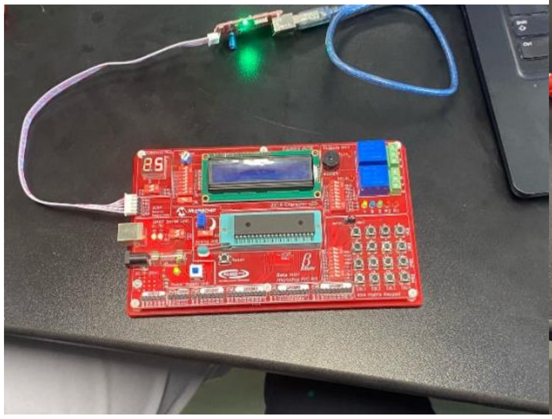

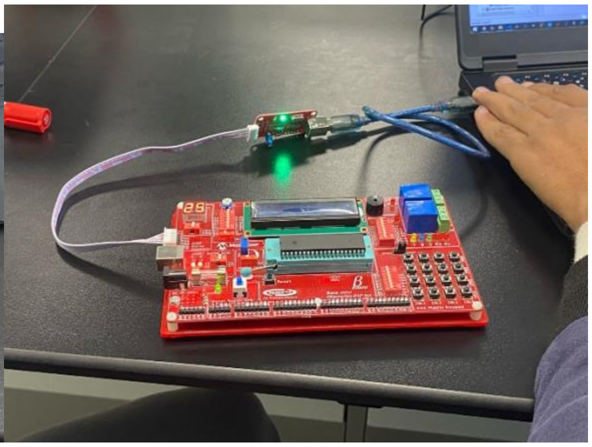

Code

```
#pragma config FOSC = HS       // Oscillator Selection bits (HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)

#pragma config WDTE = OFF      // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON      // Power-up Timer Enable bit (PWRT enabled)

#pragma config MCLRE = ON      // RE3/MCLR pin function select bit (RE3/MCLR pin function is MCLR)

#pragma config CP = OFF        // Code Protection bit (Program memory code protection is disabled)

#pragma config CPD = OFF       // Data Code Protection bit (Data memory code protection is disabled)

#pragma config BOREN = ON      // Brown-out Reset Selection bits (BOR enabled)

#pragma config IESO = ON       // Internal External Switchover bit (Internal/External Switchover mode is enabled)

#pragma config FCMEN = ON      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor is enabled)

#pragma config LVP = OFF       // Low-Voltage Programming Enable bit (RB3/PGM pin has digital I/O, HV on MCLR must be used for programming)


#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)

#pragma config WRT = OFF       // Flash Program Memory Self Write Enable bits (Write protection off)



#define _XTAL_FREQ  4000000

#include <xc.h>
```

```
#define motor_pin_1 0

#define motor_pin_2 1

#define motor_pin_enable 2


#define SET_BIT(reg , bit) (reg |= (1<<bit))

#define CLR_BIT(reg , bit) (reg &= ~(1<<bit))


//

//int main()

//{

//    TRISE = 0 ;

//    PORTE = 0xff;

//

//    TRISC = 0 ;

//    PORTC = 0xff;

//

//    TRISD = 0 ;

//    PORTD = 0b00110000;

//

//    ANSEL = 0 ;

//    ANSELH = 0;

//

//    SET_BIT(PORTE, motor_pin_enable);

//

////    PORTB = 0x01;

////    direction portd

////    portc c

//
```

```
//
//   while(1)
//   {
////       // set direction to right
////       SET_BIT(PORTE, motor_pin_1);
////       CLR_BIT(PORTE, motor_pin_2);
////       __delay_ms(1000);
////       SET_BIT(PORTE, motor_pin_2);
////       CLR_BIT(PORTE, motor_pin_1);
////
////       __delay_ms(1000);
//   }
//
//
//
//   return 0 ;
//}

const uint8_t segment_map[10] = {
  0b00111111, // 0
  0b00100001, // 1
  0b01110110, // 2
  0b01110011, // 3
  0b01101001, // 4
  0b01011011, // 5
  0b01011111, // 6
  0b00111001, // 7
  0b01111111, // 8
  0b01111011  // 9
```

```
};



void display_number(uint8_t num) {
    uint8_t tens = num / 10;  // Extract tens place
    uint8_t ones = num % 10;  // Extract ones place
    for(int i = 0 ; i<5 ;i++)
    {



    PORTC = 0b00100000; // Select left digit (C5)
    PORTD = segment_map[ones];
//    for(unsigned int i=0; i<65000; i++);
    __delay_ms(10);

    PORTC = 0b00010000; // Select right digit (C4)
    PORTD = segment_map[tens];
//    for(unsigned int i=0; i<65000; i++);
    __delay_ms(10);
    }
}

void main(void) {
    ANSEL = 0;
    ANSELH = 0;

    TRISD = 0b00000000;
    TRISC = 0b00000000;
    TRISB = 0b00000011;


    INTCON |= 1<<7;


    INTCON |= 1<<3;


    IOCB0 = 1;
    IOCB1 = 1;


//    while(1)
//    {
//        PORTD = 0b00100001;
//        for(unsigned int i=0; i<65000; i++);
//        PORTD = 0b01110110;
//        for(unsigned int i=0; i<65000; i++);
//        PORTD = 0b01110011;
//        for(unsigned int i=0; i<65000; i++);
//    }


    while (1) {
//        display_number(25);
        for (uint8_t i = 0; i <= 99; i++) {
            display_number(i);
            __delay_ms(100);
        }
    }



    return;
```

```
}
```