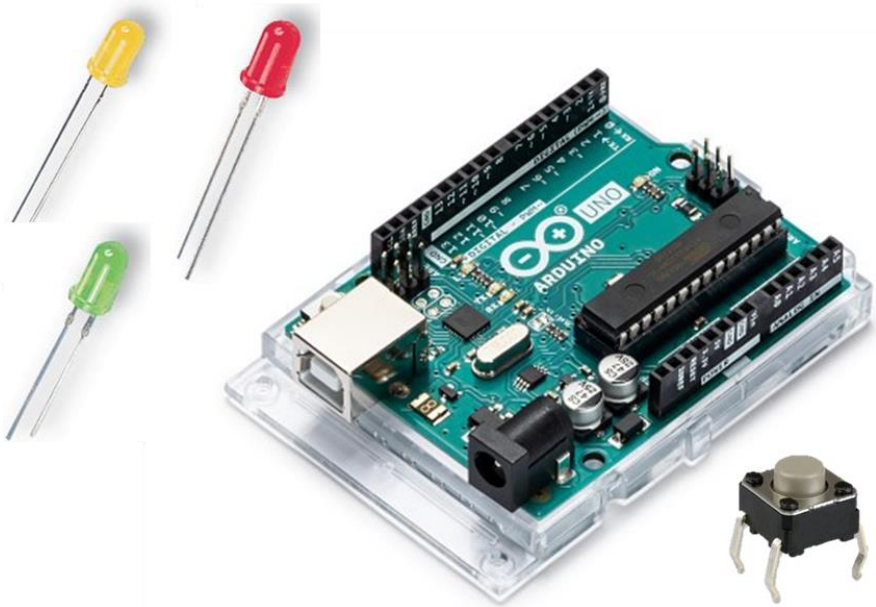
Introduction to Arduino – Digital I/O

Ben Taylor
Shuhei Miyashita and Dana Damian



Learning Outcomes:

Practical	Upload a simple program from the Arduino IDE to an Arduino module.
Produce	Build simple circuits to interface switches and LEDs on an Arduino device.
Practical	Implement code on an Arduino, to interface with simple digital I/O devices, such as switches and LEDs.

Contact: B.P.Taylor@Sheffield.ac.uk Pre: -%, During -%, Post 100%

Introduction to Arduino – Digital I/O: In-Laboratory Activities

1 Aims and Objectives

The aims of this laboratory session are to familiarize yourself with the Arduino Development Environment, IDE, and to gain experience of uploading user programs to the embedded target. During the laboratory session you will connect the Arduino device to the button switches and LEDs, to provide the microcontroller with controllable digital inputs and measurable outputs. You will then write some simple programs to measure and manipulate the input signals, and display digital outputs.

2 Names of collaborators for the experiment

Use this space to record the member of the group you are conducting the experiment with.

Name of Lab partner(s)	Lab Desk Number

3 The Arduino IDE

This section is aimed at providing you with a brief overview into installing and setting up the Arduino IDE, ready for use with these laboratory activities.

3.1 Installation and Setup of the Arduino Software

NOTE: The procedures laid out in this section are only for use on your home computer. The PCs you will be using in the University laboratories and computer rooms already have the Arduino software installed, or it must be installed from the Managed Desktop software Centre.

Also Note: All the documentation for this module assumes you are using a Windows 10 PC. We do not support other operating system versions.

Information about installing the Arduino IDE can be found on the Arduino website at: https://www.arduino.cc/en/Guide/Windows

Procedure:

1. Navigate to the Arduino IDE download page: https://www.arduino.cc/en/Main/Software and download the appropriate version of the Arduino IDE for your operating system.
2. Launch the Arduino IDE, to check it has been installed correctly
3. Ensure you have the latest version of the hardware drivers for the Arduino board:
    - From within the Arduino IDE, select the Tools Menu and navigate to the Board Manager: **Tools** > **Board:** *[name is the current board]* > Boards Manager…
    - From the Boards Manager dialogue box, shown in Figure 1, ensure the latest version of the **Arduino AVR Boards** driver is installed. You can check if there

is a more up-to-date version by clicking on the dropdown box in the **Arduino AVR Boards** section, and seeing if there is a higher version available than the installed version indicated at the top of the section – do not install a lower version.

Figure 1. Screen shot of the Boards Manager Dialogue box

4. At this point you should connect the Arduino Uno to your system and direct the Arduino IDE to the correct Arduino board type and Port to allow your Arduino device to be programmed.
    - It you do not know how to do this, please look at the Getting Started with Arduino UNO guide on the Arduino website: https://www.arduino.cc/en/Guide/ArduinoUno

4 Laboratory Exercises

The Arduino microcontroller is a powerful tool to build a wealth of mechatronics and robotic projects. We're just getting started - so let's introduce how to work with push buttons and LEDs. These elements are digital inputs and outputs, and are representative of the type of actions we typically want in mechatronics projects: we want that upon a command (e.g., pressing a button), an action to take place that can modify the environment in a sensible way (e.g., a noticeable light on or off).

Before starting the laboratory exercises, you should review the contents of the Mechatronics KIT Information folder, located in the Practicals content area of the Blackboard site for the Mechatronics course.

4.1 Exercise 1: Blink Example

The aim of this exercise is to demonstrate how to upload and run a simple Arduino program.

The Blink example is a good way to test the installation of your Arduino software, and also to identify if there is a major problem with your Arduino hardware.

Procedure:

1. Load the Blink example from the Arduino IDE files menu: Files > Examples > 01.Basics > Blink

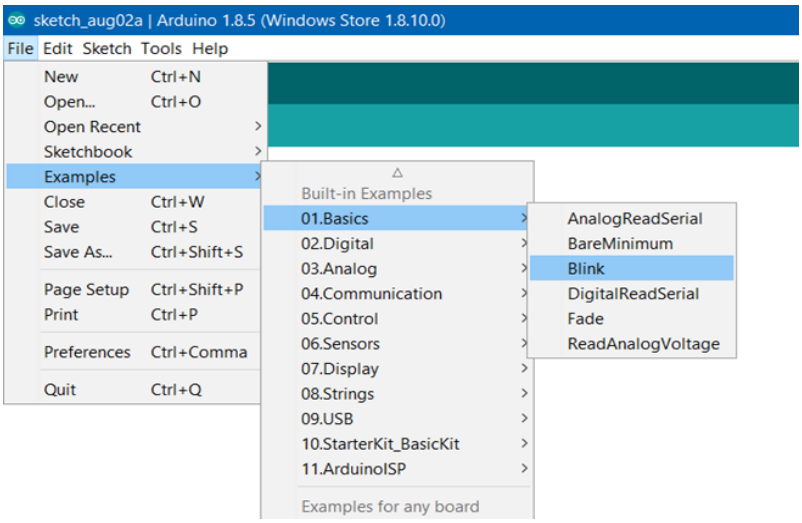


Figure 2. Screen shot of how to find the blink example in the Adruino IDE

2. You compile and upload your Arduino program, or Sketch, by pressing the Press Upload button, as highlighted with a red circle in Figure 3:

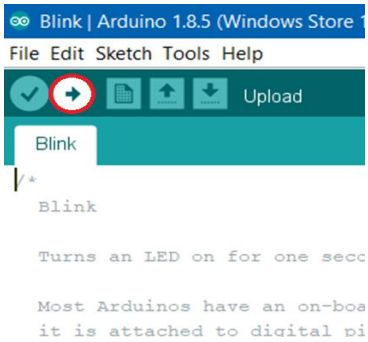


Figure 3. A screen shot of the Upload button, highlighted with a red circuit, in the Arduino IDE.

After the IDE compiles and uploads your program, the Arduino should start to run your code. For this example, this will be indicated by the LED, “L”, (highlighted in Red in Figure 4), which should be slowly flashing. (Note: LED “L” is also connected to pin 13).

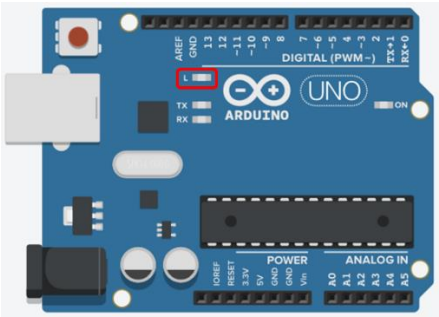


Figure 4. The Arduino Board, with the LED, “L”, highlighted with a red box.

3. The setup() and loop() functions of the Blink example are shown below. Modify the

values in the delay() functions, (highlighted in red) and investigate its effects again. What happens?

(See the Arduino Language Reference, from the Arduino website, for details of the delay() function: https://www.arduino.cc/reference/en/language/functions/time/delay/)

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

You should observe that changing the upper delay() function changes the on-time of the LED, and changing the lower delay function of the changes the off-time for the LED.

4.2 Exercise 2: Push Button and LED

The aim of this exercise is to use a digital input, in this case from a push button, to trigger the Arduino to produce a digital output. In this case and LED, as illustrated in the circuit shown in Figure 5.

Wiring Procedure:

1. Using the Kit of parts provided, connect the circuit as shown in Figure 5.
    - It is not essential to maintain the colour of wires illustrated in the diagram.
    - The digital input from the button is connected to pin 2
    - The digital output to the LED is connected to pin 13, (shared with the LED “L” on the Arduino board)
    - The resistor connected between the button and the GND line is 10K, and the resistor connected between the anode of the LED and pin 13 of the Arduino is 470Ω.

(There is a guide to reading resistor values in the Mechatronics Kit Information section of the Blackboard site.)

Note: The digital Input, on pin 2, is ‘pulled’ down to the 0V supply rail by the a 10K resistor, (a resistor used in this manner is often referred to as a pull-down resistor). In this configuration, when we press the button, the digital input is connected to the +5V supply rail. When the button is released, the 10K resistor ‘pulls’ the digital input back down to 0V. without the 10K resistor, the input would ‘float’, and may produce a spurious input when the button is released.

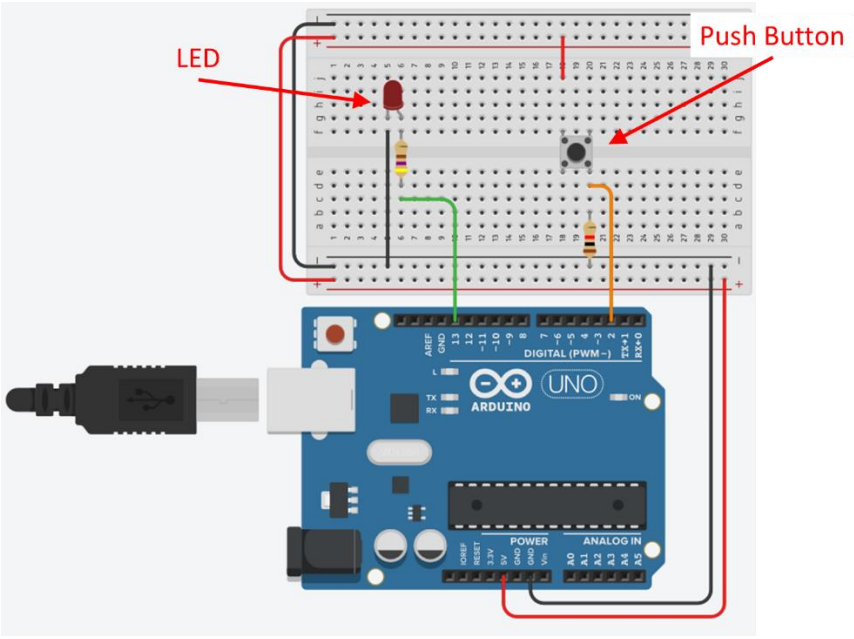


Figure 5. Diagram illustrating the Circuit configuration for the Push Button and LED Exercise

The code for this exercise can be found by selecting the in-built Button exercise from the Files menu in the Arduino IDE: Files > Examples > 02.Digital > Button

Procedure:

- Run Button example Arduino sketch: you should see the LED light up when you press the button, and the LED should go out when you release the button.

It is often the case that we may wish to view data directly from within an Arduino program. To achieve this we can use the Serial library to stream data from the Arduino, across a COM interface (this can be the same COM port you are using to program the Arduino) to a terminal – we can use the Serial Monitor in the Arduino IDE for convenience.

The code, shown below, is based on the Button example, but has two extra lines, highlighted with a red box. These extra commands initialise the Serial interface, and write data to the COM port. A description of the Serial commands can be found in the Serial functions section of the Arduino reference guide:

https://www.arduino.cc/reference/en/language/functions/communication/serial/

Links to all the Serial function descriptions are listed towards the bottom of the Serial commands reference page.

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize communication with the Serial monitor. Parameter : baud rate
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

To open the Serial Monitor, from the Arduino IDE, click the button in the top right of the IDE, as highlighted in Figure 6.

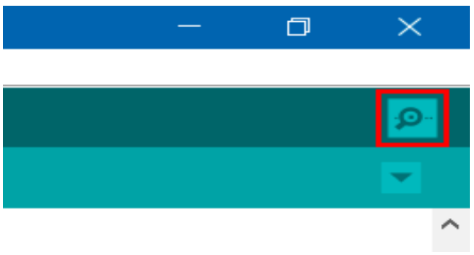


Figure 6. Screen shot of the Serial Monitor button, in the Arduino IDE

Procedure:

- Modify the Button example sketch, and open the Serial Monitor
- Run the code: You should see a ‘1’ printed on the serial monitor when the button is pressed, and the serial monitor should display ‘0’ when the button is released.

4.3 Exercise 3: LED Pattern

Write a program to create a changing sequence of LEDs, using the circuit shown in Figure 7.

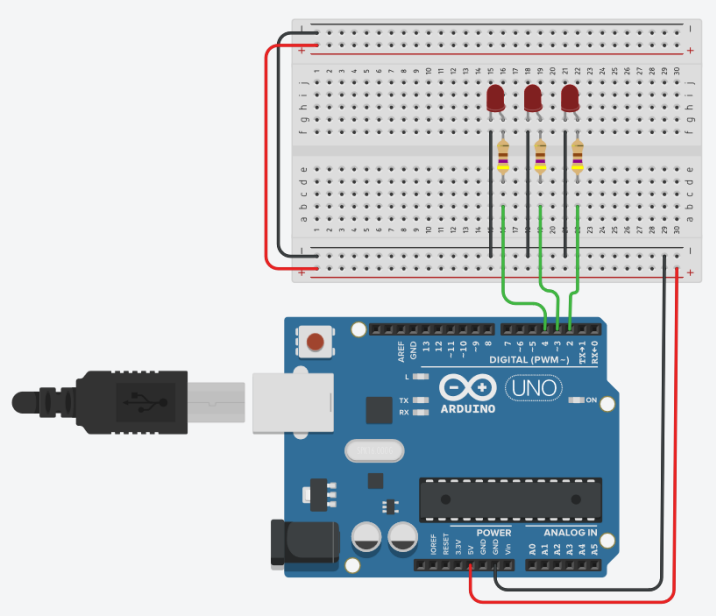


Figure 7. Diagram illustrating the Circuit configuration for the LED Pattern

Your program should change the pattern on the LEDs at fixed time steps, e.g. every 0.5 second, and loop through a non-random sequence. An example of a non-random sequence could be counting up in binary, as illustrated in Table 1.

Sequence Index	Pin 4 Output	Pin 3 Output	Pin 2 Output
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 1. Example of a simple repeating sequence, (counting-up in binary)

We will not provide any example code to facilitate this exercise, you must create this code from scratch, possibly using some of the example code provided in this lab sheet or tutorial examples.

Note: *This Exercise will need to be demonstrated in the final video assignment. See the Introduction to Arduino assignment brief on the Blackboard site for more details.*

4.4 Exercise 4: Switch Debounce

Switch bounce occurs when the mechanical contacts of a switch move from one position to another, causing intermittent toggling of the switch signal, a description of this is provided on the Pololu website:

https://www.pololu.com/docs/0J16/4

The BounceDemo.ino demonstration program, available from the Blackboard folder for this activity, provides a demonstration of the problem of switch bounce. A jumper wire is connected in and out of circuit between pin 2 and GND, to simulate the opening and closing of a switch

or button contacts. The unDebouncedCounter counts all of the detected on pin 2 of the microcontroller.

Note: *During this exercise, you should use a jumper wire directly between the pin 2 socket and GND terminals, because the quality of the dynamic contact is far poorer than the button provided in the kit box, (switches and buttons are optimised to minimise this affect). Consequently, using the jumper wire better demonstrates the problems of switch bounce in digital circuits.*

This code uses an interrupt to detect the change in state of the input pin, and the associated interrupt service routine increments a counter on each detected state change.

Note: *Interrupt are not covered in the teaching material for this module, and you will not expected to use them for core elements of this module, but they are a very useful tool to interact with external events, outside the microcontroller, and may be a useful toll when you come to implement your end of module project.*

A short demonstration video of BounceDemo.ino is provided on the Blackboard site to demonstrate this program and the requirements for the following exercise:

Procedure:

- Further investigate switch debounce, and how this can be achieved in hardware, as well as software.
- Within the user code area of BounceDemo.ino, implement a method to debounce the switch signal on pin 2, and increment the debouncedCounter variable each time the wire links or unlinks pin 2 to GND – see the demonstration video on the Blackboard site for more details.

Note: *your code should be inserted in between the starred lines marked in BounceDemo.ino:*

```
// ************************************************************

// Add your user code between these two starred lines



// ************************************************************
```

BounceDemo.ino contains some extra functions/keywords that you may not have previously come across. Below is some of these functions/keywords and links to the Arduino reference guide describing them:

Millis()

https://www.arduino.cc/reference/en/language/functions/time/millis/

Volatile

https://www.arduino.cc/reference/en/language/variables/variable-scope-qualifiers/volatile/

attachInterrupt

https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

A further resource, with a brief introduction to the use of interrupts with Arduinos is provided, below:

https://www.tutorialspoint.com/arduino/arduino_interrupts.htm

Note: *This Exercise will need to be demonstrated in the final video assignment. See the Introduction to Arduino assignment brief on the Blackboard site for more details.*

5 Acknowledgements

This document was written by Ben Taylor, with contributions from a number of authors: Dana Damian, Shuhei Miyashita, and Kilian Marie.

Introduction to Arduino: Post-Laboratory Assignment

There is a video assignment for the three Introduction to Arduino laboratory exercises. The exercises you are required to demonstrate in the video are:

- **Digital I/O - Exercise 3: LED Pattern**
- **Digital I/O - Exercise 4: Switch Debounce**
- **Analogue Sensors - Exercise 2: Calibrate Potentiometer Angle**
- **Analogue Sensors - Exercise 3: Measure the Output of the Sharp IR Distance Sensor**
- **Analogue Sensors - Exercise 5: Use the 1-D Lookup table to calibrate the IR sensor**
- **PWM Control of Actuators - Exercise 2: Controlling a Standard Servomotor**
- **PWM Control of Actuators - Exercise 3: Controlling a DC motor, using a driver board**

See the Introduction to Arduino assignment brief on the Blackboard site for more details.