LDR (Light Dependent Resistor) Exercise

1.0 Learning Outcomes

After completing this Exercise, you will be able to:

1. Understand how to map the analogue input reading from the light dependent resistor (LDR) to a suitable range and use it to control LED brightness via pulse width modulation (PWM).
2. Understand how to use a MATLAB Function block to implement control logic that switches the LED on or off based on measured LDR values.

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

2.0 Hardware Set Up

The exercise involves connecting an LED to a pulse width modulation (PWM) Arduino output pin and controlling the brightness via the readings from the LDR sensor, see Figure 1.

Note that while the ACE-Box can be used for all the exercises, it is not required and only the individual components are needed.

Required hardware for this exercise:

- Arduino Uno board (supported by Simulink)
- USB Cable Type A to B
- Breadboard
- LED
- 2 x 220Ohm resistor
- LDR sensor
- 5 x male-male breadboard wires

Set-up the hardware as shown and following these steps:

1. Place the LED on the breadboard and add a 220 Ohm resistor to its anode.
2. Connect the LED's cathode to GND, and the free end of the resistor to Pin 9.
3. Place the LDR on the breadboard and connect its other leg to 5V.
4. Add a 220Ω resistor from the second LDR leg to GND to form a voltage divider.
5. Connect the junction between the LDR and resistor to A0 on the Arduino.

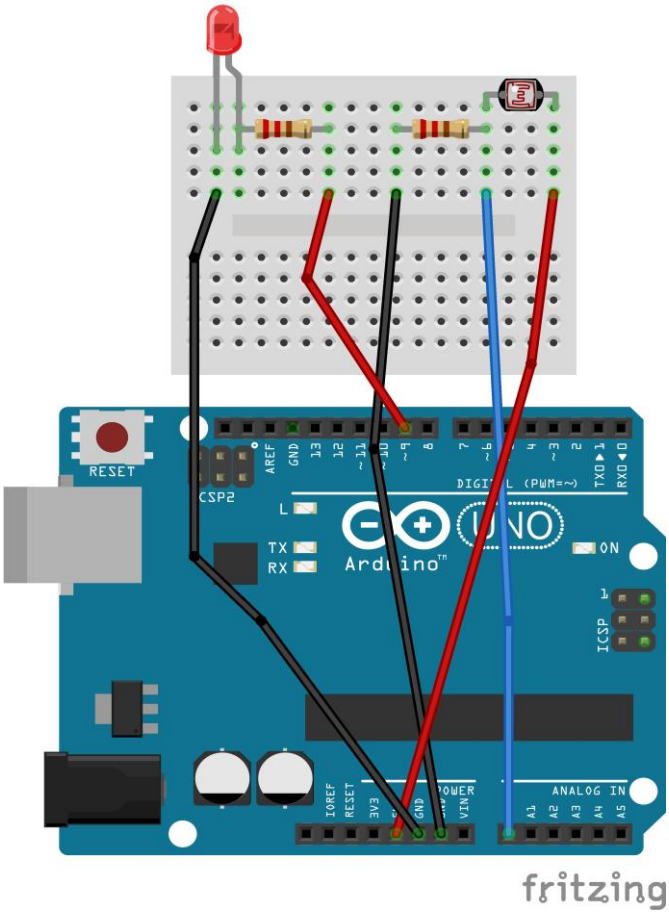

Figure 1: *Hardware Set-Up for LDR (Light Dependent Resistor) Exercise*

3.0 Simulink Set Up and LDR Testing

In this part of this initial exercise, you will develop a Simulink model to control the brightness of an LED using PWM, based on the analogue reading from the light dependant resistor (LDR), i.e., as the analogue reading of the LDR increases, so will the brightness of the LED, see Figure 2.

The steps to set up the model are as follows:

1. Add an Analog Input block and changing the pin number of A0 and sample time to 0.01. This reads the sensor voltage and outputs a value between 0 and 1023.
2. Insert a Gain block with value 255/1023. The gain block scales the ADC reading into the 0–255 range needed for PWM duty cycle.
3. Connect the scaled signal to a PWM Output block and change the Pin to 9. The PWM block uses the 0–255 value to generate a proportional PWM signal on the Arduino.
4. Connect two display blocks to the analogue output and after the gain block. This will enable the raw sensor reading to be viewed in real-time as the model runs, and also show the scaled PWM value, such that you can verify the conversion.
5. Deploy the model to the Arduino. Once uploaded, adjust the sensor levels of light to see the PWM output automatically change, and the brightness of the LED.
6. Be aware that the LDR-to-LED mapping might be operating in the wrong direction. A lower LDR value, corresponding to darker conditions currently produces a lower LED brightness, whereas typically, it would be expected that the LED to brighten as the environment darkens. The next task within this exercise addresses this.

Recall how to perform code-generation by visiting HERE.

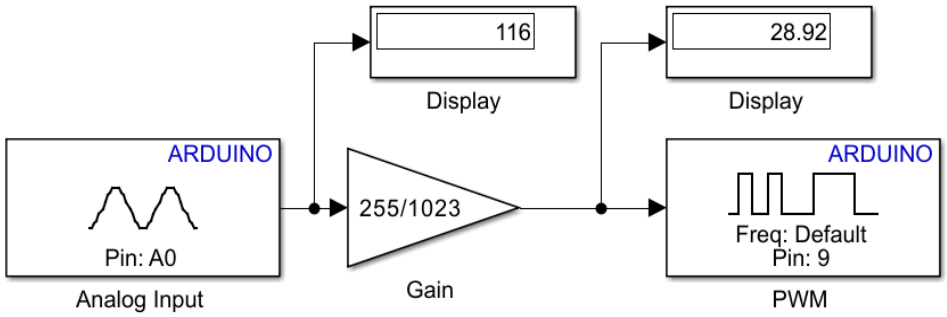

Figure 2: *Simulink Set-Up for LDR (Light Dependent Resistor) Exercise*

⚠ If you encounter any errors, click the link HERE for troubleshooting help.

4.0 MATLAB Function and LDR Testing

The previous example is now updated to include a MATLAB Function block after the gain. In this exercise, instead of sending the scaled value straight to the PWM

output, we now run it through a small piece of code that decides whether the output should be on or off.

The steps to set up the model are as follows:

1. Add a MATLAB Function block between the gain and PWM blocks. This will let us apply some simple decision-making to the signal before it reaches the PWM output.
2. Open the MATLAB Function block and replace the default code with the following:

```
function y = fcn(u)

if u <= cast(255/2,'like',u)
    y = u;
else
    y = cast(0,'like',u);
end
```

3. Deploy the model. Now the PWM output will only turn on when the input signal is above half of its range, creating a simple on/off threshold behaviour.

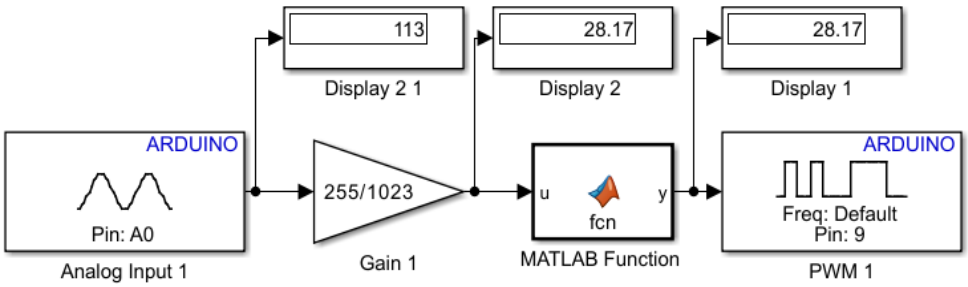

Figure 3: *MATLAB Function Set-Up for LDR (Light Dependent Resistor) Exercise*

Understand the MATLAB Function block

- u is the input from the Gain block (0 – 255).
- 255/2 is the midpoint (127.5).
- If the input is greater than or equal to this midpoint, the value passes through.
- If the input is below the midpoint, the output is forced to zero.