The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project

Laboratory Exercise 1: AVR and ARM Interfacing

(Deadline: Check the course information)

Objective:

To develop C programs to cooperate with different external devices under AVR and ARM platforms.

Equipment:

Atmel Studio 7 (software)
Atmega328p (hardware)
Keil uVision5 with ARM support (software)
STM32F103RBT6 (hardware)

Procedure:

Section A: Generate a wave (PWM pulses) by using PWM (Pulse Width Modulation) in an Atmega328p (AVR) microcontroller.

Write a C program to send pulses (generate a wave) by using PWM mode. You should use Timer 0 in Fast PWM mode and the frequency of the generated wave should be 500 Hz. The duty cycle of the wave should be 50%. You may connect a LED to the pin which generates the wave. You can change the duty cycle to control the brightness of the LED. Note that you will not see the LED is flashing since the frequency is too high.

Section B: Generate a wave (PWM pulses) from the AVR microcontroller and capture it back to measure its pulse width.

You are required to write a C program and it has two parts.

1. It gets an integer from Tera Term and sets it as the pulse width (in clock cycles) of a wave generated by the AVR microcontroller (see Figure 1). The frequency of the wave should be 500 Hz. The received integer should be echoed in Tera Term. You can assume that the integer must be between 10 and 99.

2. It gets the wave from its ICP1 pin and measures the pulse width of the wave. Then it sends the value of the measured width to Tera Term again.

If your application can be executed properly, the setting of the pulse width should be equal or very close to the measured pulse width.

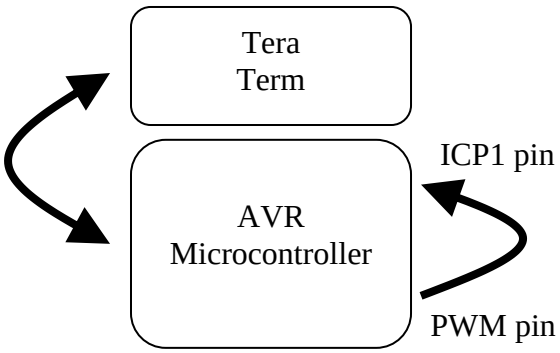

Figure 1

Section C: Generate a wave (PWM pulses) by using PWM (Pulse Width Modulation) in a STM32F103RBT6 (ARM) microcontroller.

Repeat Section A but this time you use the ATM microcontroller.

Section D: Generate PWM pulses (a wave) from the ARM microcontroller and capture it back to measure its pulse width.

Repeat Section B but this time you use the ATM microcontroller.

Section E: Generate a wave (PWM pulses) from the AVR microcontroller and capture it from the ARM microcontroller to measure its pulse width.

You are required to write two C programs.

1. The first C program gets an integer from Tera Term (PC1) and sets it as the pulse width (in clock cycles) of a wave generated by the AVR microcontroller (see Figure 2). The frequency of the wave should be 500 Hz. The received integer should be echoed in Tera Term. You can assume that the integer must be between 10 and 99.

2. The second C program gets the wave from its ICP1 pin and measures the pulse width of the wave. Then it sends the value of the measured width to Tera Term again.

If your application can be executed properly, the setting of the pulse width should be equal or very close to the measured pulse width.

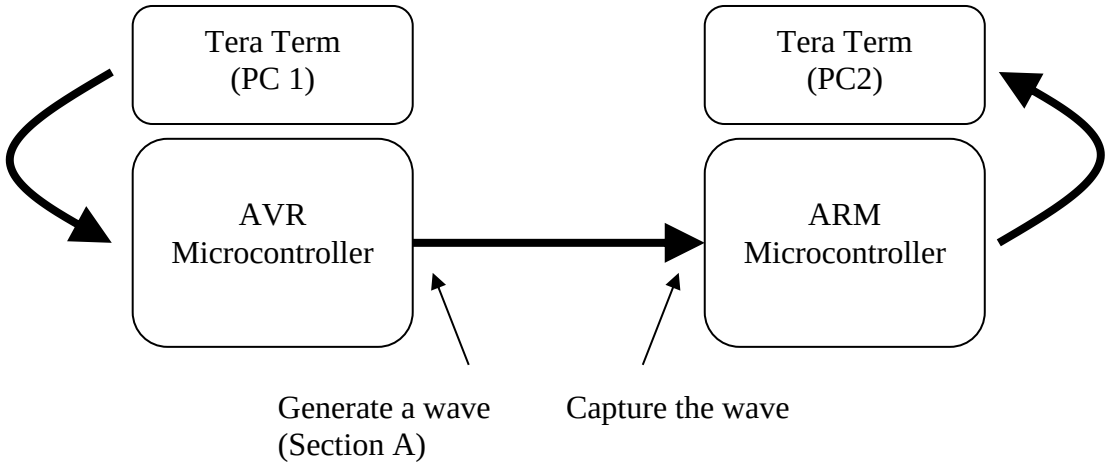

Figure 2

Section F: Generate a wave (PWM pulses) from the ARM microcontroller and capture it from the AVR microcontroller to measure its pulse width.

Repeat Section E but this time the ARM microcontroller generates a wave and the AVR microcontroller captures it and measure its pulse width.

Section G: Capture the light intensity of photo-resistors to control a Tri-color LED by using an AVR microcontroller.

A photo-resistor (see Figure 3 and 4) is a sensor that changes its resistance depending on the amount of light that hits them. It is also known as photocells or light-dependent resistors. You should connect one end of the resistor to the Arduino ADC pin then measure the change in resistance by checking the voltage on the pin. You should write a simple program to check whether you can identify the change of the voltage of the photo-resistor or not.

A tri-color LED (see Figure 3 and 5) with four legs is a common cathode RGB LED. The LED has separate red, green and blue elements inside and one common ground (the cathode). The brightness of its three colors can be changed by applying 500 Hz PWMs with different duty cycles on these three color pins. You may freely choose which PWM pins to generate PWMs. Note that you should use AVR PWM pins to generate 500Hz-PWMs to the LED (i.e., you should not write a program to control the high and low of the output pin.).

The AVR microcontroller connects to the LED and three photo-resistors (see Figure 6). Write a C program to get the light intensity of those resistors through ADC (Analog-to-Digital Converter). Note that you should use an interrupt to get the data. Then use such data to generate three waves by using PWM (use Timer0, Timer1 and Timer2 in Fast PWM mode) to control the brightness of three colors of the LED. Note that the brightness of three colors of the LED is depended on the light intensity of those resistors. It means if the light (from the surrounding) received by those resistors is bright, the brightness of three colors of the LED is high and vice versa. For simplicity, you can set a threshold the light received by those resistors. If the light received by a photo-resistor is bright (normal), you can set the duty cycle of the corresponding PWM signal as 100%; otherwise (the photo-resistor is covered by hand), yo can set the duty cycle as 0%.

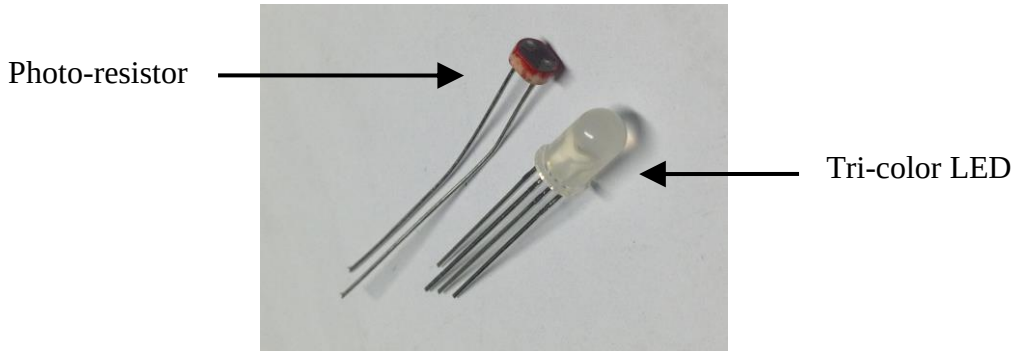

Figure 3

Figure 4

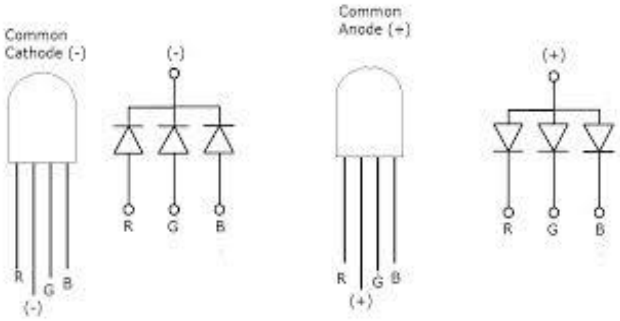

Figure 5

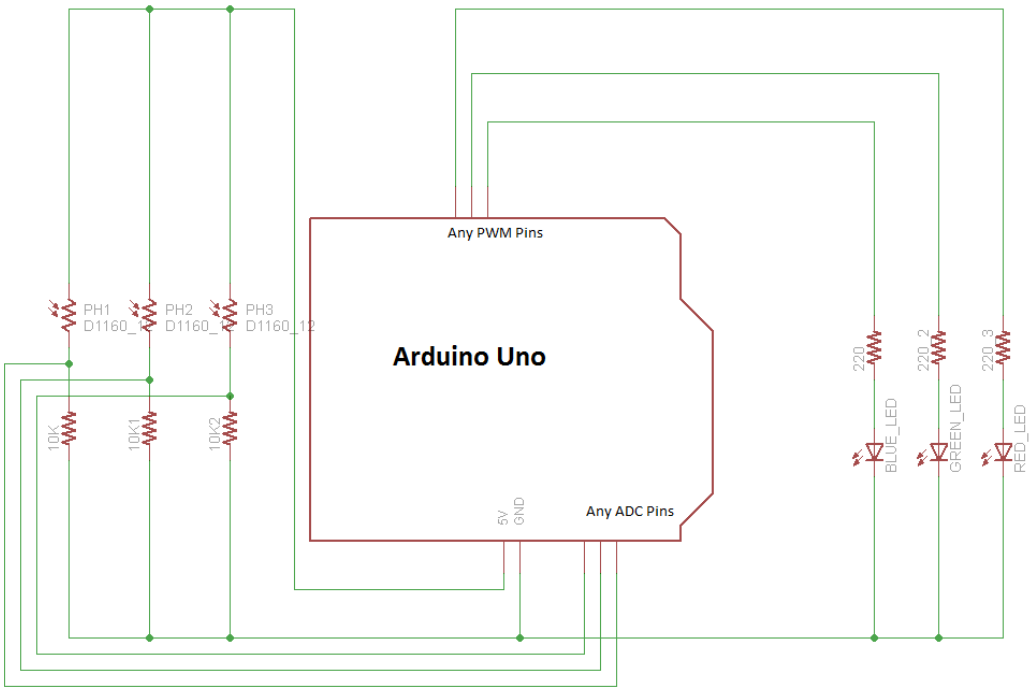

Figure 6

Section H: Capture the light intensity of photo-resistors to control a Tri-color LED by using an ARM microcontroller.

Repeat Section G but this time you use the ATM microcontroller.

Demonstrate your applications in Section E, F, G and H to our tutors or technicians.

Instructions:

1. You are required to demonstrate your programs to our tutor or technicians.
2. Zip all programs (including the whole projects) in Section E, F, G and H to a single file, and submit it to Blackboard.
3. Deadline: **Check the course information**.

Lawrence Cheung
December 2018

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

EIE3105 Integrated Project

Laboratory Exercise 2: PID Control

(Deadline: Check the course information)

Objective:

To develop C programs to make use of the real-time counter and PID control.

Equipment:

Robot Car

Procedure:

Write a C program so that the robot can move on a straight line. You need to do it on two different straight lines. They are on the ground and the length of the straight lines is the same as the width (the longer side) of the arena used in Demonstration 2. To make sure your robot car can move on a straight line properly, you need to monitor the speed of two wheels. You need to change the speed of two wheels if the car is not moving properly. No sensors should be used in this exercise.

You will score full marks if your robot car can move on top of two straight lines from the beginning to the end.

Demonstrate it to our tutors or technicians.

Instructions:

1. You are required to demonstrate your programs to our tutor or technicians.
2. Zip all programs (including the whole projects) to a single file, and submit it to Blackboard.
3. Deadline: **Check the course information**.

Lawrence Cheung
January 2019