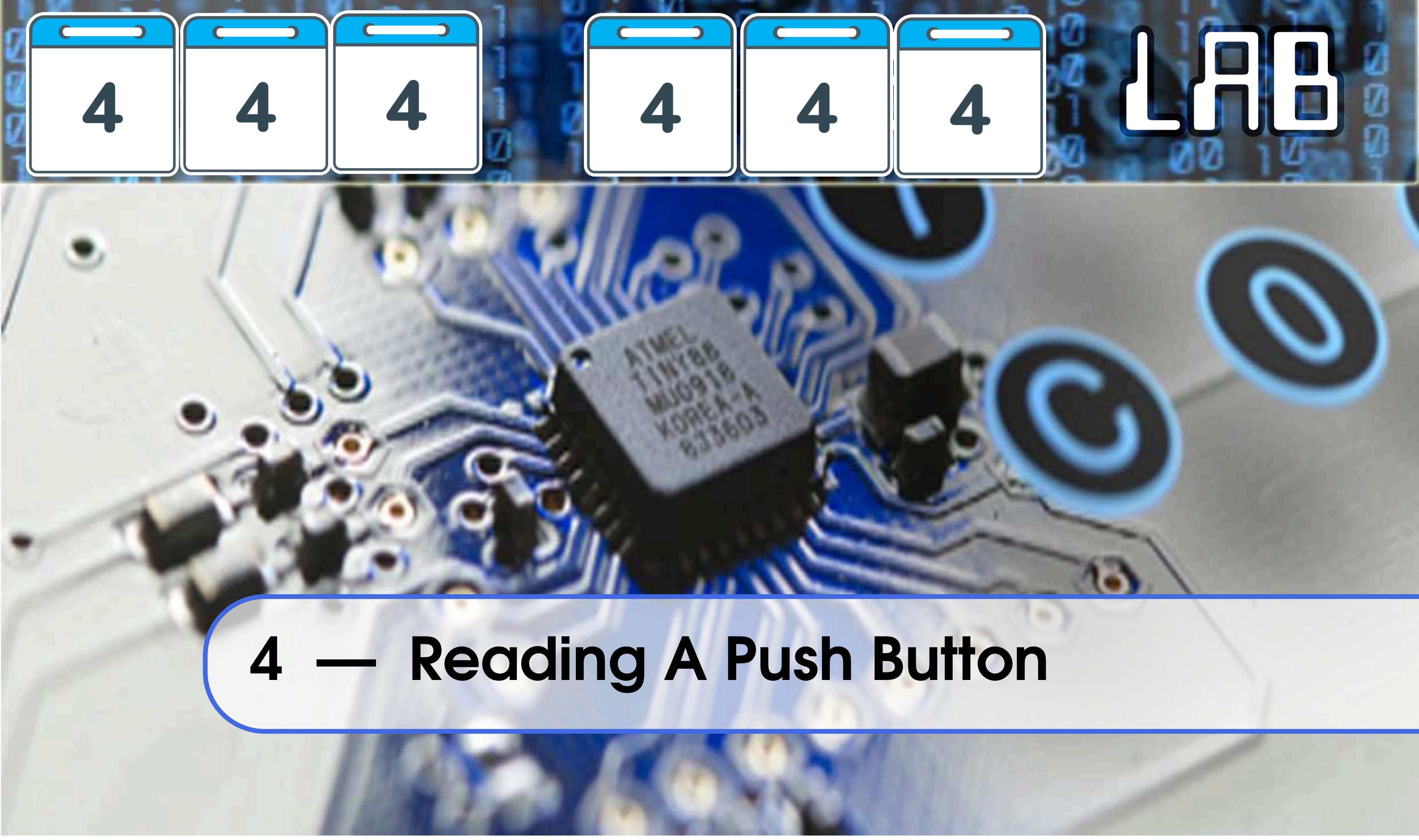

4 — Reading A Push Button

LAB Objectives

The aim of this LAB experiment is to teach students how to interface and read simple digital input devices such as push button. This experiment also shows how to use software debouncing.

4.1 Hardware Required

1. Arduino Uno Board
2. Breadboard
3. Push Button
4. Resistor

4.2 Circuit

The circuit to be implemented in this experiment is a simple interface of a push button to pin 4 of the Arduino board. The schematic of the circuit is shown in Figure 4.1. The implementation of the push button interface using Virtual Breadboard software is show in Figure 4.2.

To build this circuit follow the steps discussed in Experiments 1 and 2 to place and connect the following components :

1. the *VBBExpress* breadboard component,
2. the *Arduino Uno Board*,

3. a *Resistor*, and
4. an *Push Button*.

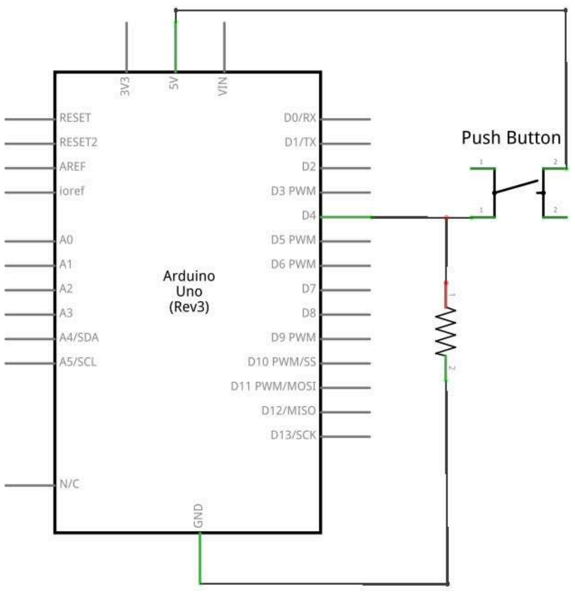

Figure 4.1 – Schematic of the push button circuit.

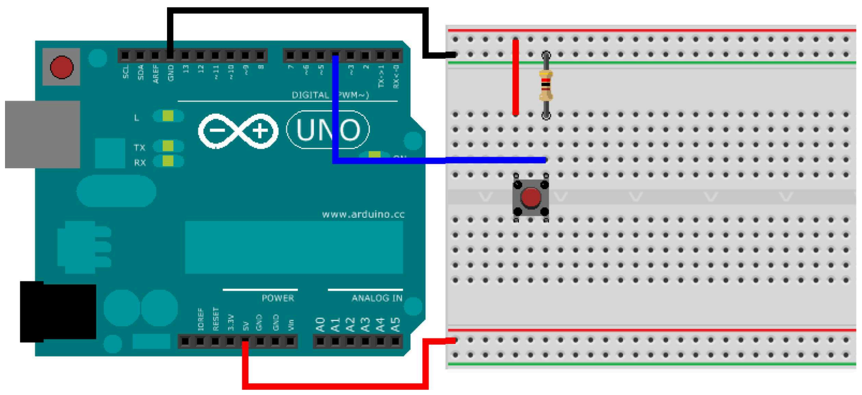

Figure 4.2 – Push button circuit.

4.3 Program

In this part, you will write a program to read the state of a button connected to pin 4, and then display the message "Push Button Pressed" on the Arduino environment's built-in serial monitor each time the button is pressed.

Writing The Code :

To accomplish this, follow the steps described in Experiment 2 to write the code shown in Program 4.1.

Discussing The Code :

In this program, the first thing you do is to declare a variable (*buttonState*) to hold the state of the button. Next you initialize the pin connected to the button (pin 4) as an input pin with the command : pinMode(4, INPUT). In the loop function, you read the state of the button and assign it to the *buttonState* variable with the command : buttonState = digitalRead(4). Then, you check if the read state (the value of the *buttonState* variable) is HIGH. If it is so, you display the message "Push Button Pressed" on the Arduino environment's built-in serial monitor with the command : Serial.println("Push Button Pressed").

Program 4.1 Reading the state of a push button.

```
//Repeatedly reads the state of a push button connected on pin 4.
//If the button is pressed, it displays the message "Push Button Pressed".
import muvium.compatibility.arduino.*;
public class BlinkingLED extends Arduino{
   int buttonState;  // variable for reading the pushbutton status

   void setup() {
       // initialize the push button pin (pin 4) as an input:
       pinMode(4, INPUT);
       // initialize and start the serial port:
       Serial.begin(9600);
   }
   void loop(){
       // read the state of the push button value:
       buttonState = digitalRead(4);

       // check if the pushbutton is pressed, then display the message:
       if (buttonState == HIGH) {
           Serial.println("Push Button Pressed");
       }
   }
}
```

4.4 Circuit Emulation

To validate your design you need first to build the source code and then run the emulator. On the *Debug Toolbar*, locate and click the *Build* button to build your code. If there are no errors in the compilation process, emulate the your program by clicking the run button located on the *Application Toolbar*. Observe the display of the message "Push Button Pressed" each time you press the button. You will notice that the message is displayed more than once although the button is pressed only once. This is due to the switch bounce phenomena.

Exercise 4.1 Modify Program 4.1 to solve the switch bounce problem using software debounce method discussed in class. Emulate your program to verify the debounce method. ■