CG1112: Engineering Principles and Practices II

Week 4 Studio 2: AT328P PWM Programming (GRADED LAB)

Objectives:

1. Understand how the Atmega328's PWM peripheral block needs to be configured.
2. Develop Low-Level (Bare-Metal) code for the PWM module.
3. Develop code for complete Motor Control (Forward, Reverse, Right, Left)

Equipment Needed:

1. Laptop with Arduino IDE installed
2. Arduino Uno Board + Prototyping Board
3. Motor Driver Chip
4. Battery Pack (Optional, can be replaced with DC Power Supply)
5. Magician Chassis with Motors and Wheels (1 Set of Motors and Wheels is enough)

LAB REPORT:

- Remember to show any necessary workings.
- Waveform images can be captured through your camera or they can be redrawn in your report.
- Submit using Template given in LumiNUS.

Appendix Code:

- The code that is required is given in the Appendix.

1. Introduction

In Week 4, you first learnt about GPIO Programming and how to achieve that using bare-metal programming. You subsequently learnt about Interrupts and how they can help you to better manage events while handling other activities. In this studio, you are going to be using Interrupts to generate your PWM signals.

In the e-lecture (https://youtu.be/Vf7-cub5Q1o), you have gotten an idea (hopefully ☺) on how the Timer module in the microcontroller helps us to achieve our objective. By the end of this studio, you should be able to control both your DC motors and achieve full motor functionality for your robot.

2. Exercise A

2.1 Create a sketch in your Arduino IDE. A sample code for PWM generation is provided in Appendix A. Rewrite that code in your IDE, compile and download it to your Uno board. Observe the PWM waveform on Arduino Pin 6.

LAB REPORT

Q1. Include a copy of the waveform in your report.

Q2. Note down the following parameters from the waveform

Period:

High-Time:

Low-Time:

Duty-Cycle:

Q3. **Calculate** the Duty Cycle based on the formula given in the E-Lecture. If you haven't seen it yet, now would be a good time. ☺

Duty-Cycle:

Q4. Explain why we don't need to do anything in the ISR.

2.2 You are now tasked to generate a complement of the current waveform ('1' becomes '0' and '0' becomes '1').

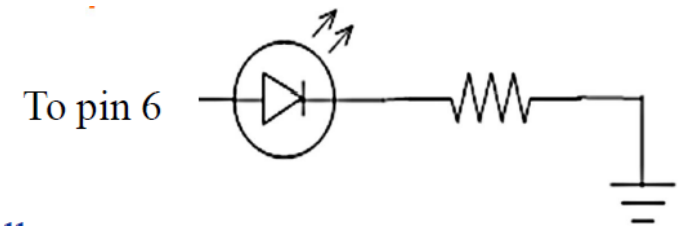
Provide the code change that is need to do this WITHOUT changing the OCR0A register value.

LAB REPORT

Q5. State the changes that you made in the code and provide an explanation on how your changes achieve the desired outcome.

Q6. Include a copy of the waveform in your report.

2.3 Now **revert** back to the original code that has been given to you. Construct the circuit as shown below (Use a 220 ohms Resistor) and observe that the LED lights up. Modify the OCR0A register value and observe changes in the intensity of the LED.



Connection to Arduino Pin 6

LAB REPORT

Q7. Implement a Fade In / Fade Out effect for the LED. You must only change the code in the ISR. Other parts of the code must **NOT** be changed, but you are free to declare additional variables in your code.

Include your ISR code in the report and state the declaration of any additional variables.

It is observed that the fading effect is very fast and you wish to slow it down.

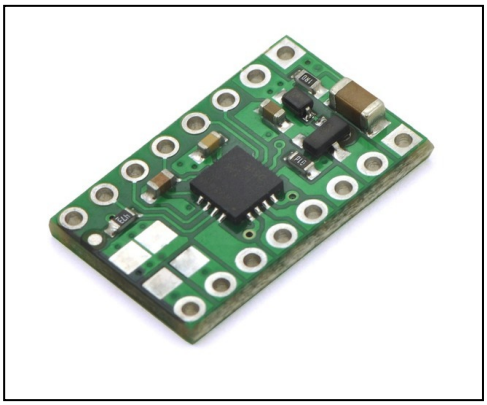
LAB REPORT

Q8. Provide the code snippet for this. You only need to change the value in one register for this effect.

3. Exercise B

In this exercise you are going to be using the PWM signals that you generated to control your DC motor. As you have seen in your Week 4 Tutorial, the microcontroller pin has very limited output current. To drive a motor, you have much higher current requirements, and that's why we need to use a driver chip.

DRV8833 Dual Motor Driver

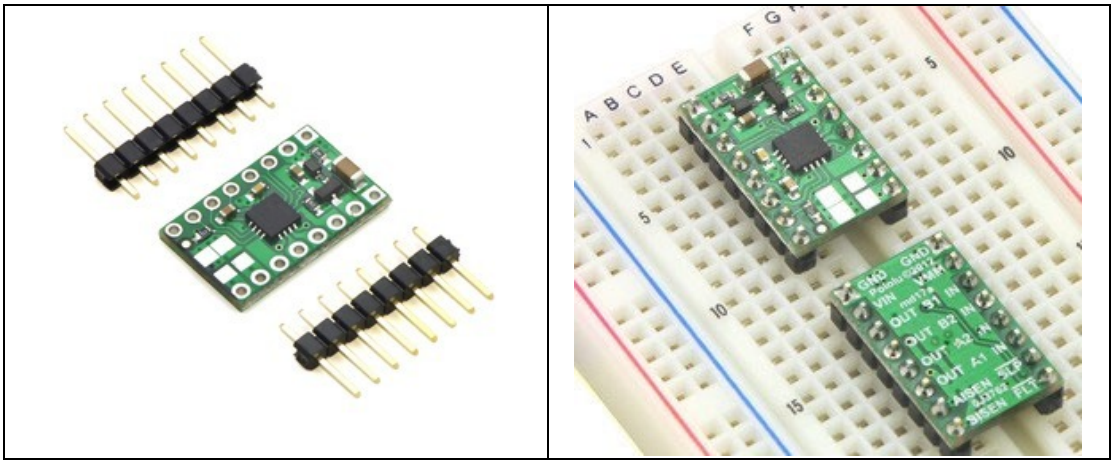


Texas Instruments' DRV8833 is a dual H-bridge motor driver IC that can be used for bidirectional control of two brushed DC motors at 2.7 V to 10.8 V. It can supply up to about 1.2 A per channel continuously and can tolerate peak currents up to 2 A per channel for a few seconds, making it an ideal driver for small motors that run on relatively low voltages.

To learn more about the chip, please refer to the datasheet in LumiNUS.

3.1 Connecting the Motor Driver

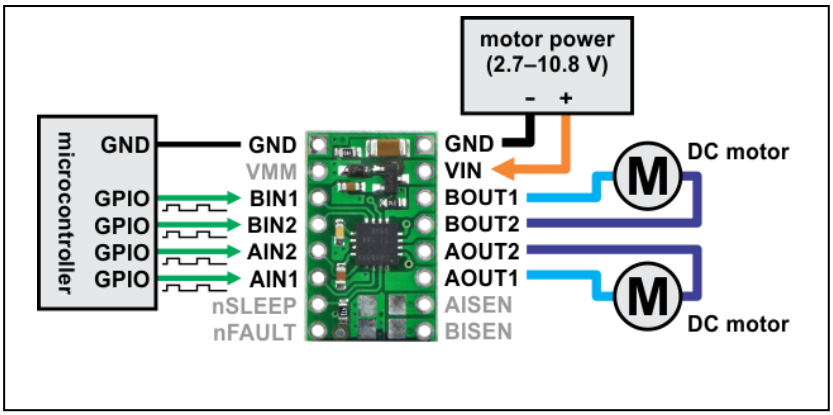
Two 1×8-pin breakaway 0.1″ male headers are included with the DRV8833 dual motor driver carrier, which can be soldered in to use the driver with breadboards. It is up to you to decide how the headers are soldered with the IC chip.



Soldering the headers onto the board

Each of the two motor channels has a pair of control inputs, xIN1 and xIN2, that set the state of the corresponding outputs, xOUT1 and xOUT2; The Pulse Width

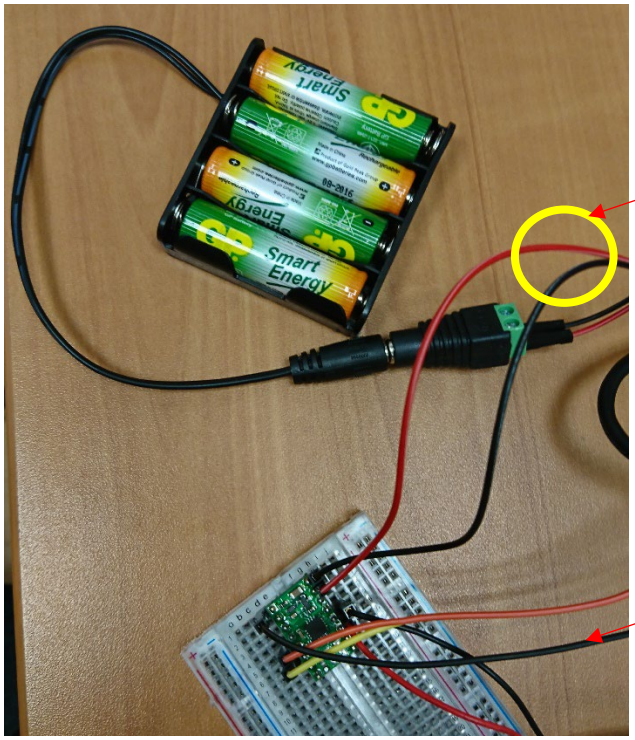
Modulated (PWM) signal can be applied to each of these inputs. The control inputs are pulled low internally, effectively disabling the motor driver outputs by default.



Connection Guide

(The directions right and left are relative to the board with the chip on-top. If you board is soldered the other way, then adjust accordingly. Use the labels on the board as a guide)

- The Motor Power is obtained through the AA Battery Pack. Connect it to the Barrel Connector and tap the '+' and '-' to your chip VIN and GND on the right of the board. (If the Project Kit is not issued yet, please use the DC Power Supply for this step and set it to 10V)
- Connect the GND from the Uno board to the GND on the left of the board.
- Connect AIN1 to Pin 6 of your Arduino. For now, ignore the rest of the connections to AIN2, BIN1 and BIN2.
- Connect AOUT1 and AOUT2 to both pins of you DC motor. Ignore BOUT1 and BOUT2 for now.
- Run the default code as given in the **Appendix A**, but change the **OCR0A value in the setup() to 128.**



You will observe that the Motor spins in one-direction. Change the OCR0A to hold a higher value and observe a corresponding increase in the speed of the Motor.

3.2 Driving the motor in both directions

To drive the motor in both directions, we need to be able to generate two different PWM signals to AIN1 and AIN2 of the driver chip, but only one of them should be active at any time in order to drive the motor in a particular direction. When one of the PWM lines is active, the other line must be driven LOW.

Timer 0 has two separate counters, Counter A and Counter B. We can use both of them for bi-directional control of a single motor.

Additional Connection:

- Connect AIN2 to Pin 5 of your Arduino.

Copy the code that is given in **Appendix B** in to your project. Compile and download the code onto your board. You should observe that the motor is able to move Forward and Reverse continuously.

LAB REPORT

Q9. Examine the code in right_motor_forward() and right_motor_reverse(). Explain what the code is trying to achieve by manipulating the values in the TCCR0A registers.

4. Summary

In this studio, you have been exposed to the concept of PWM generation using the Timer Block.

It is important to note that the code for motor control is provided only as a guide. You are free to develop your own functions for the various type of motor control movements that you wish to implement for your project.

Good Luck! ☺

APPENDIX A

```
#include "Arduino.h"
#include <avr/io.h>
#include <avr/interrupt.h>

#define PIN7 (1 << 7)
#define PIN6 (1 << 6)
#define PIN5 (1 << 5)
#define PIN4 (1 << 4)
#define PIN3 (1 << 3)
#define PIN2 (1 << 2)
#define PIN1 (1 << 1)
#define PIN0 (1 << 0)

void setup() {

  TCNT0 = 0;
  TCCR0A = 0b10000001; // Set OCOM0A to 10 and WGM to 01
  TIMSK0 |= 0b10;      // Enable Int for Output Compare Match
  OCR0A = 25;
  TCCR0B = 0b00000011; // Set clk source to clk/64
  //Set PORTD Pin 6 (Arduino Pin 6) as Output
  DDRD |= PIN6;
  sei();
}

ISR(TIMER0_COMPA_vect)
{

}

void loop() {


  while (1)
  {
  }

}
```

APPENDIX B

```
#include "Arduino.h"
#include <avr/io.h>
#include <avr/interrupt.h>

#define PIN6 (1 << 6)
#define PIN5 (1 << 5)

unsigned int interval = 500;
unsigned long prevMillis;

void setup() {

  DDRD |= (PIN6 | PIN5);
  TCNT0 = 0;
  TIMSK0 |= 0b110;   // OCIEA = 1 OCIEB = 1
  OCR0A = 128;
  OCR0B = 128;
  TCCR0B = 0b00000011;
  prevMillis = millis();
  sei();
}

ISR(TIMER0_COMPA_vect)
{

}
ISR(TIMER0_COMPB_vect)
{

}

void right_motor_forward(void)
{
  TCCR0A = 0b10000001;
}
void right_motor_reverse(void)
{
  TCCR0A = 0b00100001;
}


void loop() {

  right_motor_forward();
  while(millis() - prevMillis < interval);
  prevMillis = millis();

  right_motor_reverse();
  while(millis() - prevMillis < interval);
  prevMillis = millis();
}
```