DC MOTOR SPEED MEASUREMENT USING AN ENCODER

1.0 Learning Outcomes……1
2.0 Hardware Set Up……1
3.0 Simulink Set Up and Results……..……..4
4.0 Concluding Remarks…….……7

1.0 Learning Outcomes

After completing this Exercise, you will be able to:

1. Assemble a DC motor control circuit using components such as: ATmega328p microcontroller, Maker-Drive motor driver and DC motor with an encoder.
2. Explain how encoder counts can be used to calculate the rotational speed of a DC motor in revolutions per minute (RPM).
3. Configure Simulink to apply a PWM input signal and capture encoder-based speed data from the DC motor.

After completing this Exercise, it is recommended that you refer to the Learning Outcomes.

2.0 Hardware Set-Up

In this exercise, the motor encoder is introduced and configured to estimate the rotational speed of the DC motor in terms of revolutions per minute (RPM).

When setting-up the hardware and using the ACE-Lab (Base + Sense, Actuate), it is advised you place them side-by-side and use the relevant locations for the actuate components, see Figure 1.

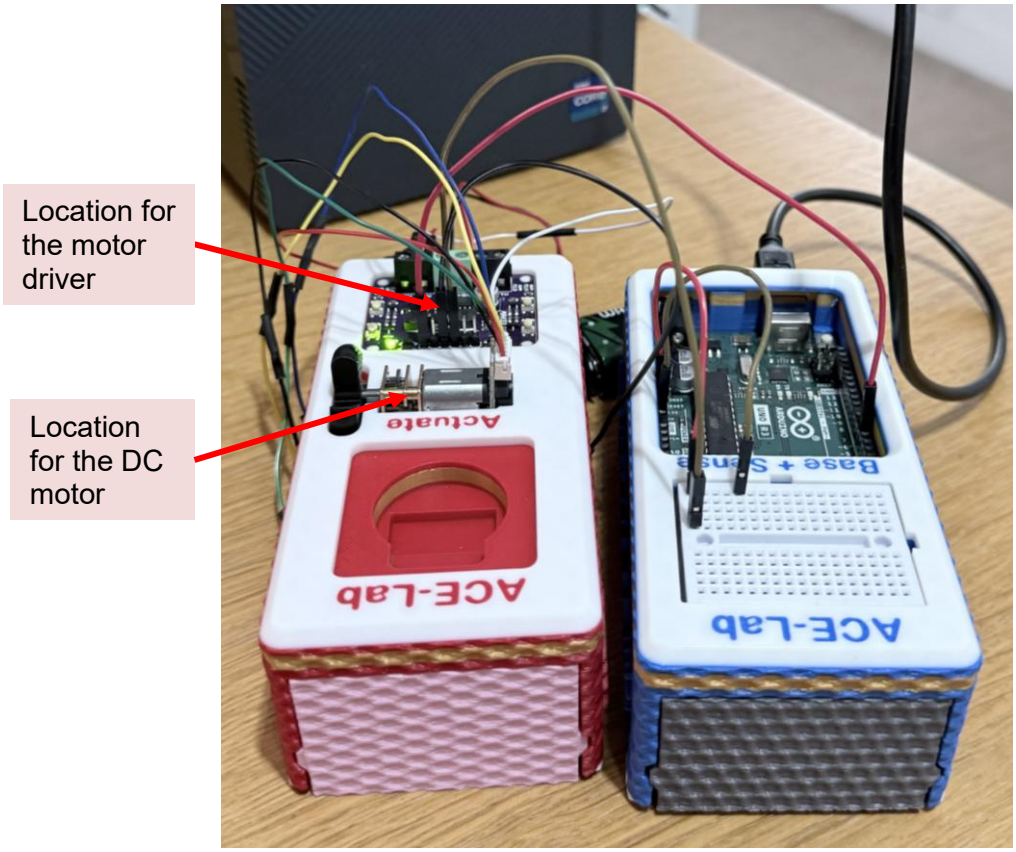


Figure 1: *Use of ACE-Lab (Base + Sense, Actuate) Together, Side-by-Side*

The Arduino provides the control signal to the Cytron Maker Drive motor driver, which regulates the power supplied to the motor. A $9V$ external supply powers the motor driver, while the DC motor itself is rated at $6V$. The motor encoder generates pulses as the motor rotates, which are measured by the Arduino and used to estimate the motor speed. As in the previous exercise, the Arduino and motor driver must share a common ground to ensure that the control signals are correctly referenced and recognised.

Required Hardware

For this exercise, the following hardware is required:

1. ATmega328P microcontroller
2. Breadboard
3. Cytron Maker Drive motor driver board
    a. Supply voltage: 2.5–9.5V DC
    b. Maximum continuous current: 1.0A
4. DC motor with encoder and 1:298 gearbox
    a. Rated voltage: $6V$ DC
    b. Rated current: approximately 0.3A
5. $9V$ battery clip
6. $9V$ battery
7. Breadboard jumper wires
    a. 2 × male-to-male wires (10cm)

   b. 3 × male-to-female wires ($20cm$)
8. USB Type-B cable

Step-by-step set-up instructions

To set up the hardware, use the circuit diagram provided in **Figure 2** as a reference. A combination of **male-to-male (m-m)** jumper wires, supplied with the **ACE-Lab (Base + Sense)**, and **male-to-female (m-f)** jumper wires, supplied with the **ACE-Lab (Actuate)**, will be required.

As shown in Figure 2, the hardware should be connected as follows:

1. Connect the two DC motor wires to the Maker Drive motor output terminals for the selected motor channel.
2. Connect the positive terminal of the 9V battery to the Maker Drive motor power input, labelled VB+.
3. Connect the negative terminal of the 9V battery to the Maker Drive ground terminal, labelled VB−.
4. Connect one of the Arduino GND pins to the Maker Drive GND connection. This creates a common ground between the Arduino and motor driver, ensuring that the control signal has the same voltage reference.
5. Connect the encoder VCC (+5V) connection to the Arduino 5V supply.
6. Connect the encoder GND connection to an Arduino GND connection. This should form part of the same common ground used by the Arduino and motor driver.
7. Connect encoder Channel A to Arduino digital pin 2.
8. Connect encoder Channel B to Arduino digital pin 3. Channels A and B provide the encoder signals required to determine the direction and rotational speed of the motor. These pin assignments correspond with the Simulink encoder configuration used later in the exercise.
9. Connect Arduino Pin 5 to the appropriate Maker Drive control input for the selected motor channel, as shown in Figure 2. Pin 5 provides the PWM control signal used to regulate the voltage applied to the motor.
10. Ensure that the second Maker Drive control input for the selected motor channel is connected or configured as shown in Figure 2, so that the required direction of motor rotation is obtained.
11. Where convenient, use the breadboard power rails to organise and distribute the 5V and GND connections required by the encoder.
12. Finally, connect the Arduino to the computer using the USB Type-B cable.

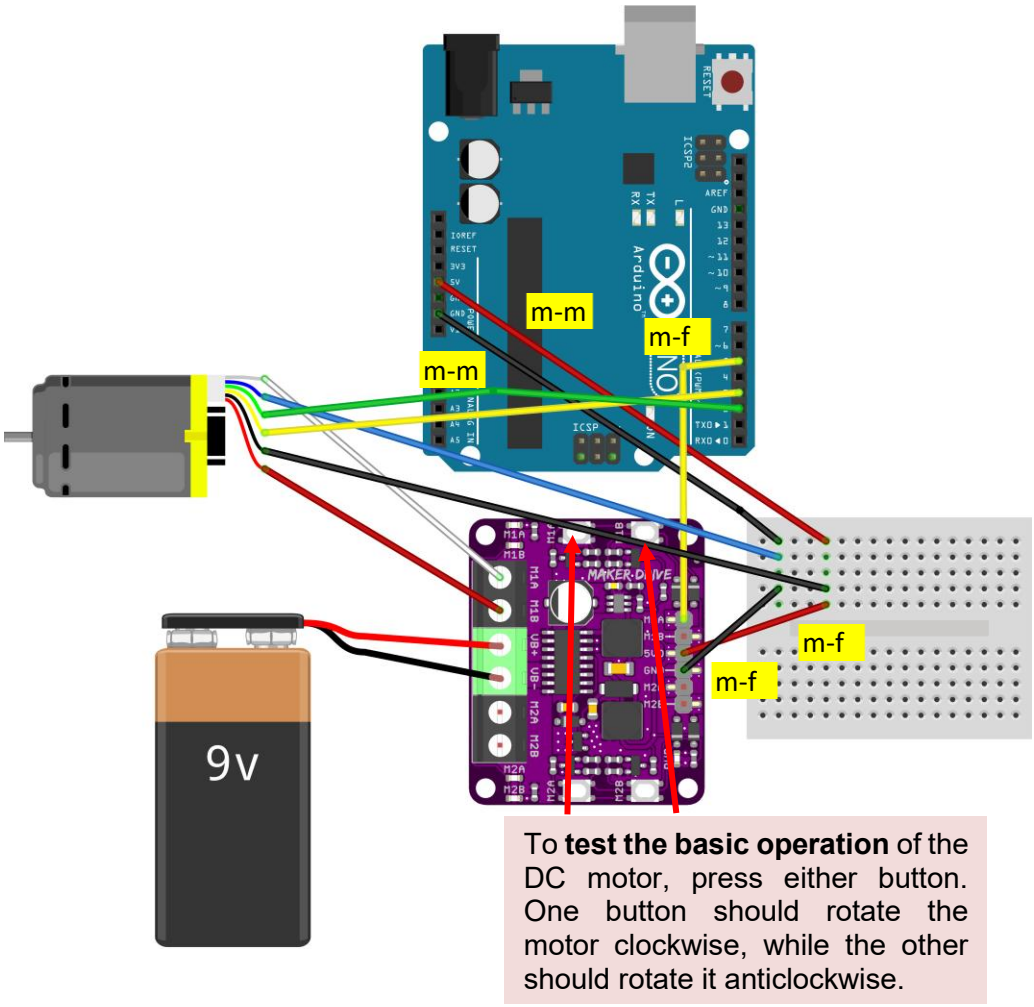


Figure 2: *Hardware Set Up for DC Motor Speed Measurement using Encoder*

3.0 Simulink Set Up and Results

A PWM signal is applied to Arduino pin 5, which provides the voltage input to the DC motor driver, as shown in Figure 3.

The DC motor is rated at $6V$, while the motor driver is supplied by a $9V$ battery. To limit the average voltage applied to the 6V motor to approximately 6V, the maximum PWM duty cycle is limited to:

$$D_{\max} = \frac{6}{9} = 0.667$$

or approximately 66.7%.

The Arduino PWM command ranges from 0 to 255. Therefore, the corresponding maximum PWM command is:

$$255 \times \frac{6}{9} = 170$$

Hence, the command applied to pin 5 should be limited to:

$$0 \leq \text{PWM command} \leq 170$$

For the initial motor test, a PWM command of 170 is applied. This corresponds to a duty cycle of approximately 66.7%, limiting the effective voltage applied to the $6V$ motor when using the $9V$ supply.

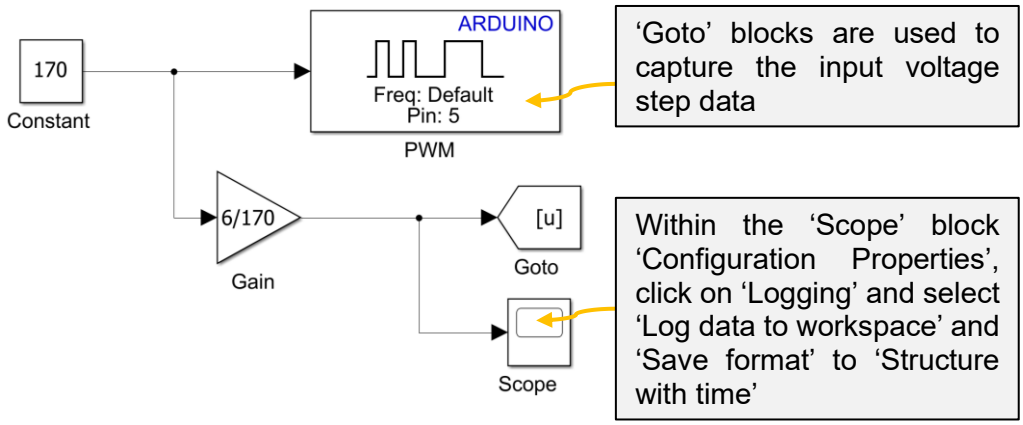


Figure 3: *Input Voltage Applied to the DC Motor with an Encoder*

In Simulink, the encoder on the DC motor is configured to determine the RPM of the motor. This is achieved by using the encoder counts, which indicate the motor's position at any given time. By analysing the change in position over a specific sampling interval, the motor's speed can be calculated. Essentially, the encoder acts as a feedback device, measuring the motor's position in discrete steps. By calculating the rate of change of these steps, the RPM is estimated, providing a crucial metric for controlling the motor's speed in various applications. The details of this setup are illustrated in Figure 4.

Figure 5 presents the complete Simulink diagram used to apply the known input and measure the corresponding system output. The input voltage Simulink diagram from Figure 3 is included because these components need to be 'Run' together. A scope has also been added to display the counts per second from the encoder, with the scope data provided in Figure 5. The 'Goto' blocks are used in Simulink to capture both the input (voltage) and output (RPM) signals. These signals are then sent to a scope using 'From' blocks, allowing the input-output data to be viewed on the scope.

Note: A negative encoder speed indicates that the motor is rotating in the opposite direction to the defined positive direction. If clockwise rotation is intended to be positive, either reverse the motor direction or interchange encoder channels A and B, depending on the required convention.

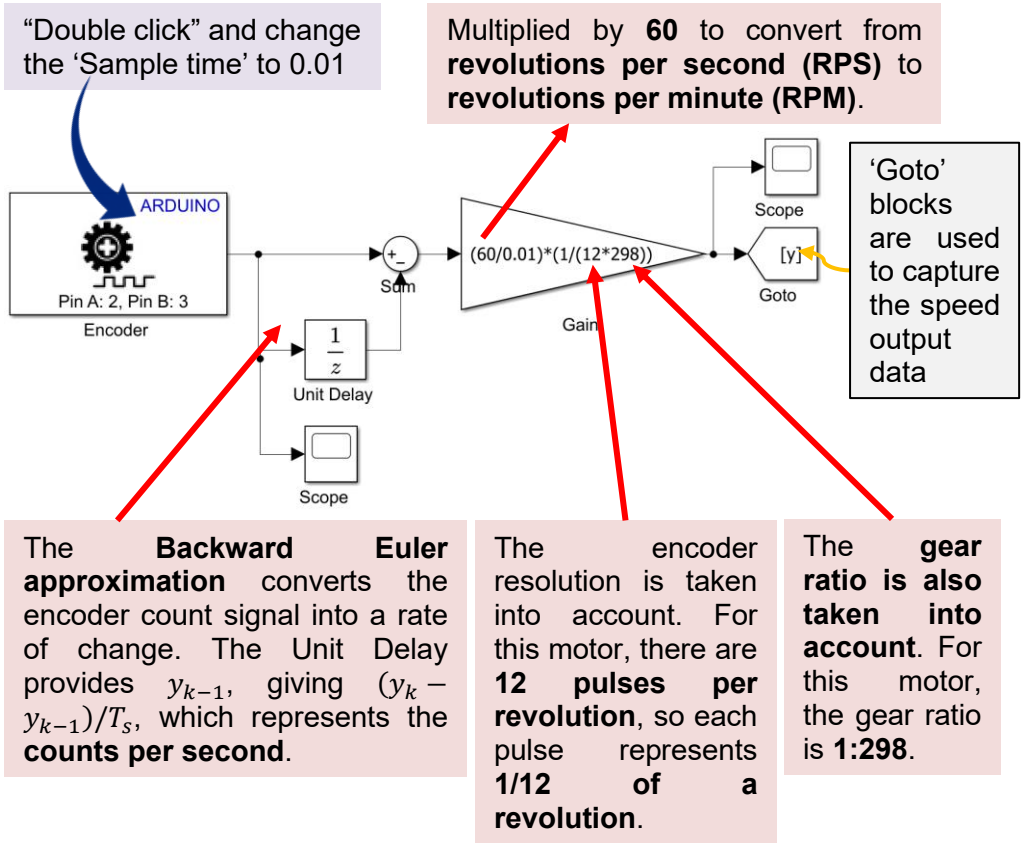
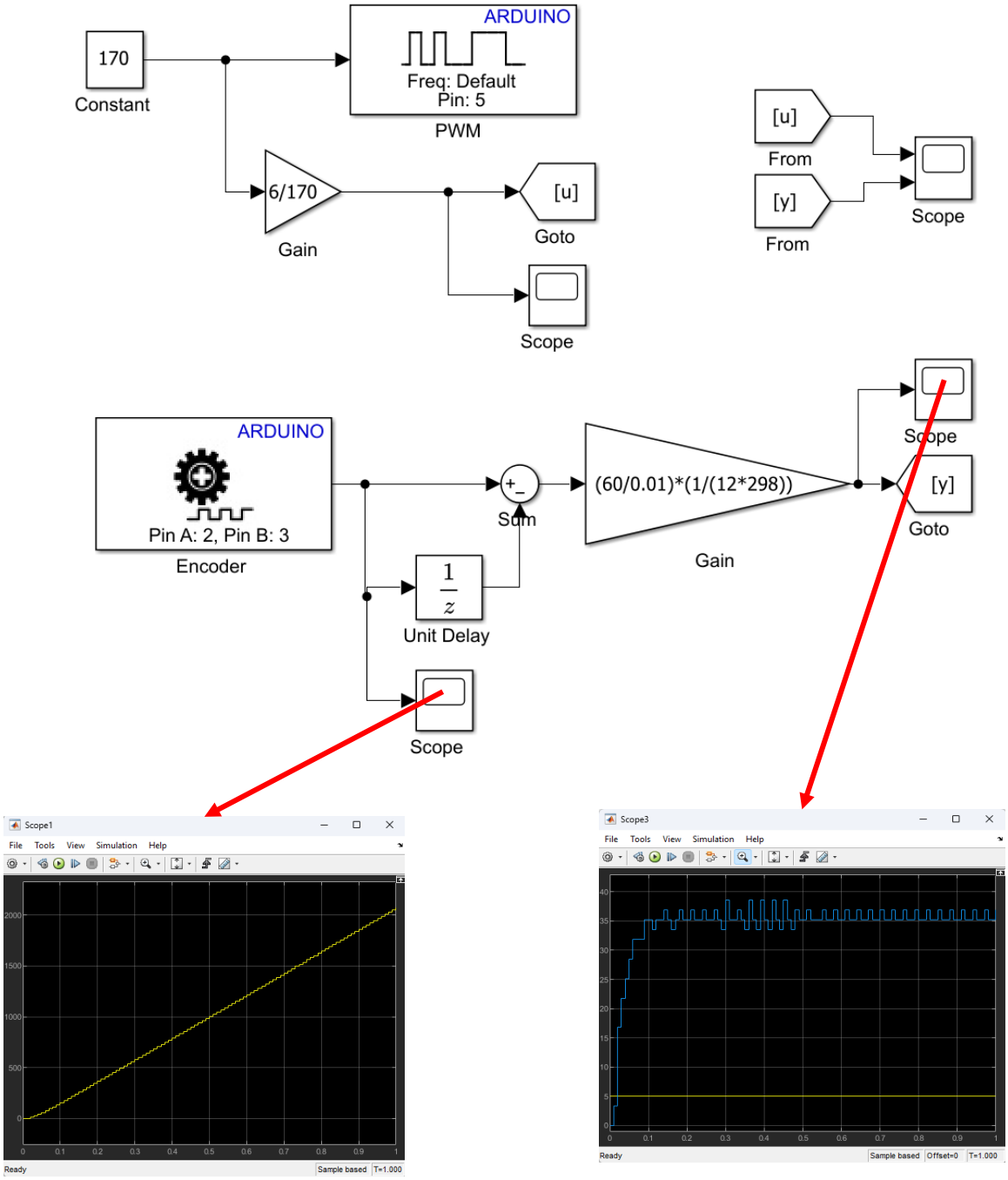


Figure 4: *Encoder Data Capture and Configuration using Simulink*



The data observed from the scope illustrates counts/second from the DC motor encoder, with this increasing when the DC motor is rotating in the clockwise direction

From observation of the RPM response, the motor dynamics appear to exhibit approximately first-order behaviour and can therefore be represented initially using a first-order transfer-function model.

Figure 5: *Encoder Data Capture and Configuration using Simulink* (With One Gain Block used for the Main Components)

4.0 Concluding Remarks

This exercise introduced the use of encoder feedback to measure the speed of a DC motor controlled by an Arduino Uno and Maker Drive motor driver. By applying a PWM input and processing the encoder counts in Simulink, the motor speed can be estimated in RPM and displayed for analysis. The measured input-output data obtained here can subsequently be used to identify a mathematical model of the motor in the following exercise.

Recommended citation:

Pickering, J. E. (2026). *DC Motor Speed Measurement Using an Encoder*. ACE-Lab Teaching Notes: ACE-Lab Practical Exercises, Version 1.0. Zenodo. https://zenodo.org/records/23019221