Laboratory Report

To: Dr. Randy Hoover

From: Sam Pfenning, Pratik Kunkolienker

Subject: Lab Assignment 5: PWM motor control

Date: April 5, 2019

Introduction

Lab 5's goal was use Timer0 to generate a PWM signal to run the motors on the Elegoo robot using the LM298 motor controller module and the provided header file.

Equipment

The equipment for the lab was used as follows:

1. Arduino UNO R3
2. LM298 motor controller module
3. 4x motors
4. A bread board
5. A USB-A to USB-B cable to program the Arduino
6. Connector cables
7. A computer with AtmelStudio or avr-gcc and avrdude installed

Implementation

The lab was implemented using AVR C. Compiled using avr-gcc and uploaded using the avrdude utility on a Linux system.

A separate header file containing all the functions pertinent to initializing Timer0 and setting up the correct frequency was made so it would be easier to reuse the code for the upcoming labs.

The *pin_map.h* file contains pin mappings from the appropriate bits on in the registers and PORTS to the peripheral sensors.

It must be noted that for the motor controller to work the PWM pins OCRA and OCRB is connected to *EN_A* and *EN_B*. For the left side to work either *IN1* or *IN2* must be set high. The motor won't work if both the pins have the same logic level. The same applies to *IN3* and *IN4*. Hence the direction of the motors can be controlled by which pin is set to high and which is set to low.

The Code

The *main.c* files only contains some necessary includes and the *main()* serves as an entry point to whole program. Foremost of these includes is the "L298.h" header.

The *L298.h* includes the motor class description that includes the initialization and the movement functions. The *init* member of the L298 class initializes the Timer0 such that Output Compare Register A and B are cleared on a compare match. It also sets the PWM frequency and the mode of the Timer to fast PWM.Further, it sets the appropriate ports on PORTD to output.

The Circuit

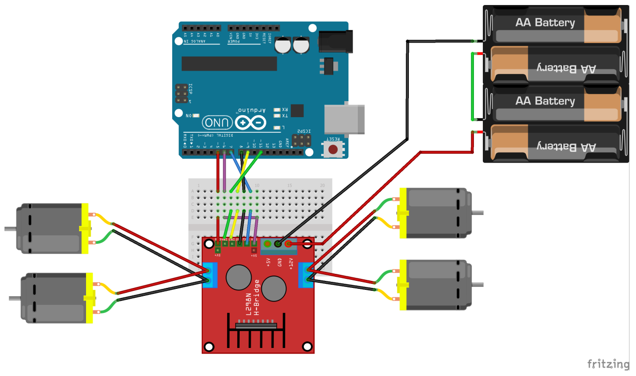


Figure 1: Wiring diagram for Lab 5

Discussion

PWM is a dirty way of getting a appropriate analog voltage level from a digital device that can only output 0V to Vcc. Where Vcc is the operating voltage of the device. Switching regulators are an ideal example of where PWM signals are used to convert DC voltage levels.

Since the DC motors are mechanical devices they only operate in a certain frequency range. If the frequency is too low there isn't enough enrergy supplied to the motor and thus it doesn't move. On the other hand at a higher frequency the, the motor acts a high impedance load and yet again not enough power gets to the load.

Responses

1. As frequency is increased from 15625 Hz to 16 MHz, a significant decrease in torque was noticed. On a carpeted floor the robot would take longer to turn in a circle when it was operating at 16MHz than when it was operating at 15.625KHz

   Consider a loop of area **A** in a constant magnetic field of magnitude **B** and carrying a current **I**. The torque on the loop of **N** turns can be give by:

   $$\tau = \text{NIAB}sin\theta \tag{1}$$

   A DC motor has numerous loops on the rotor and uses permanent magnets in the stator. If **K** is a constant that is affected by the construction of the motor and takes into account the number of turns and the field strength and if the root mean square value of $sint\theta$ is used so as to get the mean value of the torque, equation 1 becomes

   $$\tau_{\text{motor}} = \frac{\text{KI}}{\sqrt{2}} \tag{2}$$

   Clearly, torque is linearly dependant on current.

   Since a motor is basically just loops around a core, it can be simplified as an inductor and some resistance. The values for the winding resistance and inductance were measured

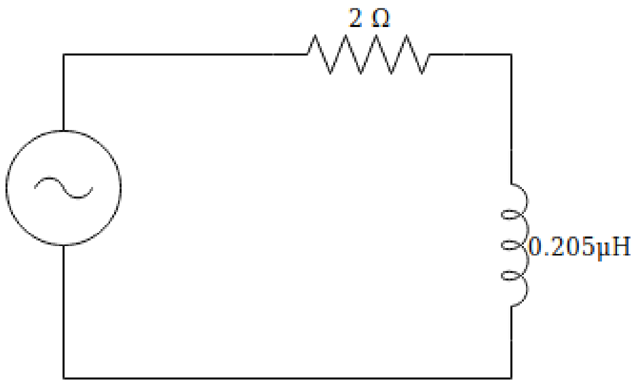


Figure 2: Simplified motor model

   using experimental means. In the frequency domain, the current in the circuit **I** is given by

   $$\text{I} = \frac{\text{V}_{\text{supply}}}{s\text{L} + \text{R}} \tag{3}$$

   Equation 3 suggests that the current is inversely proportional to the frequency $s = j2\pi f$. Substituting the value of current in equation 2 the resulting torque is:

   $$\tau = \frac{\text{KV}_{\text{supply}}}{\sqrt{2}(sL + R)} \tag{4}$$

   And thus at a given supply voltage, $\text{V}_{\text{supply}} = 5 \times D$ where D is the duty cycle, the torque decreases with frequency.

2. ”The ATMEGA328P has an Enhanced Watchdog Timer (WDT). The WDT is a timer counting cycles of a separate on-chip 128 kHz oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value”[Mic].

   Assuming a 5V operating voltage and the WDTON fuse is 1. The WDTSCR must be set as follows to obtain a system reset on 1s of inactivity.

WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
0	0	0	1	1	1	1	0

   WDCE is cleared after 4 clock cycles and thus the resulting register looks like the one shown below

WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
0	0	0	0	1	1	1	0

Contributions

Contributions for this lab are scored ”X/100” for each member of the team, where 100 means the team member contributed maximum efforts in completing the lab, and 0 means the team member contributed little to no effort.

Pratik - 100/100

Sam - 40/100

Appendices

A Code

A.1 main.c

```
/**
 * @file     main.c
 * @author   Pratik Kunkolienker
 * @date     27 March 2019
 * @brief    This is the main file for Lab 5. It declares the motor object
 *      and calls the appropriate functions to complete the required task
 *
 * @details This program was written has two versions. One compiled in C that
 *      uses functions. The other, compiled in CPP and uses classes. The
 *      main loop initializes the timer and calls the appropriate function
 *      such as circ and square turn.
 */

/** @mainpage
 * @section Introduction
 *          This will serve as documentation for lab 5 for the CENG447: Embedded
 *      systems classes for the Spring of 2019.
 *
 *          Lab 5's goal was use Timer0 to generate a PWM signal to run the
 *      motors on the Elegoo robot using the LM298 motor controller module
 *      and the provided header file. The source code can be found
 *      <a href="https://github.com/warlock31415/Embedded-CENG447/tree/u
 *      se_classes/Lab05">here</a>
 *
 *      The clock is set according to the following table:
 *      Set #    | Prescaler
 *      -------- | ------------------------------------------
 *      0    | No clock source
 *      1    | No prescaler
 *      2    | clk/8
 *      3    | clk/64
 *      4    | clk/256
 *      5    | clk/1024
 *      6    |   Ext clk on T0 pin. Clock on falling edge
 *      7    | Ext clk on T0 pin. Clock on rising edge
 *
 *      The set # is passed into the init member function. This sets the
 *      frequency of the PWM according to the table above.
 *
 * @section Video
 * Please follow <a href="https://youtu.be/2wUh4nnQhK4">this link</a>
 *
 * @section Document
 * Download the PDF by clicking <a href="./Lab4.pdf"> here</a>
 *
 * @section Issues
 *-# Half power
 *    + Since two motors are connected to the same output on the motor
 *      controller output, the motors only get half the power each thus
 *      are never at full output.
 *-# Incorrect frequency
 *    + The motors make a whiney noise if too high of a frequency or too
 *    low a frequency is supplied
 */
```

```


#include <avr/io.h>
#include <util/delay.h>
#include "L298.h"

/**
 UART baudrate
 */
#define BAUDRATE 9600

/**
UART baudrate prescaler
*/
#define PRESCALER F_CPU/(BAUDRATE*16UL)-1


/**
* Create a motor object that contains the initilization function and movement
* functions
*/
L298 Motor;

/**
* @details Main function that uses the Motor object to complete the tasks
*     stated in the problem statement
* @returns void
**/
int main(){

  //Initialize motor with a fixed clock cycle
  Motor.init(5);
  // Go in a circle
  Motor.turn_left(80);

  while(1){
  }

  return 0;
}
```

A.2 L298.h

```
/**
 * @file     L298.h
 * @author   Pratik Kunkolienker
 * @date     27 March 2019
 * @brief    This file contains functions declarations for using the L298 motor
 *       driver.
 *
 * @details This file declares the L298 class and all the public and private
 *       functions.
 *
 */
#ifndef L298_H
#define L298_H
  #include <avr/io.h>
  #include <util/delay.h>
  #include "pin_map.h"

  /**
    The L298 class object that has all the motor related functions
  **/
  class L298{
  public:
    void init(char clk);
    void turn_right(int speed);
    void forward(int speed);
    void back(int speed);
    void turn_left(int speed);

    void square_turn(int speed);
    void circ();

  private:
    char map(int d_cyc);
  };


#endif
```

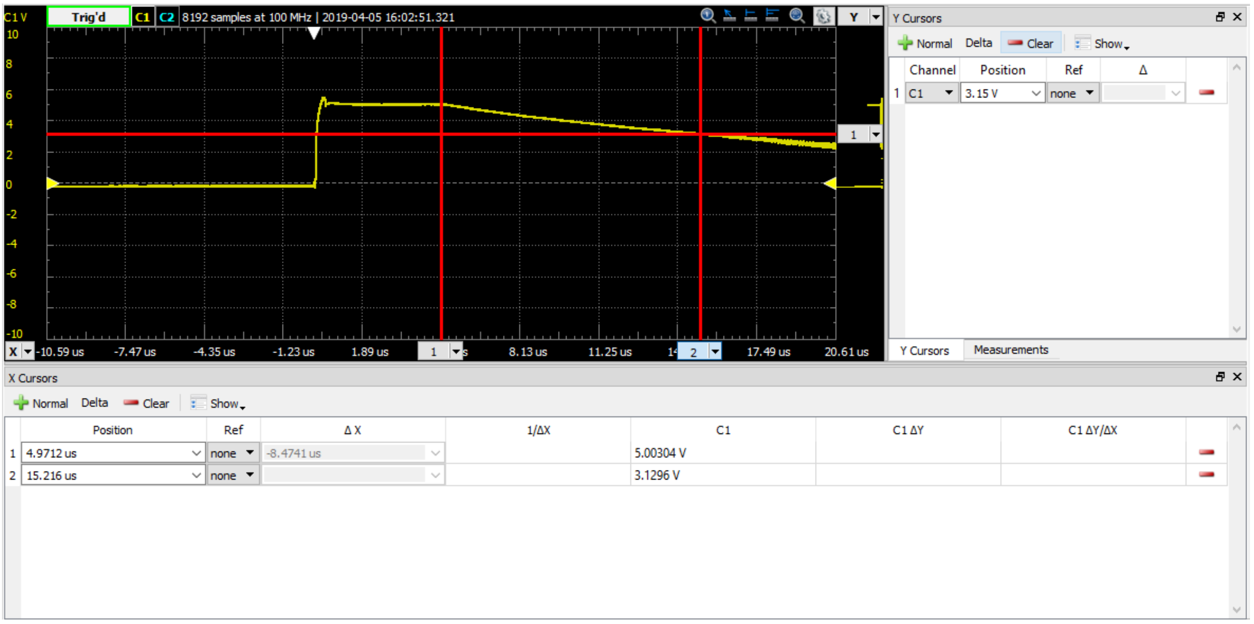
A.3 L298.c

```
/**
 * @file     L298.c
 * @author   Pratik Kunkolienker
 * @date     27 March 2019
 * @brief    This file contains functions for using the L298 motor driver
 *
 */




#include "L298.h"
#include "pin_map.h"
#include <avr/io.h>
#include <util/delay.h>



/**
* @details  This function initializes Timer0 by setting the compare match
*     registers A and B to clear on compare match. Setting the
*     Waveform generation mode bits 1 and 0 and the PWM clock frquency.
*     This function also sets appropriate pins on PORTD as output.
* @returns  void
**/

void L298::init(char clk){
  TCCR0A |= (1<<COM0A1); //Clear OC0A on compare match
  TCCR0A |= (1<<COM0B1); //Clear OC0B on compare match

  TCCR0A |= ((1<<WGM01)|(1<<WGM00)); // Fast PWM mode 3

  TCCR0B |= clk;//(1<<CS02)|(0<<CS01)|(0<<CS00);

  OCR0A = map(0);
  OCR0B = OCR0A;

  DDRD |= (1<<H_A_EN);
  DDRD |= (1<<H_B_EN);

  TCNT0 = 0;



}

/**
* @details     This is a private function which is only available to functions
*        inside the class. It takes a percentage of speed as input 100
*        being max and 0 being the minimum and maps it from 0 to 255.
* @param [in]    duty_cyc Percentage of max speed
* @returns  char duty_cyc*255/100
**/
 char L298::map(int duty_cyc)
{
  return duty_cyc*255/100;
}

/**
* @details  This function sets the direction of the motors such that the left
*    side motors turn forwards and the right side mmotors turn the opposite
*    direction. The function calls the \a map() such that a percentage is
*    mapped to a 0-255 range.
*@param [in] speed A percentage of max speed 0-100%
* @returns  void
**/
void L298::turn_right(int speed){

  PORTD |= (1<<H_IN1);
  PORTB &= ~(1<<H_IN2);
  PORTB |= (1<<H_IN3);
  PORTB &= ~(1<<H_IN4);

  OCR0A = map(speed);
  OCR0B = OCR0A;
}

/**
* @details  This function sets the direction of the motors such that the both
*    run forwards. The function calls the \a map() such that a percentage is
*    mapped to a 0-255 range.
*@param [in] speed A percentage of max speed 0-100%
* @returns  void
**/
void L298::forward(int speed){
  PORTD |= (1<<H_IN1);
  PORTB &= ~(1<<H_IN2);
  PORTB |= (1<<H_IN4);
  PORTB &= ~(1<<H_IN3);

  OCR0A = map(speed);
  OCR0B = OCR0A;
}
```

```

/**
* @details  This function sets the direction of the motors such that the right
*   side motors turn forwards and the left side mmotors turn the opposite
*   direction. The function calls the \a map() such that a percentage is
*   mapped to a 0-255 range.
*@param [in] speed A percentage of max speed 0-100%
* @returns  void
**/
void  L298::turn_left(int speed){
  PORTD &= ~(1<<H_IN1);
  PORTB |= (1<<H_IN2);
  PORTB |= (1<<H_IN4);
  PORTB &= ~(1<<H_IN3);

  OCR0A = map(speed);
  OCR0B = OCR0A;
}

/**
* @details  This function sets the direction of the motors such that the both
*   run backwards. The function calls the \a map() such that a
*   percentage is mapped to a 0-255 range.
*@param [in] speed A percentage of max speed 0-100%
* @returns  void
**/
void L298::back(int speed){
  PORTD &= ~(1<<H_IN1);
  PORTB |= (1<<H_IN2);
  PORTB &= ~(1<<H_IN4);
  PORTB |= (1<<H_IN3);

  OCR0A = map(speed);
  OCR0B = OCR0A;
}

/**
* @details  Calls the \a turn_right() function 4 times such that the robot
*     traces a square whose side length is as long as 3s in equivalent
*     distance. The spedd is set to 0 at the end.
*@param [in] speed A percentage of max speed 0-100%
* @returns  void
**/
void L298::square_turn(int speed){
  forward(speed);
  _delay_ms(3000);
  turn_right(speed);
  _delay_ms(1000);
  forward(speed);
  _delay_ms(3000);
  turn_right(speed);
  _delay_ms(1000);
  forward(speed);
  _delay_ms(3000);
  turn_right(speed);
  _delay_ms(1000);
  forward(speed);
  _delay_ms(3000);
  init(0);

}

/**
* @details  Sets the right side speed a little faster than the left speed
*        so that the robot traces a circle. The radius of the circle is
*      a relationship of the difference between the wheel speeds.
* @returns  void
**/
void L298::circ()
{
  PORTD |= (1<<H_IN1);
  PORTB &= ~(1<<H_IN2);
  PORTB |= (1<<H_IN4);
  PORTB &= ~(1<<H_IN3);

  OCR0A = map(100);
  OCR0B = map(30);
}
```

B Calculating inductance

In order to calculate the inductance of the motor the circuit shown in figure 2 was hooked up on a bread board. A function generator was used to supply the motor with a 5V square wave at 5kHz. The voltage at the terminals of the motor were measured using an oscilloscopes. The results are shown in figure 3



8192 samples at 100 MHz | 2019-04-05 16:02:51.321

Y Cursors

	Channel	Position	Ref	Δ
1	C1	3.15 V	none	

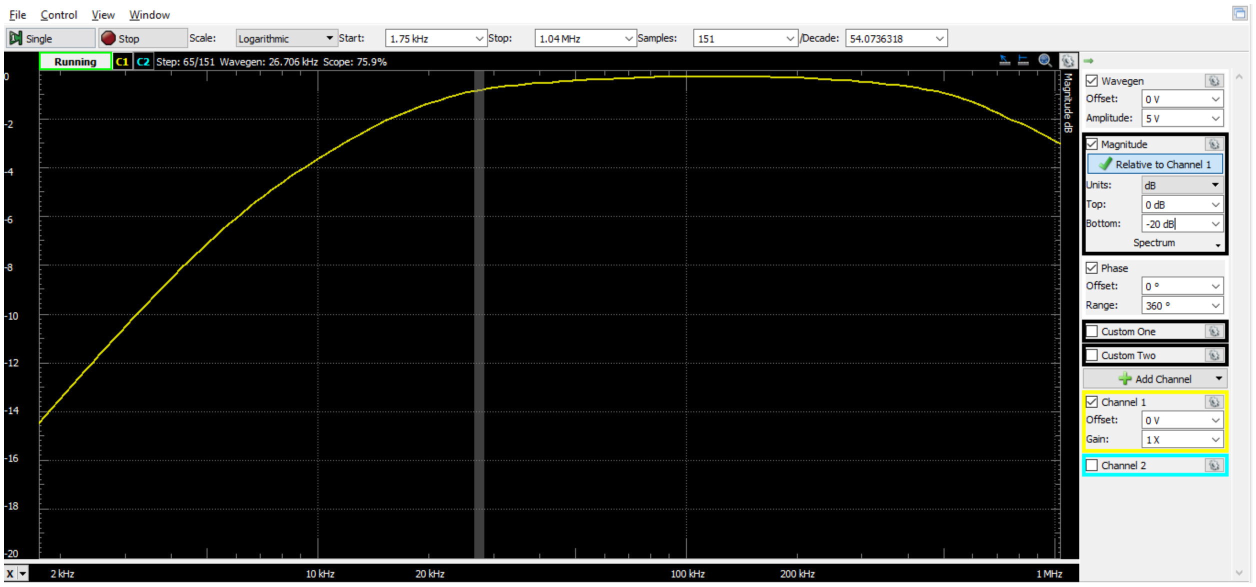
X Cursors

	Position	Ref	Δ X	1/ΔX	C1	C1 ΔY	C1 ΔY/ΔX
1	4.9712 us	none	-8.4741 us		5.00304 V		
2	15.216 us	none			3.1296 V		

Figure 3: Oscilloscope screen dump

The time taken for the voltage to drop to 63% of its original value was measured. Since the time constant is given by $T = \frac{L}{R}$, L can be calculated if R is known. R can be measured using a multimeter.

The resulting corner frequency for the circuit for the calculated time constant ended up being around 16 kHz. This was verified in the bode plot shown in figure 4. Notice the droop in the in gain on the right side. This might be due to stray capacitance from the breadboard where the circuit was built and thus the network looks like a low Q band-pass filter.



Scale: Logarithmic Start: 1.75 kHz Stop: 1.04 MHz Samples: 151 /Decade: 54.0736318

Step: 65/151 Wavegen: 26.706 kHz Scope: 75.9%

Wavegen Offset: 0 V Amplitude: 5 V

Magnitude Units: dB Top: 0 dB Bottom: -20 dB

Phase Offset: 0 ° Range: 360 °

Channel 1 Offset: 0 V Gain: 1 X

Figure 4: Bode plot for a RL network

C Invisible Formatting

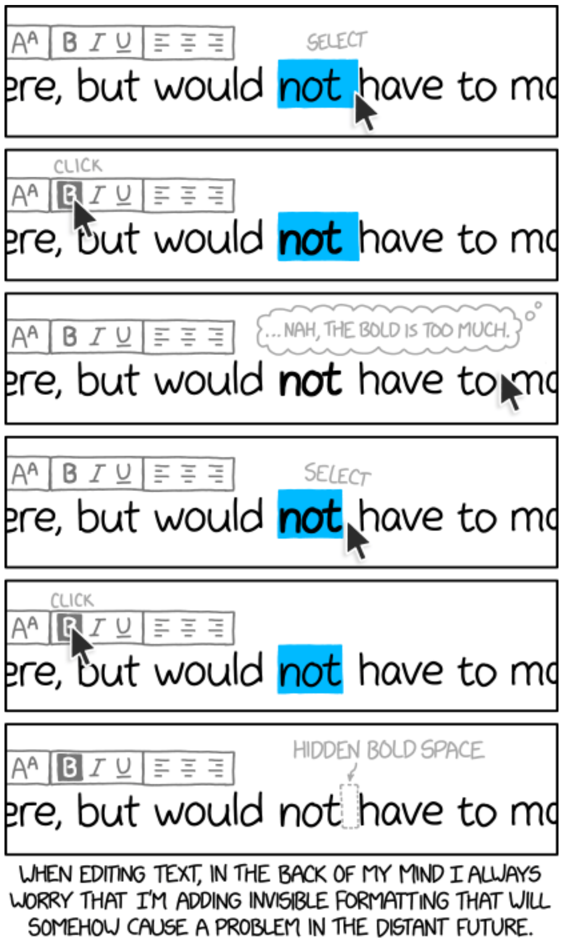

SELECT
ere, but would not have to mo
CLICK
...NAH, THE BOLD IS TOO MUCH.
SELECT
CLICK
HIDDEN BOLD SPACE
WHEN EDITING TEXT, IN THE BACK OF MY MIND I ALWAYS WORRY THAT I'M ADDING INVISIBLE FORMATTING THAT WILL SOMEHOW CAUSE A PROBLEM IN THE DISTANT FUTURE.

Figure 5: I guess that's why we use Latex

Glossary

ATMEGA328P ”A microcontroller chip manufactured by Microchip as a part of their AVR line of microcontroller chips” 4

AVR is a line of 8-bit microcontrollers developed by Microchip and are today found on most Arduinos. They can programmed using C or Assembly 1

References

[Mic] Microchip. *ATmega48P/88P/168P/328P*. Version 8025I AVR 02/09.