School of Engineering, Discipline of Mechanics and Mechatronics, pp. 1–9

Lab 2: Actuators
Digital I/O and PWM

MECH2110

Semester 1 2019

Nicholas O'Dell, Johannes Hendriks, Craig Wheeler.

---

1 Introduction

In this lab you will use digital output and PWM to control several actuators which could be used in your projects. These are:

Figure 1: Servomechanism

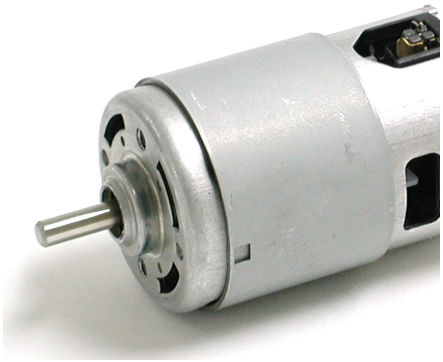

Figure 2: DC motor

Figure 3: Stepper motor

The lab is worth 4% of your course grade and is graded from 0-4 marks.

2 Servomechanism, (1 mark)

Task: Use the <`servo.h`> library to control a servo motor.

A servo motor consists of a DC motor and driver. It has a limited range, normally between 0 and 180 degrees, and it is controlled by varying the width of a pulse (a timing diagram is shown in Figure 4). In this lab, we will use the Arduino library `<servo.h>`.

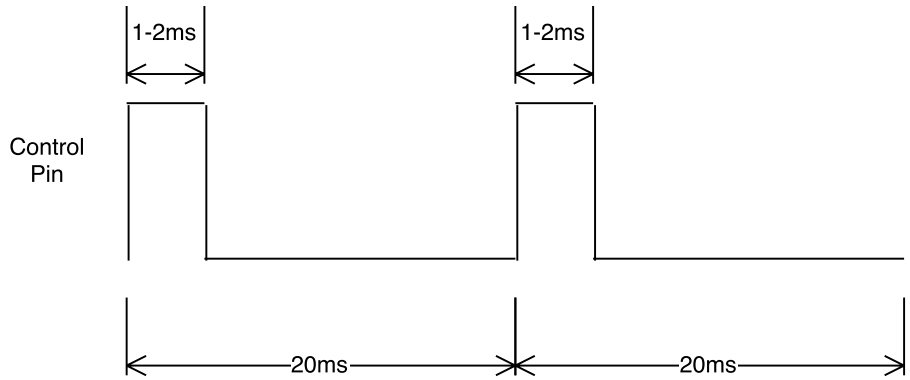

Figure 4: Servo timing diagram, $1ms = 0° \ldots 1.5ms = 90° \ldots 2ms = 180°$.

1) Connect the servo as shown in Figure 5, according to Table 1.
2) Complete the **TO DO** statements in the code, shown in Listing 1 (available on Blackboard in `lab2_files.zip` along with all other code templates in this lab).

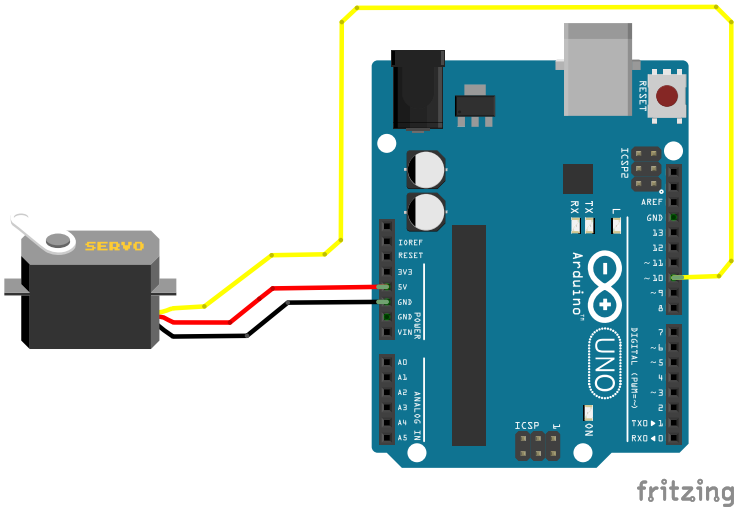

Figure 5: How to connect a servo to the Arduino: Connect the brown wire on the servo to GND on the Arduino, the red wire to 5 V and the orange wire to the digital pin used for control (in this case the connected pin is pin 10).

Table 1: Servo motor pin-out

Wire colour	Purpose	Arduino Pin
Brown	GND	GND
Red	5 V	5 V
Orange	Signal	digital pin xx

i) Change the range of the servo so that the servo sweeps between 0 and 90 degrees.

ii) Add the line of code necessary to physically move the servo, see the `<servo.h>` library reference on the Arduino website (https://www.Arduino.cc/en/Reference/Servo).

3) Run the program and demonstrate to your tutor.

Listing 1: Servo code

```
#include <Servo.h>
#define SERVOPIN 10
Servo myservo;  // create servo object to control a servo

int pos = 0;    // variable to store the servo position
int dir = 0;    // variable to store the servo direction

void setup() {
  Serial.begin(115200);
  myservo.attach(SERVOPIN);  // initialises the servo by attatching the servo on pin
      //...10 to the servo object
}
void loop() {
        //TODO: Change the range to sweep 0-90 deg
    if(dir == 0)  //if forwards
```

```
    {
        if(pos<180) //keep moving forwards
        {
          pos++;
        }else      //start moving backwards
        {
          dir = 1;
          pos = 179;
        }
    }else          //if backwards
    {
      if(pos>0)    //keep moving backwards
      {
        pos--;
      }else       //start moving forwards
      {
        dir = 0;
        pos = 1;
      }
    }
    //TODO: Send the updated position command to the servo
    Serial.print("Position=");  //print to the serial monitor
    Serial.println(pos);        //prints the position to the serial monitor + a new
         //...line
    delay(15);                  // waits 15ms for the servo to reach the position
}
```

3 PWM, DC motor (2 marks)

Task: To achieve bi-directional control of a DC motor.

PWM (Pulse Width Modulation) controls the duty cycle of an input signal to control the average voltage as shown in Figure 6. The pins that can be used to output a PWM signal are indicated with a '~' (pins 3, 5, 6, 9, 10, and 11) on the Arduino. The microcontroller is not capable of producing enough current to drive a DC motor from its outputs, so an external motor driver is used to drive the DC motor given the PWM signal as an input. To send a PWM signal from the Arduino, look up `analogWrite()` on the Arduino reference page (https://www.Arduino.cc/en/Reference/AnalogWrite).

The motor driver used in the lab, the L298N, is configured to control either two DC motors or one stepper motor. The 3 pins connected to the Arduino in Figure 7 control direction and speed. In order to set the motor forwards, IN1 must be set `LOW` and IN2 must be set `HIGH` and vis visa for reverse as shown in table 2, while the PWM signal is sent to ENA pin to drive the DC motor on output **A** using `analogWrite()`.

1) Build the circuit shown in Figure 7, ensuring that the **ENA** pin is connected to PWM enabled pin: **9**, and the pins **IN1** and **IN2** are connected to pins: **3** and **4** respectively.

2) Ensure the **green** and **blue** motor wires are connected to output A of the motor driver, and that the **GND** pin of the Arduino is connected to the **GND** pin of the motor driver.

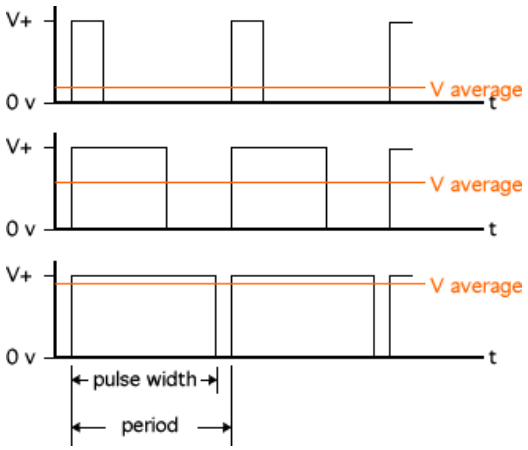

Figure 6: PWM graphic, showing average voltage as a function of duty cycle.

Table 2: L298N Logic table

IN1	IN2	The State of DC Motor
0	0	OFF
0	1	Rotate Clockwise
1	0	Rotate Counterclockwise
1	1	OFF

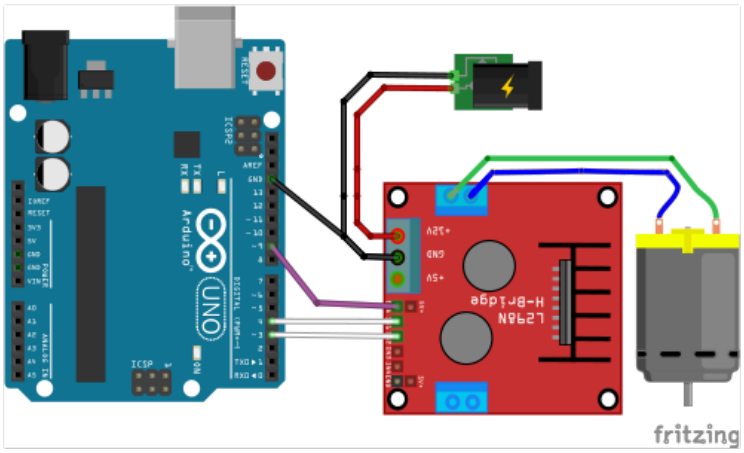

Figure 7: How to How to connect the DC motor and driver to the Arduino: Connect the pin **IN1** and **IN2** to digital pins **3** and **4** respectively; connect **ENA** to the PWM enabled pin **9**. Use an attached jumper wire to connect the GND terminal of the motor driver to the Arduino.

Listing 2: DC motor task.

```
/*
MECH 2110 - Lab 2
DC motor
*/

#define IN1 3
```

```
#define IN2 4
#define ENA 9


void setup() {
  // put your setup code here, to run once:
  //TODO: initialise pins as output.

  goForwards();
}

void loop() {
  // put your main code here, to run repeatedly:
  //TODO:
  //Gradually increase speed
  //Maintain speed
  //Gradually slow down

}


void goForwards(void)
{
  //TODO: Set the pins IN1,IN2 appropriately.
}
```

3) Begin with the code template above in Listing 2 and define the pins **IN1, IN2, ENA** that connect to the motor using `#define`.

4) In the setup function enable the pins IN1, IN2, and ENA as outputs.

5) Write a function `void goForward(void)` that sets the pins IN1 and IN2 appropriately.

6) Write a program utilising these functions that executes the following steps:

   i) Gradually increase the motors speed (duty cycle) from 0 to 255 over (approximately) 5 seconds in the FORWARD direction.

   Hint

   Consider using `for` https://www.Arduino.cc/en/Reference/For and `delay()`

   ii) Maintain this speed for 5 seconds.

   Hint

   When you call the function `analogWrite`, it will keep working at all times until changed even during a delay.

   iii) Gradually decelerate to 0.

   iv) Repeat the previous steps in the BACKWARD direction. The program should do this continually.

v) Run the program and demonstrate to your tutor.

4 Stepper motor (1 mark)

Task: To control the position of a stepper motor.

Stepper motors are DC motors that move in discrete steps controlled by energizing its coils in the correct order; these steps are dependant on how many steps there are in a full revolution. If a stepper motor has 200 steps per revolution 1 step is $360°/200 = 1.8°$. By manually energising the coils the position of the stepper motor is known, and with some assumptions the position of a vehicle is known too.

1) Build the circuit shown in Figure 8 below. Connect the pins **IN1, IN2,IN3, and IN4** to any four digital pins of the Arduino **(2-13)**. and connect the pins **ENA** and **ENB** to the **5V** pin of the Arduino. Ensure the **GND** pin of the motor driver is connected to the **GND** pin of the Arduino. The red wire from the stepper motor should be connected to the positive power supply wire.

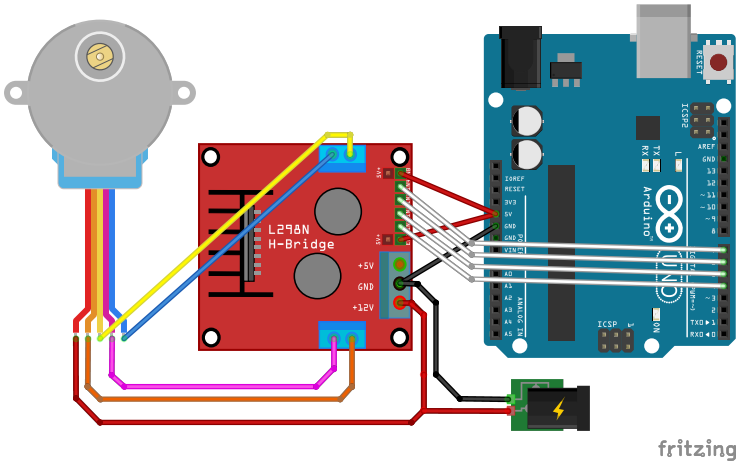

Figure 8: How to connect the stepper motor to the motor driver and Arduino: Connect the **pink** and **orange** wires to **OUT1** and **OUT2**, connect the **blue** and **yellow** wires to **OUT3** and **OUT4**, connect the red wire to power using the attached jumper wire.

2) Create a new project and define the pins for **IN1, IN2, IN3**, and **IN4** that connect to your Arduino.

3) In the setup function enable the pins IN1, IN2, IN3, and IN4 as outputs.

4) Include the built in stepper library by selecting, **Sketch** → **Include Library** → **Stepper**. Refer to the online reference for how to use this library. (https://www.Arduino.cc/en/Reference/Stepper), you will need the following functions:

   - Stepper: https://www.Arduino.cc/en/Reference/StepperConstructor
     Example:
     ```
     Stepper myStepper(100,4,5,6,7);//initialises a stepper with a 4 wire interface
      //... with 100 steps per revolution
     ```

- setSpeed: https://www.Arduino.cc/en/Reference/StepperSetSpeed
  Example:
```
myStepper.setSpeed(20);//sets the speed to 20 rpm, note that the steppers
 //...used in this lab can not achieve high speeds, try 30 rpm as a max speed
```

- step: https://www.Arduino.cc/en/Reference/StepperStep
  Example:
```
myStepper.step(100);//rotates the stepper 100 steps.
```

5) Using the information from the technical specifications (https://www.adafruit.com/product/858), write a program to rotate the motor 1 full revolution forwards and then backwards and repeat.(Look at the examples found in **File→Examples→Stepper→ ....)**

6) Run the program and demonstrate to your tutor.

Tip

As it turns out the stepper motors we purchased for these labs are not to the data sheet's specifications. Using the information in the data sheet and your results from this lab exercise, determine how many steps are actually required for a full revolution.

5 Recommendations

- **Important:** It is highly recommended that you attempt next week's lab exercises, Lab 3, before the lab. To do this at home or on any computer, consider using an online Arduino simulator such as Autodesk Circuits, https://circuits.io/, which provides a graphical environment for you to test your code on a virtual Arduino and includes features such as a virtual oscilloscope.

- Think about the advantages of the different actuators used in this lab and why they could be useful for your Warman design.

- Although we energized the stepper motor coils manually in the lab; if considering stepper motors there are better options. For example the A3967 "Easy Driver" and the A4988/DRV8825 drivers require just a direction and a step pin.

Warning

If you connect L298N driver we used in the labs to an unregulated voltage source for use with steppers, you can expect to damage the source (battery) and the motors.

Use one the drivers below for steppers.

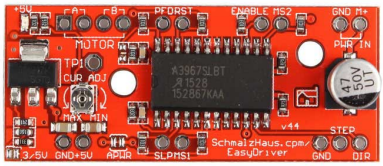

Figure 9: A3967 Easy Driver

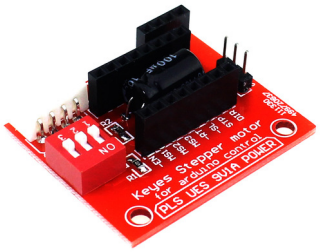

Figure 10: Breakout shield for →

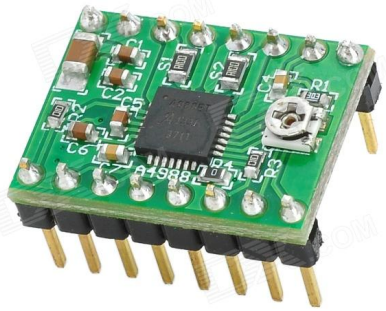

Figure 11: A4988 stepper driver

- To understand more how a stepper works, see https://upload.wikimedia.org/wikipedia/commons/6/67/StepperMotor.gif.
- Another suitable option for running stepper motors is a CNC shield used for 3 axis CNC routers, or RAMPS shield used on an Arduino mega for reprap 3D printers.
- For controlling higher voltage on/off devices, perhaps a small motor, solenoid or fan. A 5 volt relay module is quite useful.
  E.g. (https://www.jaycar.com.au/arduino-compatible-5v-relay-board/p/XC4419)
- If you require a solenoid for your design, it would be worth considering whether it is the best actuator to use, or if a servo doing some linear work would be suitable. But if a solenoid is what you need, make sure you connect it properly with a fly back diode, otherwise the equipment you use to activate the solenoid may be destroyed (Remember from ELEC1300/1310 that an inductor produces a large voltage to resist a change in current when disconnected). To see how to properly connect a solenoid to power electronics look online, e.g., http://web.cecs.pdx.edu/~eas199/B/howto/fishtank/wiring/solenoid_wiring.html.
- There are continuous servos available that regulate speed rather than position, and are operated the same as ordinary servos.
- Remember DC motors are high speed devices and need gearboxes! (You can buy DC motors that come with appropriate gearboxes)