1 Introduction

The traffic control system programmed on a microcontroller will be deployed on a one-lane entrance to a freeway, and it's made up of 4 sub-systems, a normal traffic light, a configurable traffic light, a speed monitor and a red light camera.

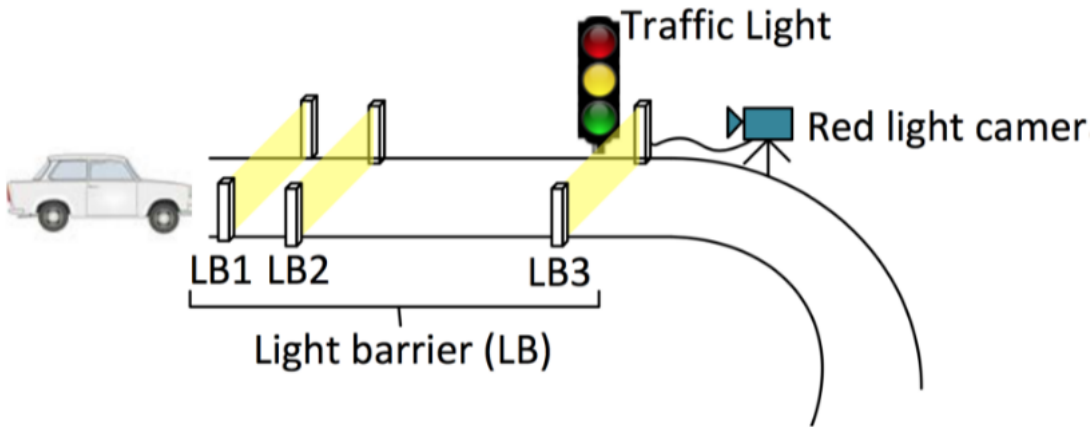

Figure 1: Traffic Control System

The traffic light system will control the flow of vehicles at a one-lane entry to a freeway, with the function of changing the time of each signal based with the input of a potentiometer.

The speed monitor system estimates the speed of the vehicle entering by using the time for the vehicle to cross two light barriers (LB1 and LB2) which are placed 20meters apart. This system can estimate the correct speed for up to 10 vehicles between LB1 and LB2, and maintain an accuracy of ± 1km/h. A PWM signal will be generated from the microcontroller, where the duty cycle is proportional to the speed of the vehicle.

The red-light camera will be an addition to the non-configurable traffic light system where the red-light camera will get triggered when a vehicle passes a third light barrier (LB3) when the traffic signal is on red. A PWM signal will be generated from the microcontroller, where the duty cycle is proportional to the number of times the red-light camera got triggered.

2 Methods

2.1 Basic Traffic Light

The implementation for Basic Traffic Light consisted of two main sections: timer 1 in CTC operation to achieve a period of 1 second and changing the state of the traffic light.

Timer1 CTC Compare Match A interrupt to change the state of the traffic light. Timer1 was chosen due to it being 16bit and able to count to 1 second before overflowing with a prescaler. A Prescaler of 1024 was chosen so Top value (OCR1A) for 1s was 15625 calculated as shown below

$$1s \div \frac{Prescaler: 1024}{Calculation\ Freq: 16 * 10^6} = 15625\ counts$$

Changing the state of the Traffic light:
During ISR of timer 1 compare match with OCR1A, the control variable of the Traffic light will be updated from 0 to 1 to 2 (Red to Green to Yellow, respectively) and it will check for the control variable and turn the LEDS on and off accordingly.

2.2 Configurable Traffic Light

Task 2 Consists of 4 main sections: A super-loop that triggers the traffic light and configuration light, a push button to toggle between configuration mode and cycle mode, timer1 on CTC mode to change light control signal and a ADC to read the potentiometer Value.

Super-loop:
The Super-loop will be responsible for controlling the traffic LEDs and the configuration LED, it will also enable/disable external Interrupt Masks based on the situation. The on and off of LEDs is controlled by setting specific output pins to high/low depending on the control variable. The mask register for the external interrupt for INT0 (push button) will be set to high only when the system is in configuration mode, or the traffic light is red.

Push button:
The push button's main function is to switch the operation mode of the system between Cycle mode and Configurational mode. The Push button will be setup as external interrupt on Port B2 (INT 0) and set interrupt flag on rising edge. When switch from Cycle mode to Configuration mode, the ADC will be enabled, initiate the first ADC conversion, and update OCR1A to control the flashing frequency of the configuration light. When switch from configuration mode to cycle mode, the ADC will be disable, and update OCR1A to control the switching time of the traffic light.

Timer 1:
Timer 1 is set to mode 4 in CTC operation with a 1024 pre-scaler using OCR1A as TOP. This will allow a TIMER1_COMPA interrupt period vary between 0.125s to 4s required by the flashing of configuration light and switching time of traffic light depending on the value of time state. Control variable for configuration light and traffic light will be updated during the ISR.

ADC:
A 10bit ADC is used convert potentiometer value to digital signal, the ADC value is used to determine the time state variable which will vary between 1 to 4. Reference voltage use if Vcc (5v), and ab ADC presale of 128 for normal operation. ADC interrupt will occur after completion of ADC conversion and the time state will only update if it has changed.

2.3 Speed Monitor

The implementation for the Speed monitor consisted of three parts: Calculating the speed of the vehicle, the PWM output with duty cycle proportional to the speed of the vehicle and the blinking of the LEDs as the light beams were triggered.

Calculating the speed:
The two light beams are simulated by two push buttons which are connected to port B2 and B3(INT0 and INT1). INT1 and INT2 interrupt flag are set on the rising edge of the input pin. The implementation of the program will store time from when LB1 is breached to the co-

responding LB2 trigger, as we are using Timer0 on CTC mode to keep time, we need to keep count of how many times it overflows (overflowCount) to know the exact time between the switches. We will calculate the total time between the corresponding triggers to by taking the difference between the enter and exit

$$\begin{aligned} Time_{count} &= enterTime - exitTme \\ &= (TCNT0_{enter} + 256 * overflowCount_{enter}) - (TCNT0_{exit} + 256 \\ &\quad * overflowCount_{exit}) \end{aligned}$$

Speed Calculation:

$$Speed = \frac{20}{Time_{seconds}} * 3.6\ (converts\ to\ \frac{km}{h})$$

$$Time_{seconds} = \frac{Time_{count} * prescaler{:}\ 1024}{clockspeed{:}\ 16 * 10^6}$$

PWM output:
The PWM output is produced by timer 1 in mode 14, fast PWM mode with a prescaler value of 256 with ICR1 as TOP. ICR1 is set to 62500 for 1 second period of the PWM output on OC1X. The duty cycle is updated by changing the value of OCR1A, where OC1A is clear on compare match with OCR1A and set at BOTTOM (non-inverting mode for compare output mode).

Blinking LEDs:
Upon push button press, the output pin for the light beam's respective LED will be set to high to turn on the LED. Those output pin will be set to LOW after 6 timer 0 compare match to OCR0A.

Redundancy:
For cases where more than 1 car enter, to prevent time overwriting itself on all following cars an array of enterTime was created to store the values of enter times. Within INT0 ISR it will update 'vehicleEnterCounter' by 1 every time a car enters after storing the current car's enterTime in the index of the enterCount modulus 10 (to prevent index value exceeding the length of the array). When a Car reaches LB2, INT1 ISR exitCount will be used to determine the index of the car's enter time in the enterTime array, it then increased by 1 and used to calculate the speed to output to PWM.

2.4 Red Light Camera

The Implementation of red light camera is the combination of the basic traffic light (describe in 2.1), and the use of a push button to simulate LB3 and a PWM output that is proportional to the number of vehicle triggered the red light camera. It's consisted of: traffic light function, a push button to simulate LB3, and a PWM output.

Traffic Light function:
The Approach is like the approach in 2.1, except instead of control variable for LED is updated by TIMER1_OVF interrupt which is sets the flag on timer 1 reach the value of ICR1.

Push Button (LB3):

The push button's function is to simulate LB3 breach when the button is pressed. The push button is set up as an external interrupt on PORT B2 (INT0), which set the interrupt flag on the rising edge of the input pin. On INT0 ISR, the program will check the control variable for the traffic light. If the traffic light is red light, OCR1A will increase by 625 to increase the duty cycle of the PWM output, and initiate the blinking sequence on the white LED.

PWM Output:
The PWM output is set up exactly the same as it is in Speed Monitor (2.3) with a period of one second, and duty cycle determine by OCR1A with output pin on OC1A.

Timer 0:
We are using timer 0 with a prescaler of 1024 and a OVF counters to achieve a timer period of .125s for triggering the "infrared camera" on and off. Timer 0 is set up in mode 4, CTC operation with OCR0A as top, on every 7th CTC compare with OCR0A as 255, the OCR0A value will be updated to 161 to achieve an interrupt will occur very 0.125second to toggle the white LED, and OCR0A will be updated back to 255 in the next TIMER0_COMPA ISR.

3 Results

To ensure the red-light camera will respond under 10ms, and to ensure the error associated with the speed in speed monitor system is no greater than 1km/h for up to speed of 150km/h. To achieve this result, all input push button are IO driven interrupts, all timing utilises timers, whilst reducing the amount of operation within the program.

The deadline the program have to meet are as follow, for the red-light camera system, the while LED must start flashing with in 10ms of the push button press; and for the speed camera system, the LED correspond their respective light beam must be turn on with in 1.5ms of the push button press.

By implementing all input push buttons as external interrupts allows the program to execute the task associated with each push button immediately rather than waiting for a trigger.

We only included necessary commands and calculations to be executed in each section of the program. First, we avoided for-loop and while loop which can add unpredictability to the response time of a system. Secondly, we removed all Serial.print() which would requires couple milliseconds to complete if not removed. All of the above will increase the worse case response time of the system.

Specifically, for the red-light camera to flash the white LED By utilising timer 0 to flash the white LEDs instead of using _delay_ms() function, we are able to avoid any dead time in the program where it would be stuck waiting for the delay function to finish. This reduce the worst-case response time for the system.

Specifically, for the speed monitor system, the LEDs for their respective light beam are turn on by button press, and Timer 0 was utilised to turn off the LEDs to avoid dead time, and to reduce worse case response time.

During lab demonstration, the measured response time between button press of LED out pin pin set to high is under the deadline required in the specification. LB1 and LB3 response time is about 10 microseconds, and LB2 have a response time of 100 microseconds.

4 Discussion

All push button inputs are programmed as external interrupts. This will allow for a more responsive system, and prevent the MCU missing certain push button input

ADC is on auto trigger on timer 0 overflow with prescaler of 1024. This will allow the ADC value updated frequent enough such that the system will response straight away, however not too frequent which may result in too many ADC completion interrupt, and affect other part of the system

We floored the speed output of the PWM out, as this will still meet the specification requirement within the assignment brief as it will be within ±1km/h.

For Task 3, we used timer0 with overflow instead of timer 1 to determine time elapsed. It was done as we weren't aware of when ICR1A is reach timer 1 OVF interrupt flag will be set when this code was written, and we did not have sufficient time to fix this section of code. It would be beneficial to use timer 1 overflow as it will allow a larger maximum elapsed and prevent overflow.

We assumed for task 3 a maximum of 10 Cars/Bikes at any time would be between LB1 and LB2 (2.4 Redundancy) due to there only being 20 meters for we assumed there would be 2 meters between each motor bike and 4 meters between each Car. So we only coded for the case where 10 car at anytime could be between LB1 and LB2, but if the case was to change and the distance between LB1 and LB2 was differ, we can easily change the array and array calculation to be able to fit more/less cars.