Traffic Signal Control with Emergency Override

Luyahan (Leader)

Casalme (Rapporteur)

Santos (Member)

I. Background

Construct a traffic monitoring signal using a microprocessor and counters. The counter counts the number of vehicles that passed and would automatically change the color of the light to yellow then to green if it has reached the number of vehicles programmed. There will be a remote control provided to the emergency responders such as ambulances, fire trucks and the like, in order for them to pass by without any delay.

II. Project Description

This project is a monitoring signal that counts the programmed amount of cars to pass through before changing the light of the traffic light from green to yellow to red and vice-versa. The program will allow emergency respondents such as ambulances, fire trucks and the like for them to pass through without being counted and to pass by without any delay.

III. Project Specification

Objectives:

- Develop a prototype of traffic signal control with emergency override.
- Use microcontroller, which is more cost effective, instead of PLC (Programmable Logic Controller)
- Use Microchip Studio, to develop the code for the project.
- Utilize what the students learned in the course.

Expected Features:

The stoplight should have the following expected features:

- Emergency override that interrupts the normal flow of operation of the stoplight to light up green for emergency vehicles to pass through then return to the normal flow of operation
- Counts cars that have passed to interrupt the normal time of the green light and shortens it by up to a set number of cars that have passed through.

Constraints:

- The prototype will only consist of one set of LED lights that will represent a single unit of stoplight due to time constraint because of other courses having heavy final requirements.
- The Emergency Override, which is responsible for interrupting the normal operations of the stoplight to let the emergency vehicles to pass through, will be simulated by a momentary push button switch due to lack of funds and limitations on travel to go and buy a remote control module.
- The Inductive loop detector, which is responsible for monitoring the number of cars that passed through, will be simulated by a momentary push button due to lack of funds and limitations on travel to go and buy a remote control module.

Deadlines:

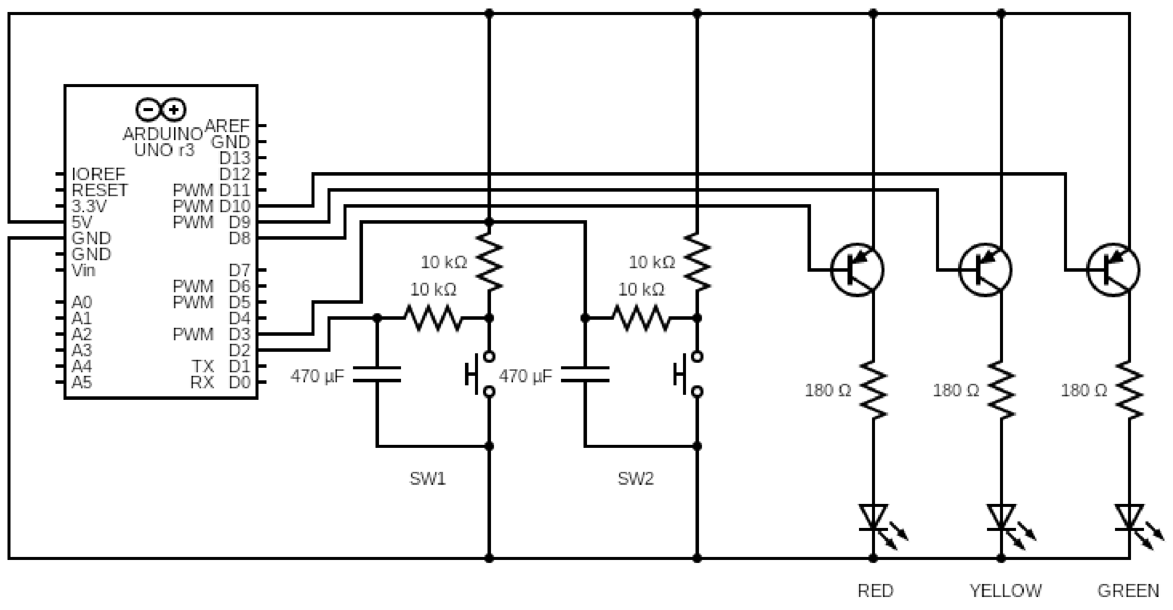
- Phase 1: Project Proposal (March 10)
- Phase 2: IPO (March 24)
- Phase 3: Project/Document Review (April 21)
- Phase 4: Presentation/Demonstration (May 12)

IV. Implementation Details

a. Circuit Description

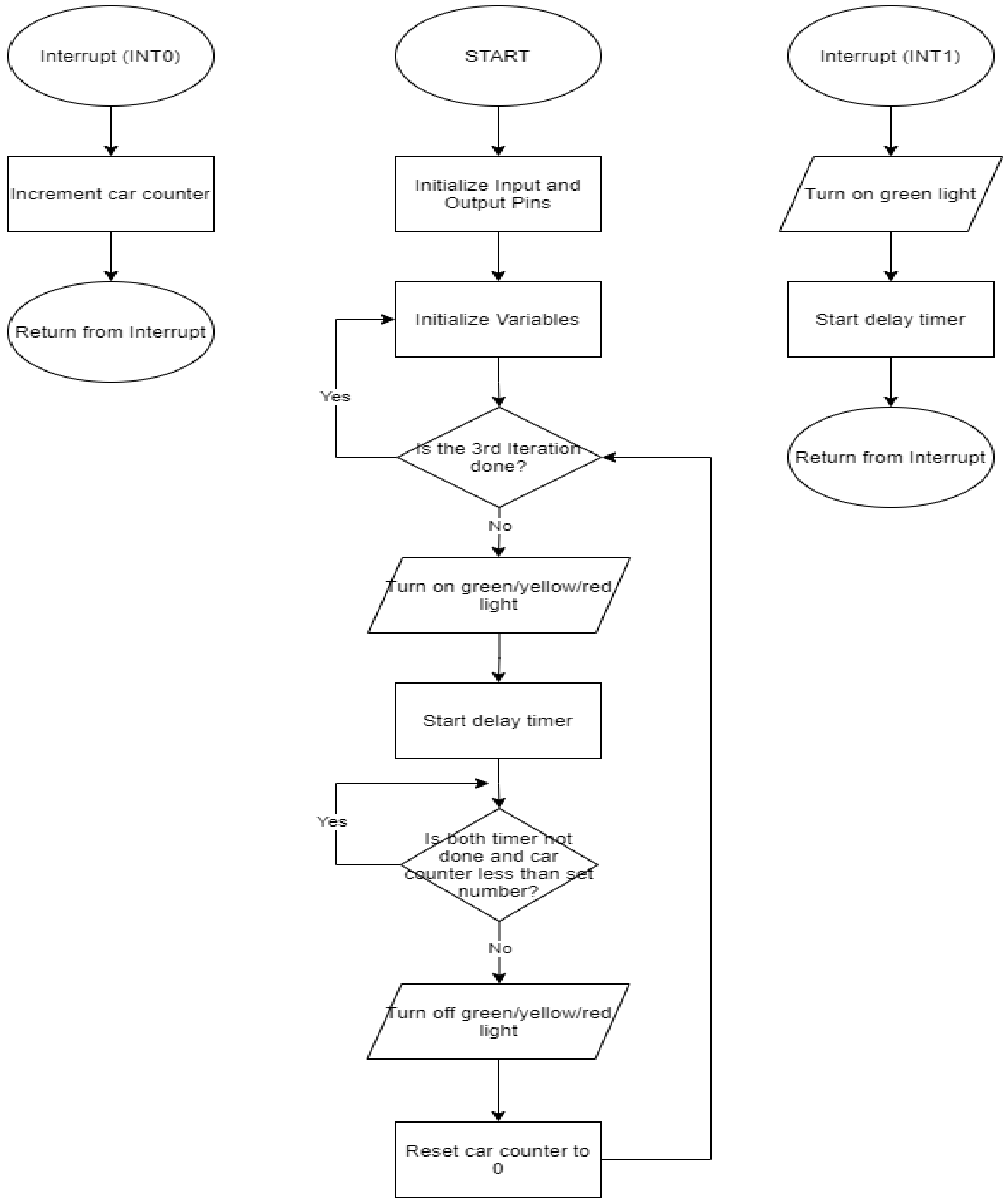
i. Schematic/Logic Diagram

The switches (SW1 and SW2) will simulate the functionality of the Emergency Override (SW2) and the Inductive Loop sensor (SW1) and these switches are connected to INT1 and INT0, respectively. The switch works by sending a low pulse to the pin to signal the actuation of the momentary switch. To avoid switch bounce, a capacitor is placed before the pins.



Schematic

b. Flowchart


Interrupt (INT0)
Increment car counter
Return from Interrupt
START
Initialize Input and Output Pins
Initialize Variables
Is the 3rd Iteration done?
Yes
No
Turn on green/yellow/red light
Start delay timer
Is both timer not done and car counter less than set number?
Yes
No
Turn off green/yellow/red light
Reset car counter to 0
Interrupt (INT1)
Turn on green light
Start delay timer
Return from Interrupt

c. Program/Code/Algorithm Description

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile char count,digit; /* set count as volatile  because the value will change in the ISR */
int delay[3] = {0x0a,0x02,0x0a}; /* delay for green, yellow, red */

int timeDelay(int x) /* set timer1 as 1 sec timer */
{
        int sec;
        for (sec = 0; sec < delay[x]; sec++)
        {
                TCNT1 = 49911;
                TCCR1B |= (1<<CS12) | (1<<CS10);
                while (((TIFR1 & (1 << TOV1)) == 0) && (count < 5));
                TIFR1 |= (1<<TOV1);
        }
        count = 0;
        return 0;
}
int light(int y) /* turn on LED */
{
        DDRB = 0xff;
        PORTB = ~digit;
        timeDelay(y);
        return 0;
}
ISR(INT0_vect) /* interrupt that handles counting of cars */
{
        if (digit == 0x04)
                count ++;
}
ISR(INT1_vect) /* interrupts the loop to turn on the green light */
{
        int digit_save;
        digit_save = digit;
        digit = 0x04;
        light(0);
        digit = digit_save;
}
int main(void)
{
        DDRD = 0x00;
        EICRA |= (1<<ISC11) | (1<<ISC01);
        EIMSK |= (1<<INT1) | (1<<INT0);
        sei();
        while (1)
        {
                int i;
                digit = 0x04;
                for (i = 0; i < 3; i++)
                {
                        light(i);
                        digit>>=1;
                }
        }
}
```

d. Work Assignment of Members

V. Test & Results

When testing the functionality of the switches to initiate the counting of the cars, the shortening of the time of the green light is immediately executed without racing the set amount of counts. Upon further troubleshooting, the problem came from the switch bounce, which is the oscillation of the input due to the mechanical spring action between contacts of the switch, that will trigger the interrupt multiple times which leads to erroneous reading/executing of the pins/interrupt. Therefore, to avoid switch bounce, adding a capacitor to the input pin will 'clean' the input form switch bounce.

Another problem that arose from the project development process is the executing of the shortening of the on time of the green light. This problem took several days to troubleshoot but the reason behind it not executing is because the variable for counting the number of cars isn't defined as a *volatile* variable. Without setting the variable as volatile, any changes done on the variable in the ISR (Interrupt Service Routine) would be disregarded because interrupt handling makes it so that the SREG (Status Register) would not be changed when servicing the interrupt.

VI. Conclusion & Recommendation

In conclusion, stoplights are the primary way of controlling the flow of traffic and adding additional features to make controlling the flow of traffic more beneficial to the drivers and for emergency situations. But using PLC (Programmable Logic Controller), while more practical, is more expensive. Thus, developing microcontroller based stoplights makes adding additional features easier and implementing it is cheaper.

The recommendations that the developers would like to make is to implement the project into a 4-way stoplight system, meaning it can be used at a 4-way intersection. Using imaging systems to count the number of vehicles is more reliable and can contain more data than using an inductive loop sensor.

VII. Reference

a. Goel, Akshat, Microprocessor Projects, August 3, 2018, engineering.eckovation.com/microprocessor-projects, March 10, 2021 Traffic Light Control Electronic Project using IC 4017 & 555 Timer, electricaltechnology.org/2014/10/traffic-light-control-electronic-project.html, March 10, 2021

VIII. Appendix

a. Bill of Materials

- Arduino Uno - 300 php
- PNP transistor (3 pcs.) - 3 php
- LED (3 pcs.) - 1.5 php
- Push button (2 pcs.) - 2 php
- Capacitor (2 pcs.) - 12 php