display() {
    initial_lcd();
    clear_screen(); //clear all dots
    int i;
    for (i = 0; i < 4; ++i) {
        clear_screen(); //clear all dots
        disp(0, 33, e_12[i], 64, 60);
        delay(2000);
    }
    for (i = 0; i < 4; ++i) {
        clear_screen(); //clear all dots
        disp(1, 43, e_8[i], 48, 40);
        delay(2000);
```

```
    }
    for (i = 0; i < 4; ++i) {
        clear_screen(); //clear all dots
        disp(2, 53, e_4[i], 32, 20);
        delay(2000);
    }
    delay(5000);
}
 */

/*
 * File:   lidar.h
 * Author: 12591
 *
 * Created on July 22, 2021, 4:51 PM
 */

#ifndef LIDAR_H
#define LIDAR_H

typedef union cmd_u {
    int command;
    char bytes[4];
} cmd_t;

void LIDAR_init();
void start_lidar();
void clear_result();
int* read_result();

#endif  /* LIDAR_H */


// TODO: receive one byte -> check if it is head
// distance range: 0.2m~14m; 200=0xc8; 14000=0x36b0;
// Normally, no two bytes will be 0x8981 except the head

#include <plib.h>
#include "util.h"
#include "lidar.h"
#include "servo.h"

const cmd_t li_start = {0x56500006}; // byte[0]: 06; response:895000D9
const cmd_t li_stop = {0x56060050}; // response:8906008F
// const cmd_t li_hardware = {0x560E0058};
```

```
// const cmd_t li_outmode = {0x56700026}; // hex:56700026, ascii:56710027
// const cmd_t li_measure_freq = {0x56330065}; // 50hz
// const cmd_t li_baud_rate = {0x56120044}; // 115200; 9600:56110047

const char data_byte_0 = 0x89;
const char data_byte_1 = 0x81;
const char data_byte_9 = 0x00; // status

int data_byte_num = 0; // the number of the next byte of data frame
int start_response = 0; // check whether response of start command is received
char data_frame[11]; // store received bytes
int distance = 0;

void LIDAR_init() {
    asm("di");
    TRISF = 0;

    // UART1
    IPC6SET = 0x40000; //Set priority level = 1;
    IFS1CLR = 0x0001;
    IEC1SET = 0x0001;

    U1AMODE = 0x0;
    U1AMODEbits.PDSEL = 0x00; // 8-bit data & No parity
    U1AMODEbits.STSEL = 0x0; // 1 stop bit
    U1AMODEbits.BRGH = 1;
    U1ABRG = 16; // Configure baud rate to be 115200
    // 8MHz/(4*115200)-1=16.36 -> 16 ; error=2.1%
    U1AMODESET = 0x8000; //Enable UART1A

    U1ASTA = 0;
    U1ASTAbits.UTXEN = 1; //Transmit is enabled
    U1ASTAbits.URXEN = 1; //Receive is enabled
    IFS0bits.U1RXIF = 0;
    IEC0bits.U1RXIE = 1; //Enable receive interrupt
    IPC6bits.U1IP = 7;

    U1ASTAbits.URXISEL = 0;

    U1MODEbits.ON = 1;  // Enable UART1

    asm("ei");
}
```

```
void _send_cmd(cmd_t cmd) {
    // TODO
    U1ATXREG = cmd.bytes[3];
    U1ATXREG = cmd.bytes[2];
    U1ATXREG = cmd.bytes[1];
    U1ATXREG = cmd.bytes[0];
    /*
    for (int i = 3; i >= 0; --i) {
        U1ATXREG = cmd.bytes[i];
    }
     */
}

void start_lidar() {
    _send_cmd(li_start);
}

void stop_lidar() {
    _send_cmd(li_stop);
}

// TODO: check response of start

int temp = 0;

void _update_byte_num(char byte_i) {
    if ((data_byte_num == 0 && byte_i != data_byte_0) ||
            (data_byte_num == 1 && byte_i != data_byte_1) ||
            (data_byte_num == 10)) {
        // continues waiting for the first byte;
        // or, fails to detect the head; back to waiting for the first byte
        // or, receives the last byte
        data_byte_num = 0;
    } else {
        data_byte_num++;
    }
}

int check_verity() {
    char result = data_frame[0];
    int i = 1;
    while (i < 10) {
        result = result ^ data_frame[i];
        ++i;
```

```
    }
    return (result == data_frame[10]);
}

int dir_detect_num[4] = {0}; // r, d, l, u

void clear_result() {
    int i;
    for (i = 0; i < 4; ++i) {
        dir_detect_num[i] = 0;
    }
}

int* read_result() {
    return dir_detect_num;
}

// Receiving ISR
#pragma interrupt RX_ISR ipl7 vector 24
void RX_ISR(void) {
    data_frame[data_byte_num] = U1RXREG;
    if (data_byte_num == 2) {
        distance = (int)(data_frame[2]) & 0x000000ff;
    }
    else if (data_byte_num == 3) {
        distance = distance | ((int)data_frame[3] << 8 & 0x0000ff00);
    }
    else if (data_byte_num == 10) {
        // receive the last byte, i.e. the whole data frame
        if (data_frame[9] == data_byte_9) {
            // 900 1300 1700 2100
            if (distance > 0 && distance < 800) {
                if (read_pwm() < 1200) {
                    dir_detect_num[0]++;
                }
                else if (read_pwm() > 1800) {
                    dir_detect_num[2]++;
                }
                else {
                    if (distance < 400) {
                        dir_detect_num[1]++;
                    }
                    else {
                        dir_detect_num[3]++;
```

```
                    }
                }
            }
            /*
            if (distance > 0 && distance < 300) {
                LATDSET = OUTPUT_1;
                LATDCLR = OUTPUT_2;
            }
            else if (distance >= 300 && distance < 600) {
                LATDSET = OUTPUT_2;
                LATDCLR = OUTPUT_1;
            }
            else {
                LATDCLR = OUTPUT_1;
                LATDCLR = OUTPUT_2;
            }
             */
        }
    }
    _update_byte_num(data_frame[data_byte_num]);

    IFS0bits.U1RXIF = 0;
}


/*
 * File:   servo.h
 * Author: 12591
 *
 * Created on July 25, 2021, 3:19 PM
 */

#ifndef SERVO_H
#define SERVO_H

void servo_init();
void change_pwm(int new_pwm);
int read_pwm();
int is_unfinished();

#endif  /* SERVO_H */


#include <plib.h>
```

```
int is_change = 0;
int is_add = 1;
int acc = 100;
int pwm = 900; // 900 - 2100

void servo_init() {
    OC1CON = 0x0000; // Turn off the OC1 when performing the setup
    OC1R = 0; // Initialize primary Compare register
    OC1RS = pwm; // Initialize secondary Compare register
    OC1CON = 0x0006; // Configure for PWM mode without Fault pin enabled
    T2CONSET = 0x0008; // Enable 32-bit Timer mode
    T2CONbits.TCKPS = 3;
    PR2 = 19999; // Set period
    // Configure Timer3 interrupt. Note that in PWM mode, the corresponding source timer
    // interrupt flag is asserted. OC interrupt is not generated in PWM mode.
    IFS0CLR = 0x00001000; // Clear the T3 interrupt flag
    IEC0SET = 0x00001000; // Enable T3 interrupt
    IPC3SET = 0x0000001C; // Set T3 interrupt priority to 7
    T2CONSET = 0x8000; // Enable Timer2
    OC1CONSET = 0x8020; // Enable OC1 in 32-bit mode.
}

void change_pwm(int new_pwm) {
    pwm = new_pwm;
}

int read_pwm() {
    return pwm;
}

int COUNT_MAX=10;
int count = 0;
int COUNT_STOP_MAX = 24; // 2400/acc
int count_stop = 24; // initialized to COUNT_STOP_MAX

void change_count_stop(int new_count_stop) {
    // change count_stop to 0 to start a new cycle
    count_stop = new_count_stop;
}

void change_count_max(int new_count_max) {
    // use adc to control
    COUNT_MAX = new_count_max;
}
```

```
int is_unfinished() {
    if (count_stop < COUNT_STOP_MAX - 1) {
        return 1;
    }
    else {
        return 0;
    }
}

void __ISR(_TIMER_3_VECTOR, ipl7) T3_IntHandler(void) {
    if (count == COUNT_MAX) {
        if (count_stop < COUNT_STOP_MAX) {
            if (is_add) {
                pwm += acc;
                if (pwm >= 2100) {
                    is_add = 0;
                }
            }
            else {
                pwm -= acc;
                if (pwm <= 900) {
                    is_add = 1;
                }
            }
            count_stop++;
        }
        count = 0;
    }
    count++;
    OC1RS = pwm;
    IFS0CLR = 0x1000; // Clearing Timer3 interrupt flag
}
/*
 * File:   util.h
 * Author: 12591
 *
 * Created on July 24, 2021, 10:37 PM
 */

#ifndef UTIL_H
#define UTIL_H

#define INPUT_0 (1 << 6)
```

```
#define INPUT_1 (1 << 7)
#define INPUT_2 (1 << 13)
#define OUTPUT_0 (1 << 0)
#define OUTPUT_1 (1 << 1)
#define OUTPUT_2 (1 << 2)

#endif  /* UTIL_H */




void delay_sw(unsigned int dl) {
    // 40000  0.5s; 1600  20ms
    int i = dl;
    while (i--) {}
}

/*
 * File:   newmain.c
 * Author: 12591
 *
 * Created on July 22, 2021, 4:47 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <plib.h>
#include "util.h"
#include "servo.h"
//#include "lidar.h"

/*
 *
 */

void TIMER_init() {
    // PBCLK = 8MHz
    SYSKEY = 0x0; // write invalid key to force lock
    SYSKEY = 0xAA996655; // write Key1 to SYSKEY
    SYSKEY = 0x556699AA; // write Key2 to SYSKEY
    OSCCONbits.FRCDIV = 0b000; // SYSCLK = FRC = 8MHz
    OSCCONbits.PBDIV = 0b00; // configure PBDIV so PBCLK = SYSCLK
    OSCCONbits.COSC = 0b111; // select internal FRC divided by OSCCON<FRCDIV>
    SYSKEY = 0x0; // write invalid key to force lock
}
```

```
void MCU_init() {
    INTCONbits.MVEC = 1; // TODO
    INTEnableSystemMultiVectoredInt(); // Enable system wide interrupt to multivectored mode.

    TRISD = INPUT_2;
    LATDCLR = OUTPUT_0 | OUTPUT_1 | OUTPUT_2;

    TRISE = 0;
}

//int pwm = 1000; // TODO

int check_result(int result[4], int dir) {
    int i;

    int has_nonzero = 0;
    for (i = 0; i < 4; ++i) {
        if (result[i] > 0) {
            has_nonzero = 1;
            break;
        }
    }
    if (!has_nonzero) {
        return 0;
    }

    int max = 0;
    int result_max = result[0];
    for (i = 1; i < 4; ++i) {
        if (result[i] > result_max) {
            max = i;
            result_max = result[i];
        }
    }
    return (dir == max);
}

int main(int argc, char** argv) {
    //OSCSetPBDIV(OSC_PB_DIV_1);
    MCU_init();
    TIMER_init();

    LIDAR_init();
```

```
servo_init();

initial_lcd();
clear_screen();

ADC_init();

/*
start_lidar();
while(1){}
 */

/*

while (1) {
    disp_rand(0x00);
    delay_sw(100000);
    disp_rand(0x01);
    delay_sw(100000);
    disp_rand(0x02);
    delay_sw(100000);
}
 */

int sw3_pushed = 0;
char disp_level = 0x00;
int disp_delay = 100000;

while (1) {
    if ((PORTD & INPUT_2) == 0) {
        delay_sw(1600); // delay for de-bounce
        if ((PORTD & INPUT_2) == 0)
            sw3_pushed = 1;
    } else {
        delay_sw(1600); // delay for de-bounce
        if ((PORTD & INPUT_2) != 0) {
            if (sw3_pushed == 1) {
                sw3_pushed = 0;

                disp_delay = 100000 + read_adc_value() * 400;
                disp_level = 0x00;
                LATDCLR = OUTPUT_1;

                // main logic
```

```
                    int matched = 1;
                    while (matched && disp_level < 0x03) {
                        disp_rand(disp_level);
                        delay_sw(disp_delay);

                        change_pwm(900);
                        clear_result();
                        change_count_stop(0);

                        while (is_unfinished()) {}
                        if (check_result(read_result(), read_lcd_dir())) {
                            disp_level++;
                        }
                        else {
                            matched = 0;
                        }
                    }

                    if (disp_level >= 0x03) {
                        LATDSET = OUTPUT_1;
                    }

                }
            }
        }
    }

    return (EXIT_SUCCESS);
}
```