SCHOOL OF MECHATRONIC SYSTEMS ENGINEERING
SIMON FRASER UNIVERSITY

MSE 352: Digital Logic and Microcontrollers

Project Report

December 14th, 2020

Mohammed Safar	301319896
Chris Vattathichirayil	301328090

Introduction

This project uses peripherals of the 8051 MCU to simulate traffic lights at an intersection. Of the features available in the MCU, the aspects used were the I/O pins, timer functions, delays, and external interrupts. Along with the pattern in which the lights turn on and off, the intersection displays a count to show how much time is left until a pedestrian activated crosswalk turns off. Additionally, a signal for cars turning left onto the road running West/East is programmed. To simulate these two components, both external interrupt pins (P3.2 and P3.3) are used on the 8051. Only cars heading West off of a left turn are free to turn while cars heading South on the vertical road must wait because of the protected left signal. Although left turn lights are typically green when active, we will use a blue LED for better contrast. The crosswalk is also limited to work on only the horizontal road for simplicity. Regarding these interrupts, the use of buttons will be incorporated into the design.

Design Process

Initially, to understand how pins and delays can be constructed with knowledge of machine cycles, a preliminary design was made in Proteus and coded in assembly with the crosswalk and left turn functions absent.

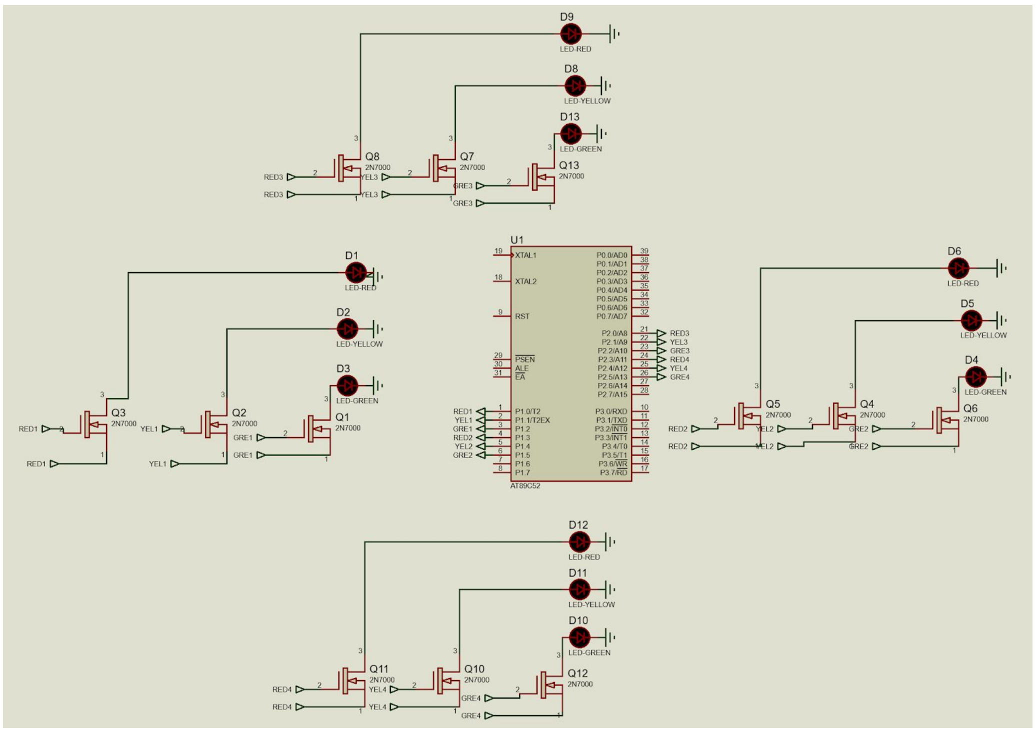

Fig. 1 - Proteus schematic of initial design.

In the figure above is the initial circuit design made in Proteus along with the assembly source code in the appendix. This initial circuit was meant to aid us in understanding the application of our I/O pins and how they can be set and cleared by our code. Notably there are no pull-down or pull-up resistors on any connections since ports 1-3 already have built-in resistors. To achieve our target design, we incorporated the two buttons, 7-segment, and the blue LED into the circuit and moved to programming in C to better understand operational flow of our code. This allowed us to reach our goal of creating the intersection to our specifications.

The HEX file used in the Proteus microcontroller is included in the supplementary deliverable. For pauses between lights changing colors, we utilize simple delays. For the countdown of the crosswalk, we use a timer while the interrupt feature is used for the activation of the left turn and the crosswalk. Although left turns are activated typically by motion sensors or pre-programmed routines at intersections, we will use a button to conveniently simulate the interrupts used in our design. The final circuit design is shown in figure 2.

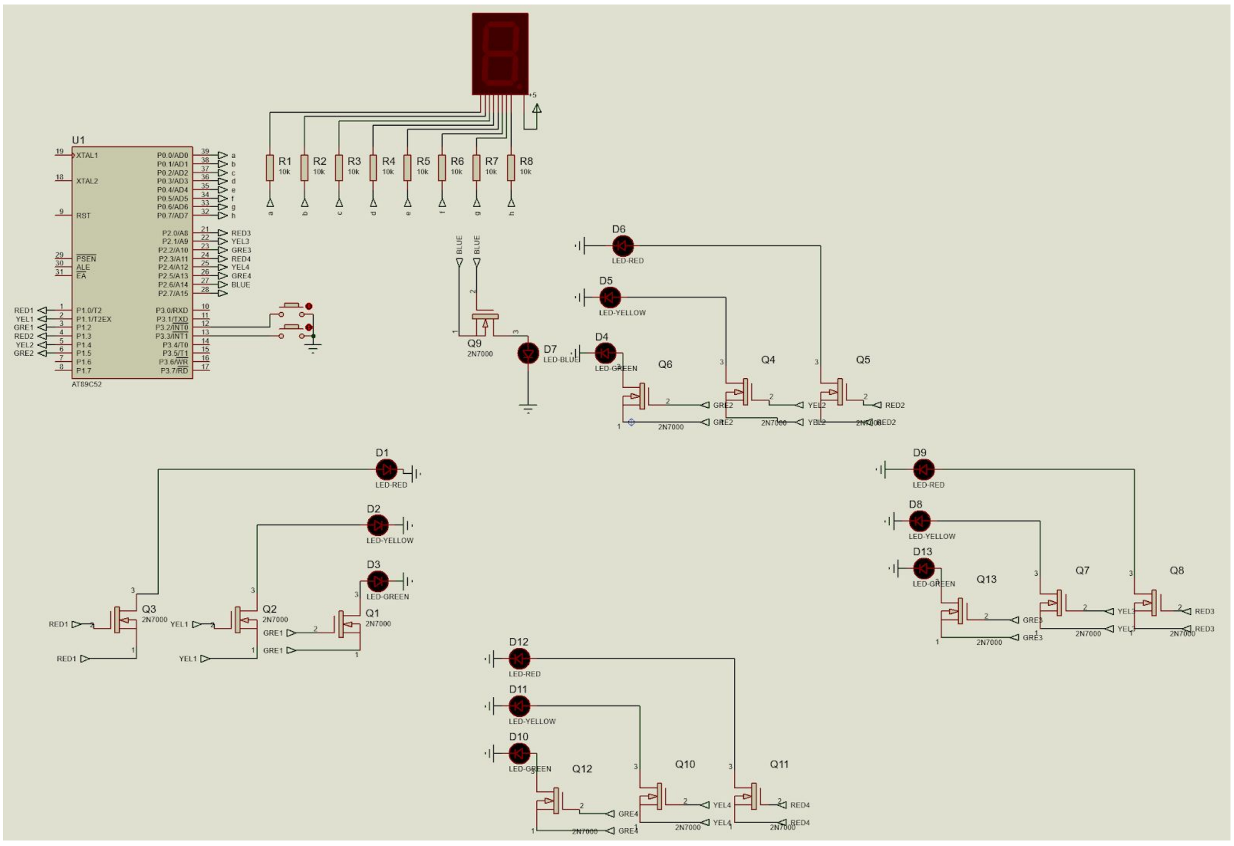

Fig. 2 - finalized circuit design with all required components.

The blue LED for the left turn signal is implemented along with a 7-segment to show the numbers 6 to 0. Crosswalks are usually anywhere from 15 to 30 seconds in practical intersections, however, the times between actions of the system take much less time to allow the simulation to move much quicker. This allowed us to find issues or room for improvements in a more convenient manner while also saving time during testing of the code. The 7-segment has resistors connected since port 0 does not have the built in, unlike the other ports. Ports 1 and 2 are used for the lights of the intersection while INT0 and INT1 are plugged into P3.2 and P3.3 as required by the architecture of the microcontroller. The C program takes these two interrupts as interrupt 0 and interrupt 2. All LED lights are connected and driven by MOSFETs connected to the I/O pins.

Utilized Peripherals

GPIOs:

The general purpose input output pins on the 8051 microcontroller are connected in the configuration shown in figure 2. A total of 13 output pins are used to power 13 LED's with three at each of the four intersections and one for the protected left turn. West and North lights are controlled using pins P1.0 to P1.5. East, South and the left turn are controlled using pins P2.0 to P2.6. A 7 segment display is the output of pins P0.0 to P0.7 used to display the countdown for pedestrians when crossing the road. Two pins of port 3 used for external interrupts are taken as inputs to the system. The buttons are programmed in a way to never cause the system to fail and only conduct their corresponding routines when appropriate. This is further explained in the interrupts section.

Delays:

Since the program runs on C code, the function used to create delays was MSDelay( ). Using a value of j = 1275 for the inner for loop of the function as shown in lecture slides, the delay would run for far too long if our input time was in milliseconds. As such, we use a value of 200 for j to allow the simulation to run quick enough for us to analyze and tweak accordingly. Delays occur after each transition between colors of the traffic lights. Timers are used for other pauses in time as explained in the next section.

Timers:

The timer function was used for the counting down of the crosswalk 7-segment. Timer 1 was used in mode 1 (16-bit mode). The high and low bytes of this timer were loaded with 4B and FD respectively. With this value, the timer would cycle through 46083 machine cycles (FFFF subtracted by 4BFD) which roughly equals 50 milliseconds. To show the number on display for an adequate amount of time, the timer function was called 8 times in a for loop after each number on the 7-segment for a time of 400 milliseconds. Starting the timer only once would make the crosswalk complete its action too quickly. Once the function is done counting down, the timer flag is raised, the timer stops, and the flag is manually cleared.

External interrupts:

The external interrupt pins INT0 and INT1 are routed to normally open switches that will ground the terminals when pressed. The two switches activate the interrupt service routine for left turn signal and the crosswalk respectively. These ISR's only change the value of two variables used in our programming that are used to check an if statement. Inside the if statements are the true actions of the left turn and crosswalk routines. This is so activating an interrupt does not halt the current state of the intersection and move to a crosswalk countdown for example. The two interrupts only occur when it is safe to do so in a practical setting. The pins for these button inputs are P3.2 and P3.3 which translate to interrupt 0 and interrupt 2 in C programs. For the left turn interrupt, cars are detected by sensors but for simplicity, a button is used to notify the system to activate the left turn ISR. The crosswalk however, uses a button just like actual crosswalks.

Contributions

Mohammed coded the project with assistance from Chris in error checking and troubleshooting. Helped to implement the code in the Proteus file and see how the code affected the operation and make proper adjustments. For example, changing delay values and implementing as many peripherals of the MC as requested by course instructors. Mostly software related contribution for Mohammed with influence in deciding where crucial components of the system are to be plugged in. For example, the connections and orientation of the pins to work in tandem with the code and offer good readability.

Chris' role was more hardware based. Decided which resistors should be used for pull-ups and how the MOSFETs should be connected to increase LED functionality (this decision was made before moving to the remote project setup). Also assisted in troubleshooting code, suggesting corrections to make the system work as intended, developed pseudo code algorithms and programmed the timers. Chris also helped with orienting the layout of the lights in Proteus to reflect the directions they would be in real life and decided to have the lights grouped in packs of six on ports 1 and 2. All in all, Chris was hardware based contributions with assistance in software with Mohammed having software based contributions with assistance in hardware.

Conclusion

This project effectively utilizes the functionality of timers, interrupts, delays, and GPIO pins. It would be possible to program traffic lights using this method as all the functionality of controlling regular street traffic lights were shown to work on the 8051 microcontroller. Additional functionality of four crosswalks and left turns for all roads would be included in a design using MCU with more GPIO pins. Two types of delays were used to demonstrate the difference between using the internal timer and using the CPU to wait for a certain number of clock cycles. The advantage of using external interrupts were demonstrated as the state of a variable used to trigger interrupt service routine could be activated independent of whether the MCU is running a program. We learned the importance of efficient programming to make maximum usage of the available hardware and their specifications.

Appendix A: Embedded C programming of 8051 Traffic Lights

```
#include <reg51.h>
// Western lights on Proteus
sbit RED1 = P1^0;
sbit YEL1 = P1^1;
sbit GRE1 = P1^2;

// Northern lights on Proteus
sbit RED2 = P1^3;
sbit YEL2 = P1^4;
sbit GRE2 = P1^5;

// Eastern lights on Proteus
sbit RED3 = P2^0;
sbit YEL3 = P2^1;
sbit GRE3 = P2^2;

// Southern lights on Proteus
sbit RED4 = P2^3;
sbit YEL4 = P2^4;
sbit GRE4 = P2^5;

// Left turn light
sbit BLUE = P2^6;

// Defining 7-segment pins
sbit a = P0^0; // 7-segment a
sbit b = P0^1; // 7-segment b
sbit c = P0^2; // 7-segment c
sbit d = P0^3; // 7-segment d
sbit e = P0^4; // 7-segment e
sbit f = P0^5; // 7-segment f
sbit g = P0^6; // 7-segment g
sbit h = P0^7; // 7-segment h

// 0 = lights are red, 1 = lights are yellow, 2 = lights are green
unsigned char WEST;
unsigned char NORTH;
unsigned char EAST;
unsigned char SOUTH;

// declaring the names of interrupts
unsigned char LeftTurn;
unsigned char Crosswalk;

// loop for creating a delay
void MSDelay(unsigned int);
void MSDelay(unsigned int itime){
                unsigned int i, j;
                for(i = 0; i < itime; i++)
                        for(j = 0; j < 120; j++);
}
// interruts for left turn and cross walk buttons
void Left(void) interrupt 0{
           LeftTurn = 1; // stores a value to be checked later
}
void Crossing(void) interrupt 2{
           Crosswalk = 1; // stores a value to be checked later
}
// timers for the 7-segment
void CrosswalkDelay(){
           // select mode 1 of timer 1 (16-bit timer mode)
                        TMOD = 0x10;
                        // 4BFD = 19453 in decimal
                        // 65536 - 19453 = 46083
                       // 46083 * 1.085 microseconds = 50 milliseconds
                        TL1 = 0xFD;
                        TH1 = 0x4B;

                // start timer 1
                        TR1 = 1
                // do machine cycles until overflow flag is raised
                        while (TF1 == 0);
                // stop the timer
                        TR1 = 0;
                //reset the flag
                        TF1 = 0;
        }
        ////////////////////////////////////////////////////////////////////////////////
        void main(void){
                    EA = 1; // enables all interrupts
                    EX0 = 1; // external interrupt 0 is enabled
                    EX1 = 1; // external interrupt 1 is enabled
                //ET0 = 1;
                //ET1 = 1;
                    BLUE = 0; // set left turn to be off initially
                    // Loop for running the lights forever
                    while(1){
                        // sets North and South to red
                        RED2 = 1; YEL2 = 0; GRE2 = 0;
                        RED4 = 1; YEL4 = 0; GRE4 = 0;
                        NORTH = 0; SOUTH = 0;
                        // sets West and East to green
                        RED1 = 0; YEL1 = 0; GRE1 = 1;
                        RED3 = 0; YEL3 = 0; GRE3 = 1;
                        EAST = 2; WEST = 2;
                        // Checks if crosswalk has been pressed and displays countdown on 7-segment.
                        // Only applied to the road heading West/East
                        if(Crosswalk == 1 && WEST == 2 && EAST == 2){
```

```
                // counts down from 6 to 0 on the 7-segment
                // works based on negative logic
            // 0.40 second delays between each number by calling 50 ms timer 7 times

            unsigned char x;
                a = 0; b = 1; c = 0; d = 0; e = 0; f = 0; g = 0; CrosswalkDelay();
            for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 0; b = 1; c = 0; d = 0; e = 1; f = 0; g = 0; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 1; b = 0; c = 0; d = 1;     e = 1; f = 0; g = 0; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 0; b = 0; c = 0; d = 0; e = 1; f = 1; g = 0; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 0; b = 0; c = 1; d = 0; e = 0; f = 1; g = 0; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 1; b = 0; c = 0; d = 1; e = 1; f = 1; g = 1; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 0; b = 0; c = 0; d = 0; e = 0; f = 0; g = 1; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}
                a = 1; b = 1; c = 1; d = 1; e = 1; f = 1; g = 1; CrosswalkDelay();
                for(x = 0; x < 7; x++){CrosswalkDelay();}

            // resets the crosswalk interruption
                Crosswalk = 0;
        }

        // delay 3.5 seconds
        MSDelay(3500);

        // sets West and East to yellow
        RED1 = 0; YEL1 = 1; GRE1 = 0;
        RED3 = 0; YEL3 = 1; GRE3 = 0;
        EAST = 1; WEST = 1;
        // delay for 0.75 seconds
                        MSDelay(750);

            // sets West and East to red
                        RED1 = 1; YEL1 = 0; GRE1 = 0;
                        RED3 = 1; YEL3 = 0; GRE3 = 0;
                        EAST = 0; WEST = 0;

        // delay for 0.75 seconds
                        MSDelay(750);

                        // sets North and South to green
        RED2 = 0; YEL2 = 0; GRE2 = 1;
                        RED4 = 0; YEL4 = 0; GRE4 = 1;
                        NORTH = 2; SOUTH = 2;

                        // checks to see if the left turn lane has cars waiting to turn
                        if(LeftTurn == 1 && WEST == 0 && EAST == 0 && SOUTH == 2 &&
      NORTH == 2){

                             // cars heading South must wait for left turning cars to finish turning
                             // cars heading North are free to go the whole time
                             RED2 = 0; YEL2 = 0; GRE2 = 1;
                             RED4 = 1; YEL4 = 0; GRE4 = 0;
                             NORTH = 2; SOUTH = 0;

                             //delay 0.25 seconds and alternating on/off of blue LED
                             BLUE = 1; MSDelay(250);
                             BLUE = 0; MSDelay(250);
                             BLUE = 1; MSDelay(250);
                             BLUE = 0; MSDelay(250);
                             BLUE = 1; MSDelay(250);
                             BLUE = 0; MSDelay(250);
                             BLUE = 1; MSDelay(250);
```

```
                    BLUE = 0; MSDelay(250);
                        BLUE = 1; MSDelay(250);
                        BLUE = 0; MSDelay(250);
                        // sets North and South to green
                        RED2 = 0; YEL2 = 0; GRE2 = 1;
                        RED4 = 0; YEL4 = 0; GRE4 = 1;
                        NORTH = 2; SOUTH = 2;
                        // resets the left turn interruption
                        LeftTurn = 0;
                    }
                    // delay 3.5 seconds
                    MSDelay(3500);
                    // sets North and South to yellow
                    RED2 = 0; YEL2 = 1; GRE2 = 0;
                    RED4 = 0; YEL4 = 1; GRE4 = 0;
                    NORTH = 1; SOUTH = 1;
                    // delay for 0.75 seconds
                    MSDelay(750);
                    // sets North and South to red
                    RED2 = 1; YEL2 = 0; GRE2 = 0;
                    RED4 = 1; YEL4 = 0; GRE4 = 0;
                    NORTH = 0; SOUTH = 0;
                    // delay for 0.75 seconds
                    MSDelay(750);
                }
            }
```

Appendix B: Assembly programming of 8051 Traffic Lights (preliminary design)

```
$NOMOD51
$INCLUDE (80C52.MCU)
;====================================================================
; RESET and INTERRUPT VECTORS
;====================================================================
    ; Reset Vector
    ORG   0000h
    LJMP   Start
;====================================================================
; CODE SEGMENT
;====================================================================
     ORG   0100h
start:
;write your code
        MOV P1, #00001001b ;RED
        MOV P2, #00100100b ;GREEN
        ACALL DELAY1
        MOV P1, #00001001b ;RED
        MOV P2, #00010010b ;YELLOW
        ACALL DELAY2
        MOV P1, #00001001b ;RED
        MOV P2, #00001001b ;RED
        ACALL DELAY2
        MOV P1, #00100100b ;GREEN
        MOV P2, #00001001b ;RED
        ACALL DELAY1
        MOV P1, #00010010b ;YELLOW
        MOV P2, #00001001b ;RED
        ACALL DELAY2
        MOV P1, #00001001b ;RED
        MOV P2, #00001001b ;RED
        ACALL DELAY2
        JMP start

DELAY1:   MOV R2, #255
AGAIN1:   MOV R3, #255
BACK1:    MOV R4, #3
HERE1:        NOP
              NOP
              NOP
              NOP
              DJNZ R4, HERE1
              DJNZ R3, BACK1
              DJNZ R2, AGAIN1
          RET

DELAY2:   MOV R2, #255
AGAIN2:   MOV R3, #200
HERE2:        NOP
              NOP
              NOP
              NOP
              DJNZ R3, HERE2
              DJNZ R2, AGAIN2
          RET
;============================================================
    END
```