LED (Digital) Exercise

1.0 Learning Outcomes

After completing this exercise, you will be able to:

1. Understand the use of the **Simulink Arduino support package** for algorithm design applied to a simple digital circuit.
2. Understand the function of the **Arduino Uno digital Input/Output (I/O) pins** through the implementation of an LED ON/OFF control task.

After completing this exercise, you are encouraged to review the learning outcomes and confirm that they have been met.

2.0 Requirements

The exercise has the following primary requirements:

- The Simulink model shall generate a digital output signal that turns an LED ON and OFF at a fixed, user-defined time interval of $0.5s$ (on) and $0.5s$ (off).
- The Simulink model shall use a sample-based pulse signal with a sample time of $0.1s$ to control an Arduino digital output pin.
- The Simulink model shall drive Arduino digital pin 9, outputting a logical LOW (0) to turn the LED OFF and a logical HIGH (255) to turn the LED ON.
- The Simulink model shall be deployable to an Arduino Uno using Simulink code generation and shall execute continuously in real-time (simulation stop time set to infinity).

3.0 Hardware Set-Up

The exercise involves connecting an LED to a digital Arduino output pin and controlling its state by switching it ON and OFF, as shown in Figure 1.

Note: While the ACE-Box can be used for this exercise, it is not required here. Only the individual components listed below are needed.

Required hardware for this exercise:

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

Hardware Assembly Steps

1. Connect **digital pin 9** on the Arduino to a chosen column on the breadboard using a male-to-male jumper wire.
2. Insert a **220 Ω resistor** with one end in the same column as the wire from pin 9 and the other end in a different row.
3. Insert the LED such that:
    - The **long leg (anode)** is connected to the free end of the resistor.
    - The **short leg (cathode)** is connected to an Arduino **GND** pin using a jumper wire.

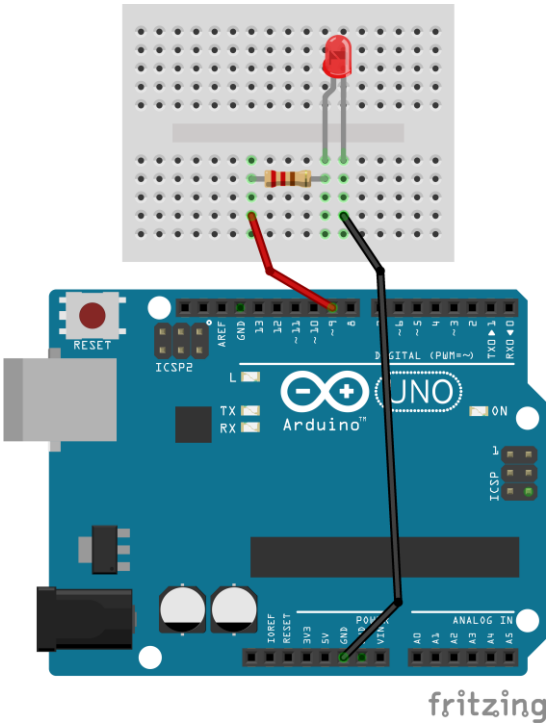

Figure 1: *Hardware Set-Up for LED (Digital Exercise)*

4.0 Simulink Set-Up

In this part of the exercise, you will develop a Simulink model to control the Arduino digital output pin and turn the LED ON and OFF. A **Pulse Generator** block is used to alternate the output between **0 (OFF)** and **255 (ON)**, as shown in **Figure 2**.

Simulink Model Construction Steps

1. From **Simulink** → **Sources**, add a **Pulse Generator** block to the model.
2. From **Simulink Support Package for Arduino Hardware** → **Common**, add a **Digital Output** block.
3. Connect the output of the **Pulse Generator** block to the input of the **Digital Output** block.
4. Open the **Pulse Generator** block parameters and configure: **Pulse Type**: *Sample based* and **Sample time**: 0.1 seconds.
5. Open the **Digital Output** block parameters and set: **Pin number**: 9.
6. From **Simulink** → **Sinks**, add a **Scope** block.
7. Branch the signal from the **Pulse Generator** output and connect it to the **Scope** input to monitor the generated waveform.

Key Block Parameters to Modify

- Pulse type → **Sample-based**
- Sample time → **0.1 s**

All other parameters may remain at their default values.

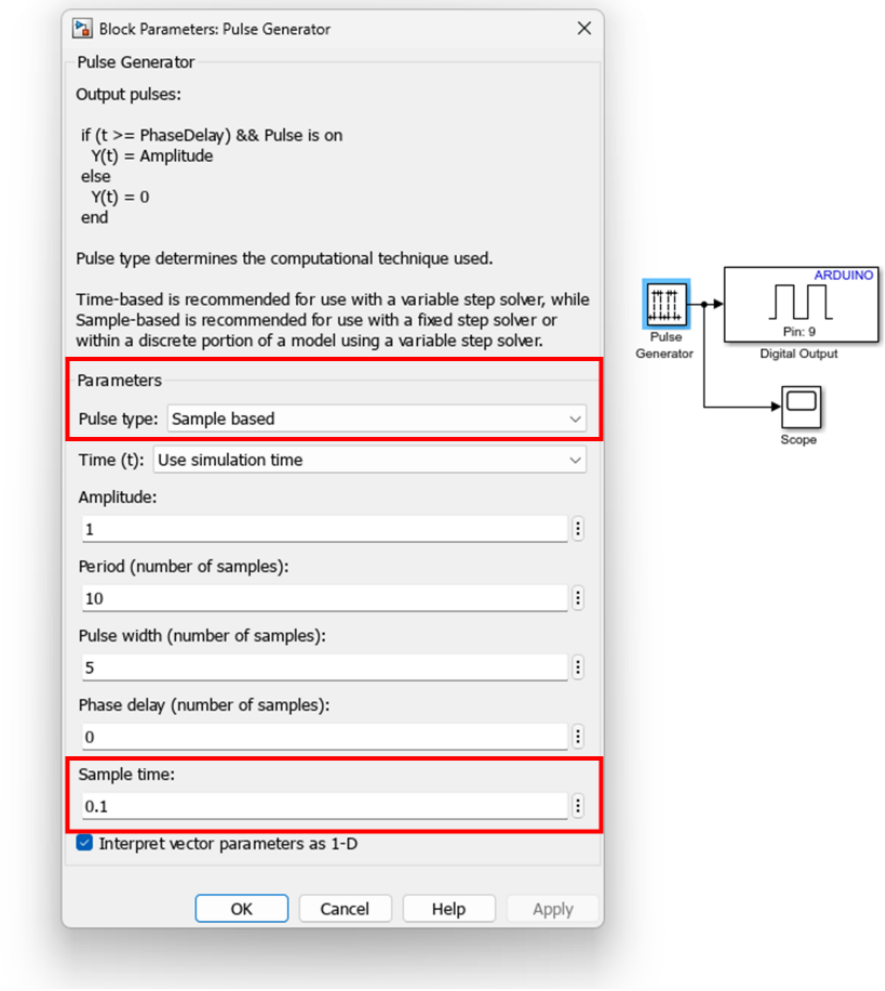

Figure 2: *Simulink Set-Up for LED (Digital Exercise)*

5.0 Running Simulink Code Generation

In this part of the exercise, you will configure the Simulink model for code generation and deploy it to the Arduino Uno, as shown in **Figure 3**.

1. Connect the Arduino Uno to your computer using a USB cable.
2. In Simulink, open **Model Settings** by selecting *Modelling → Model Settings*.
3. Select **Hardware Implementation** from the left-hand menu and choose **Arduino Uno** as the hardware board. Click **Apply** and **OK**.
4. Confirm that the **Hardware** tab appears in Simulink.
5. Ensure MATLAB is operating in a suitable **working directory** before running the model.
6. Set the simulation stop time to **inf (infinity)** and click **Monitor & Tune** to deploy the model.

The Simulink model will now be compiled into C-code and executed on the Arduino hardware.

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

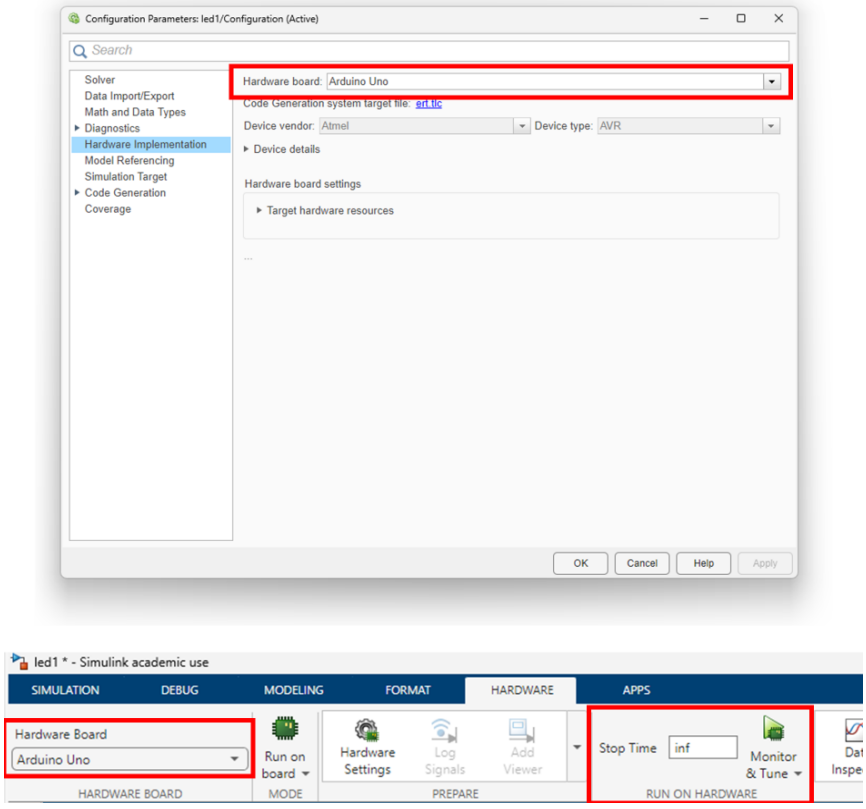

Figure 3: *Running Simulink Code Generation*

6.0 Exercises

1. **Modify Timing Requirements**: Rewrite the timing-related requirements defined in Section 2.0 (i.e., the LED ON/OFF period and the sample time) using new, user-defined values. Update the Simulink model accordingly and verify that the deployed system operates with the revised timing parameters.
2. **Out-of-Sequence LED Control**: Extend the system by adding a second LED connected to a different Arduino digital output pin.
    - First, write a new set of requirements specifying out-of-sequence (phase-shifted) LED behaviour.
    - Then, implement and deploy the Simulink model to demonstrate asynchronous operation of the two LEDs.

7.0 Concluding Remarks

This exercise has demonstrated the complete workflow for implementing a simple digital control task using Simulink and an Arduino Uno, from defining requirements through to real-time execution on hardware. By developing, configuring, and deploying a Simulink model to control an LED, you have gained practical experience in linking model-based design with embedded hardware implementation.

The exercise reinforces key concepts including digital input/output (I/O) operation, timing through sample-based signals, and automatic C-code generation using the Simulink Arduino support package. Importantly, it illustrates how abstract control logic developed in Simulink can be translated directly into executable behaviour on a physical system without manual programming.

This LED (Digital) Exercise serves as a foundational building block for more advanced tasks involving sensors, actuators, and closed-loop control. The same principles applied here: clear requirements, correct hardware interfacing, and disciplined model configuration. This will extend naturally to more complex embedded and control engineering applications introduced in subsequent exercises.