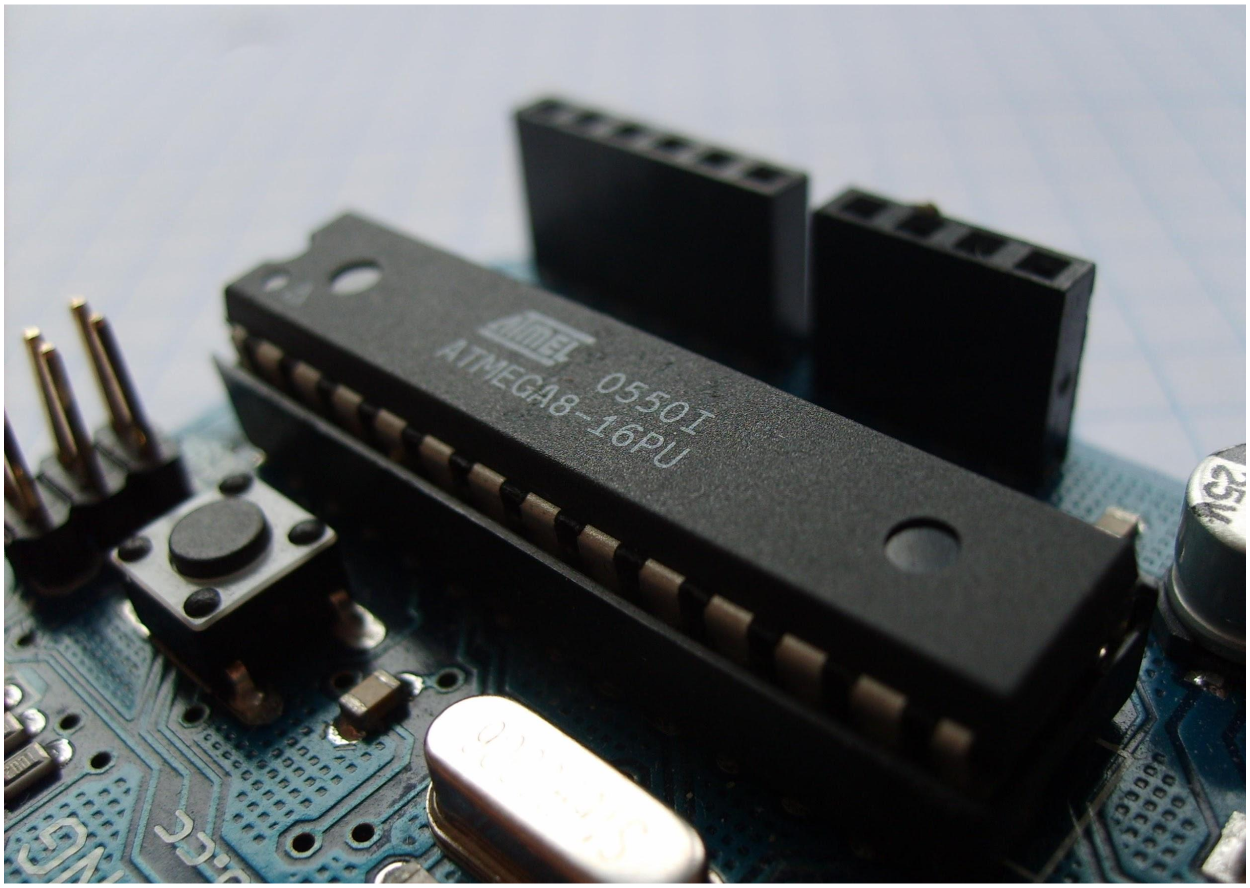
Programming A Traffic Light Controller In Low-Level C

The University of Auckland | Department of Mechanical Engineering



Samuel Reedy & Jean-Daniel Rosset

14/04/2022

MECHENG 313 - The Design of Real-Time Software

INTRODUCTION

The purpose of this project was to design a system that emulates a traffic light system monitoring traffic flow onto an on-ramp seen in modern-day motorways. Therefore this system must include aspects such as speed monitoring, red-light cameras, and vehicle counting. This can be achieved by a simple prototype version using small electronics. The two main elements of the system were speed monitoring and red light camera detection.

A microcontroller is an integrated circuit that contains essential components that would otherwise be found in small computers. These components include a microprocessor, memory, and programmable input/output peripherals which are responsible for the functionality and control of electrical devices, specifically the Arduino Uno. The AVR-328P is a microcontroller chip that is embedded at the centre of the Arduino Uno. The Arduino is a circuit board that makes use of the AVR microcontroller to operate the equipped analog and digital input/output pins which are used to control various other components that may be connected to it. In this system LEDs are connected to the Arduino digital pins, set to output. These can be used to simulate the red, orange, and green lights seen in physical traffic lights. One further LED is connected to a separate pin and simultaneously represents the functionality of a red light camera. A simple push-button is used to represent the number of cars passing through the system, or a second button is added to detect the breach of light barriers for speed monitoring. These are also connected to digital pins on the Arduino board, but set to inputs.

METHODS

Initial Setup:

For both the speed monitoring system and the red light camera system, we make use of two timers, timer/counter1 and timer/counter2. The two timers are configured in different modes to allow the correct functionality of the system. We opted to use timer/counter 1 to control the cycling of the Traffic Light LEDs as well as the output signal to the oscilloscope. Timer/Counter1 is used due to its 16-bit capacity. Control registers TCCR1A and TCCR1B are initialised to zero.

The timer is set to Fast PWM mode (Mode 14) with a prescaller of 256 by enabling bits WGM12, WGM13, and CS12 in register TCCR1A and WGM11 in register TCCR1B. TCNT1 is set to 0 to reset the timer and ICR1 to its top value of 62499. This setup allows for the timer1 overflow ISR to be triggered with a 1s period. When using traffic lights, the Timer/Counter1 interrupt register (TIMSK1) has bit TOIE1 set to high to enable an overflow interrupt (TIMER1_OVF_vect).

Similarly, timer2's two control registers TCCR2A and TCCR2B are set to zero. To configure timer2 in CTC mode with an 8 Prescaler, TCCR2A is configured with bit WGM21 high and TCCR2B with bit CS22 high. TCNT2 is set to zero to reset the timer and OCR2A is set to 250. Setting the interrupt flag register (TIMSK2) to Output Compare A Match flag (1<<OCIE2A) allows the interrupt to occur when Timer/Counter2 reaches a TCNT2 value of OCR2A which occurs every 1 ms at 16MHz (The Clock cycle on the AVR328p). Using the TIMER2_COMPA_vect interrupt, we increment a run-time integer (runTime), which represents the number of milliseconds passed.

For the speed monitoring, bits DDB[1,2,3] are set to high and for the red light camera, DDB[0,1,2,3,5] are set to high in the Port B Data Direction register (DDRB). There are 2 Light Barriers for the speed monitor, so pins 2 and 3 are set to inputs by setting bits DDD[2,3] to low in DDRD. The Red Light Camera only has one Light Barrier, so pin 3 is used by setting DDD2 to low in DDRD. The Light Barriers are represented by using buttons that are configured to detect a rising edge. To set the buttons to the rising edge, bits ISC[0,1]0 and ISC[0,1]1 are set to high in EICRA. The interrupts are enabled for the buttons by setting bits INT[0,1] to high in the External Interrupt Mask Register (EIMSK).

Oscilloscope:

In both the Speed Monitoring and Red Light Camera system, the oscilloscope is connected to pin 9 which is controlled by OCR1A compare match. This is enabled by setting bit COM1A1 to high for register TCCR1A which is the non-inverting mode. When TCNT1 reaches TOP, pin 9 is set to high, then when TCNT1 reaches OCR1A, it is set to low. This allows for accurate control over the duty cycle of the oscilloscope wave.

Speed monitoring:

When the INT0_vect (light barrier LB1) interrupt is called, the current run time is added to a queue of entry times if there is space available. If the first slot in the queue is empty, the run time is reset to 1 in order to help avoid overflows. A boolean is set to true if said boolean is false and the current time is recorded. Then in the super loop, if the current time is less than the recorded time plus 200ms and the boolean is true, the Green LED (PB2) turns on, else, the LED turns off and the boolean is set to false.

When the INT1_vect (light barrier LB2) interrupt is called, if there is an entry time in the queue the red LED (Pin 11) is then flashed the same way as the green one, but with its own variables so it can occur at the same time as the green LED. The speed is calculated based upon the separation distance of the light beams divided by the elapsed time the car is with the beams. This is then converted to kilometres per hour by multiplying the result by 3600. Overall this means the speed in Km/h is calculated by dividing 72000 by the time in ms. The duty cycle is produced by dividing the calculated speed by the maximum speed of 100. Therefore the duty cycle is expressed as a percentage of the maximum speed.

```
speed = 72000 / (float)totalTime;
```

The first entry time in the queue is then removed, and all remaining entry times are moved forward 1 index. The duty cycle only changes when a car successfully passes through both barriers. This means the oscilloscope updates to represent the last car's speed.

Red Light Camera:

The TIMER1_OVF_vect interrupt updates a colour state variable (2->1->0->2->...) every second.

In the super loop, a function is called that sets the appropriate LED to high and the others to low according to the colour state variable (Pin 9, 11, and 13 for states 0, 1, and 2 accordingly).

When the INT1_Vect interrupt is called to signal a breach of LB3, there is a check to determine if the signal LED is already flashing and the red LED is set high. Only if these conditions are met will the FlashLED bool be set to true allowing the white led to activate. The run time is reset here to avoid overflow issues occurring. Then in the super loop, the white LED turns on between 0 and 125ms and 250 and 375ms after being called.

When the INT1_vect (light barrier LB3) interrupt is called, a car count integer is incremented. The duty cycle of the oscilloscope is then set to the car count divided by 100 * 62499 (this is capped at 100%).

```
OCR1A = (62499 * ((float)(count > 100 ? 100: count))/ 100.0);
```

Nontrivial Design Decisions and Assumptions:

For the speed monitoring system, we have designed our software to be based on the assumption that the average car length is approximately 5 metres. As the two light beams (LB1 and LB2) are fixed 20 metres apart under no condition can there be more than 4 cars present between the two light beams. This assumption allows an array storing the entry times of up to and including 4 cars to be initialised to size 4. Implementing this logic makes this system robust and operable in various real-world scenarios.

Due to the limitations of unsigned integers only being able to store a max value before overflowing, we assume that there will not be a period whereby there are constantly cars between the two Light Barriers for more than (49.7 days). This prevents the overflow of the run time registers from occurring. Furthermore, on the basis of real-world scenarios, we assume that no car is going to be travelling at speeds greater than 360Km/h, and thus, both LEDs for the Speed Monitoring system will be on at the same time.

We have considered the likelihood that an incoming car triggers a breach across the LB1 light barrier simultaneously to an outgoing car across LB2. In this case, we have implemented a condition whereby both buttons can be activated at the same time causing the two interrupt routines to be evoked simultaneously. Obviously, due to the priority hierarchy of the interrupts, ISR0 will always be run prior to any other interrupt call. Moreover, another condition prevents the situation where cars are travelling in the opposite way, or in reverse. This means light beam 2 (outgoing) can never be triggered prior to light beam 1 in the first occurrence of a breach.

As well as the points mentioned above, the red light camera system encompasses a white led that is triggered in the event a car breaches LB3 during a red light. To avoid possible confusion conditions have been implemented so that the white-led can only enter its blink sequence (two blinks in rapid succession) once per red light. This condition will hold despite the number of interrupt requests called (from the button press) whilst in the red light state.

DISCUSSION

External Interrupts

Interrupt driven IO is the preferred method used in this project. Interrupt service routines are not time-consuming processors that waste CPU cycles, therefore they are efficient methods of calling a set of instructions without unnecessary delay in the system. INT0 and INT1 are both utilised in the running of this system and are set to trigger an interrupt service routine when the corresponding push buttons attached to pins 2 and 3 detect a rising edge. When the interrupt is invoked, the set of instructions stored within the ISR vector is immediately executed. Once the ISR is completed and no further commands await, the program is returned to its last location in the main loop. Transmission reliability, speed, and efficiency are among the advantages of interrupts over polling.

As well as this, INT0 and INT1 are the second and third highest priority interrupts, making them suitable for the fast response applications of a Red Light Camera and a Speed Monitoring system.

Internal Interrupts

The oscilloscope is connected to pin 9 on the Arduino board. This pin is controlled by the value stored in OCR1A, meaning that the pin can be set high and low according to the PWM timer setup and OCR1A max value. Using OCR1A to output the signal to the waveform eliminates unnecessary delay as it encompasses the internal interrupt from timer1.

Timer/Counter2 runs on a 1ms interrupt loop, this means the run time of the system can be accurately monitored and used at any time to provide precise control of LED blinking as well as to calculate time periods between two interrupts. As the Timer/Counter2 interrupt priority falls below the external interrupts of the pushButtons, the response of the system remains fast and the current time that is accessed in the INT0 and INT1 interrupts is the time that occurred when the interrupt was called.

Speed Monitor

To ensure that an accurate value for speed is calculated, we have opted to use a float for the final speed which can give us a high degree of accuracy (up to 7 d.p.). Although floats are much slower to perform arithmetic than integers, there are only 3 calculations to perform when calculating the speed, calculating the duty cycle, and then applying the duty cycle. And as this only happens once when LB2 is breached, there are no repercussions from using a float to achieve higher accuracy.

Red Light Camera

For the RED light camera, the use of INT1 means that the duty cycle of the oscilloscope can be set within just a few checks and calculations. Although the oscilloscope does not update immediately due to the limitation of TinkerCAD, all the appropriate values are set to represent the correct PWM waveform. As well as this, the 1ms timer ensures that the LED turns on as quickly as the Super Loop can perform, ensuring the response time of the LED light remains well under 10ms.