CEG 4330/6330 Laboratory 2

Timer and ADC Control

PURPOSE

In this three-week lab, you will practice the usage of UNO timer output compare feature for waveform generation. You will also use the UNO ADC for analog inputs.

PRELAB (30%): Complete the prelab assignments as teams. Remove any syntax error before you turn in prelab codes.

(1) (Week 1: 10%) Write the code for Experiment (1). Identify timer 0 registers that you will use for Experiment (2) and explain how to control the frequency and duty cycle by using those registers.

(2) (Week 2: 10%) Write the code for Experiment (3).

(3) (Week 3: 10%) Draw a complete circuit diagram and then write the code for Experiment (5).

EXPERIMENT (70%)

(1) (5%) Develop a UNO sketch that reads two values, pin and value, from the PC keyboard by using the IDE serial monitor and then calls an Arduino function analogWrite(pin, value), where pin is 3, 5, 6, 9, 10, or 11, while value is 0, 1, 2, …, 255, so as to generate a square wave on the corresponding digital pin. Use an oscilloscope to observe the waveforms on digital pins. Record the frequency on each pin.√ Note that UNO digital pins 5 and 6 are connected to the 8-bit timer 0, pins 9 and 10 are connected to the 16-bit timer 1, and pins 3 and 11 are connected to the 8-bit timer 2.

(2) (25%) Develop a UNO sketch that reads two positive integers, frequency and duty cycle, from the PC keyboard and then, based on the fast PWM mode, generates on digital pin 6 a square wave with that frequency and duty cycle. When an input frequency cannot be achieved, use one that is as close as you can get. There is no need to consider (near-)zero or (near-)100% duty cycle. Use an oscilloscope to observe the waveforms. Identify the range of frequency values that you are able to generate.√√ Note that pin 6 is connected to the timer 0 channel A output compare pin.

(3) (10%) Modify the previous sketch so that the square wave is generated on digital pin 10 instead. Again use the fast PWM mode. Use an oscilloscope to observe the waveforms. Record the frequency range that you are able to support. The range should be as large as possible.√ Note that pin 10 is connected to the timer 1 channel B output compare pin.

(4) (10%) Modify the previous sketch so that, in addition to the previous functionality, the code also *continuously* measures the frequency and duty cycle of a square wave input from digital pin 8, and then prints out the measured values to the serial monitor, when the measured values are quite different from the previous value. To verify the code operation, connect pin

10 to pin 8 and use an oscilloscope to observe the waveforms.√√ **Hint:** you may use the Arduino function pulseIn( ).

(5) (20%) Connect a force-sensitive resistor (Sparkfun SEN-09376) and another resistor (as a voltage divider) to an analog pin. The sensor output voltage depends on the amount of force placed on the sensor. Connect an LED and a resistor in series to a digital pin. Use a signal generator to produce a (500 to 1,000 Hz) square wave and connect it to digital pin 8. Write a UNO sketch so that a PWM signal is generated on the pin that the off-board LED is connected to. The frequency of the PWM signal should be fixed at a particular frequency while its duty cycle is proportional to the ADC input voltage. In addition the LED should flash at a frequency that is equal to $\mathbf{F_{in}}$/1,000, where $\mathbf{F_{in}}$ is the frequency of the square wave connected to digital pin 8. For example, if $\mathbf{F_{in}}$ is 1 KHz, then the LED should flash at 1 Hz. In that case, the LED is off for 0.5 second, and then "on" (brightness controlled with the PWM signal) for 0.5 second.√√

Keep all of your source codes. They are needed later on for system integration.

Extra Credit (10%): No partial credit for this part.

Modify the sketch in Experiment (4) so that the Arduino function pulseIn( ) is not used for the measurement of frequency and duty cycle. Use input capture interrupt instead.