Sheffield Hallam University
School of Engineering and Built Environment

Sheffield Hallam University

Activity ID	Activity Title	Laboratory Room No.	Level
Lab 1	An introduction to Arduino	4302	Beginner

Equipment (per student/group)

Number	Item
1	PC
1	Arduino kit

Learning Outcomes

	Learning Outcome
3	Develop good laboratory practice and demonstrate knowledge, understanding and ability to safely use relevant materials, equipment, tools, processes or products.
4	Demonstrate competent use of prototyping, modelling and basic analytical techniques.

An introduction to Arduino

Introduction

Computing is about more than the PC on your desktop! Embedded devices are everywhere – from wireless telecommunications infrastructure points to electronic point of sale terminals. One definition of an embedded system is:

> "**An embedded system is a computer system designed to perform one or a few dedicated functions** often with real-time computing constraints."
> (http://en.wikipedia.org/wiki/Embedded_system)

In this series of labs you are going to be introduced to the open source Arduino platform – a cheap, simple to program, well documented prototyping platform for designing electronic systems. The heart of this prototyping system is the Arduino Uno microcontroller board itself which is based on the commonly used ATMega328 chip.

The purpose of this introductory lab session is to introduce the Arduino Uno board and show you how to set up the development environment. You will then develop a simple Arduino program to blink a single LED (this is the embedded electronics version of Hello World!) and learn a bit about the programming language the Arduino platform uses.

Bibliography

There are no essential bibliographic resources for this laboratory session aside from this tutorial sheet. However the following websites and tutorials may be of help (especially if you haven't done much electronics previously or your digital logic and/or programming is a bit rusty):

- http://www.arduino.cc/
- http://www.ladyada.net/learn/arduino/index.html
- http://tronixstuff.wordpress.com/tutorials/

Methodology

Check that you have all the necessary equipment (see Figure 1)!

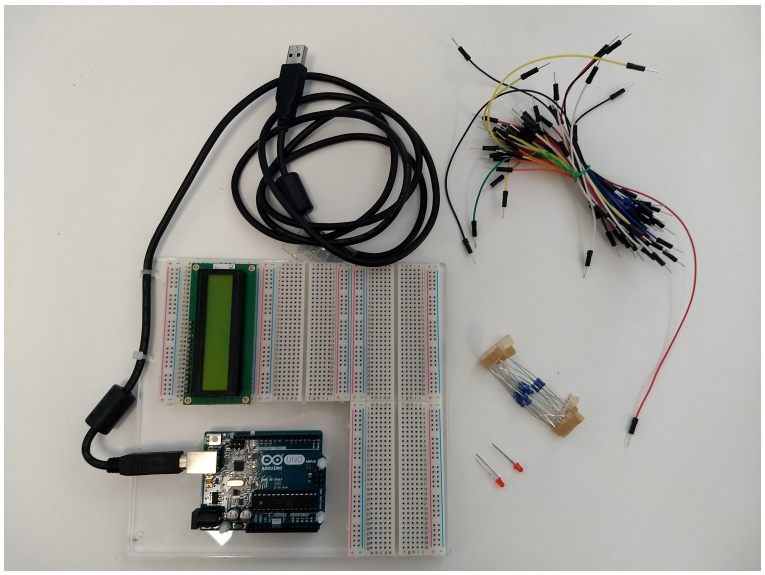

Figure 1 – The necessary equipment for this lab (don't worry if you have LEDs of a different colour)

Task 1

1. Open the Arduino IDE by double clicking the Arduino application in the Arduino folder or searching on the Start Menu.

2. Select the correct board (Arduino Uno) and serial port (see Figures 2 and 3).

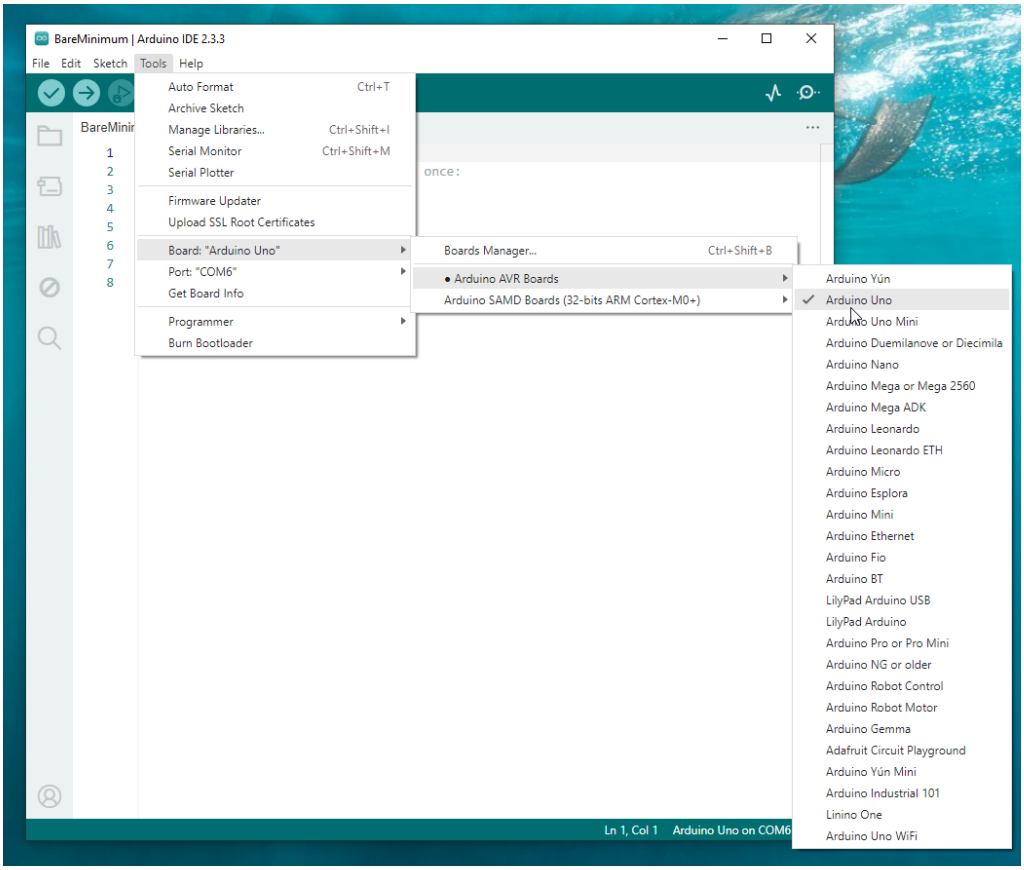

Figure 2 – Selecting the correct Arduino board

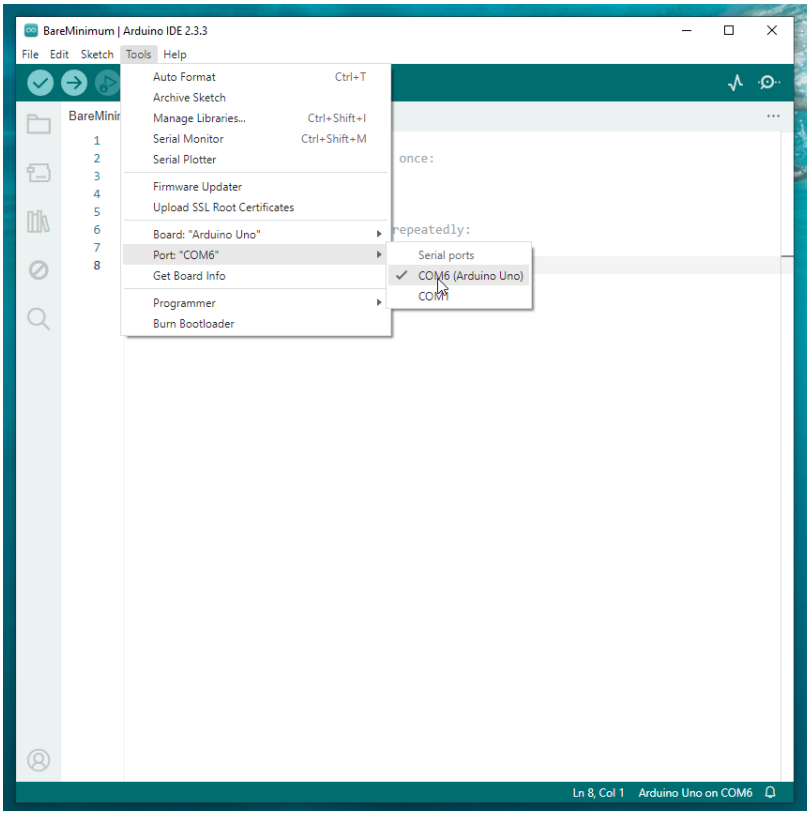

Figure 3 – Selecting the serial port

3. Enter the program shown in code listing 1 below (in Arduino parlance this is called a “sketch”).

Code Listing 1: The Blink Sketch

```
/*
 * 1_blink.ino
 *
 * this is a simple Arduino program to blink an led (the electronics
 * equivalent of hello world). this program turns an led on for 1
 * second and then off for 1 second, and so on ...
 *
 * this program uses pin 13 for the led
 *
 * author: prof. alex shenfield
 * date:   2024/11/12
 */

// the led is connected to pin 13
const int ledPin = 13;

// CODE

// this method runs once (when the sketch starts)
void setup()
{
  // sets the digital pin as output
  pinMode(ledPin, OUTPUT);
}

// this methods loops continuously
void loop()
{
  // turn led on, wait for 1 second, turn led off, wait for 1 second,
  // repeat ...
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

4. Verify / compile the sketch (by pressing the “tick” icon in the toolbar of the Arduino IDE) to make sure that it is correct.

5. Upload the sketch to the Arduino board (by pressing the right facing arrow icon in the toolbar of the Arduino IDE).

6. Insert the LED (as in Figure 4).

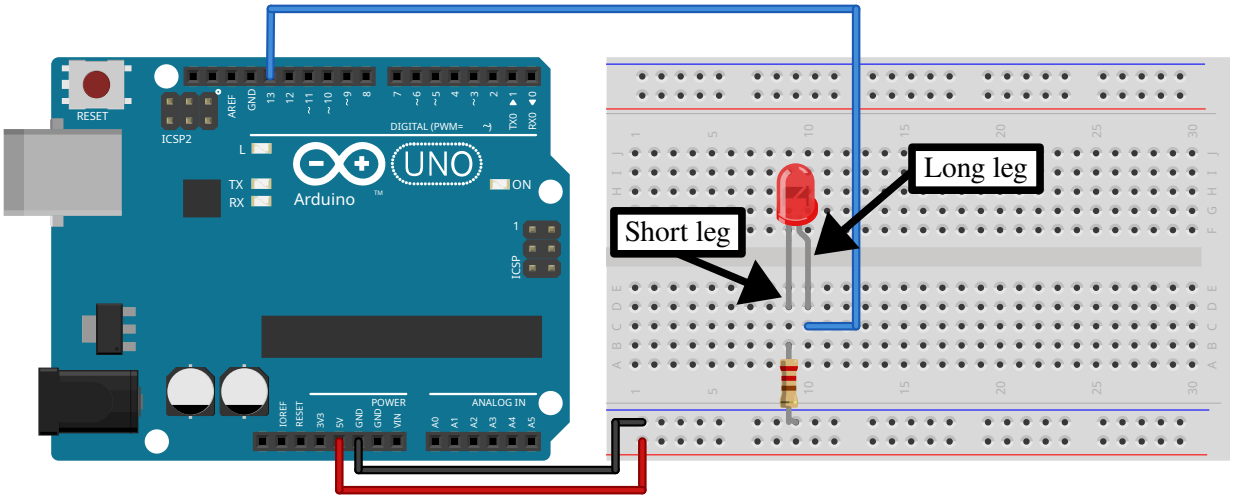

Figure 4 – Basic “blink” circuit

A detailed explanation ...

We are now going to step through the 1_blink.ino Arduino sketch entered above and explain the purpose of each element.

```
/*
 * 1_blink.ino
 *
 * this is a simple Arduino program to blink an led (the electronics
 * equivalent of hello world). this program turns an led on for 1
 * second and then off for 1 second, and so on ...
 *
 * this program uses pin 13 for the led
 *
 * author: prof. alex shenfield
 * date:   2024/11/12
 */
```

This is a **block comment**. The text between the /* and */ symbols is ignored by the Arduino; however, it still serves an extremely important purpose – making the program readable to human eyes! This is important even if the code you write is never going to be seen by other people, as it helps you understand what a program does when you come back to it (you may think "oh that's simple, I don't need to explain what that does", but when you come back to it in a week / month / year it will suddenly not make any sense to you!).

At the start of each program you write you should put a block comment with the name of the program, a short description of what it does, and the author details (name, date, etc.).

```
// the led is connected to pin 13
const int ledPin = 13;
```

This is the first line of code that the Arduino will actually do anything with. It is a **statement** assigning a value to a **variable**. If you haven't done much programming before this statement may look unfamiliar to you. However, this is the general form of all Arduino statements (and the syntax is the same in many other programming languages). Firstly we need to tell the Arduino what the type of the variable is (in this case an integer), then the identifier for the variable (in this case ledPin), and finally what the value should be (in this case 13). All statements in the Arduino language should be ended by a semi-colon (;) as this lets the Arduino know that the statement is finished.

Note also that there is a single line comment after the statement (denoted this time by //). This is just to remind us what the statement does.

```
// CODE

// this method runs once (when the sketch starts)
void setup()
{
  // sets the digital pin as output
  pinMode(ledPin, OUTPUT);
}
```

This bunch of code is called a **method** (or procedure), and it is basically a collection of statements that we can then refer to by one name. We can see that this method doesn't return anything (it's return type is **void**) and doesn't require any inputs. The purpose of the **setup()** method is just to tell the Arduino what should be initialised (in this case one of the physical pins on the chip should be treated as an output).

```
// this methods loops continuously
void loop()
{
  // turn led on, wait for 1 second, turn led off, wait for 1 second,
  // repeat ...
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

As with the **setup()** method we have just looked at, this **loop()** method also has no inputs or outputs. However, unlike the **setup()** method, this method contains multiple statements. This method calls two other methods – **digitalWrite()** and **delay()**. We will now look at these methods in a bit more detail:

digitalWrite() takes two inputs – a pin number (of type integer) and a logical value (either HIGH or LOW). In this sketch these values correspond to either turning the LED on (HIGH) or off (LOW). In more general terms this command switches the specified pin from 0 to 5V (HIGH) and back again (LOW)

delay() takes one input – the time (in milliseconds) to pause the execution of the method for.

It should also be noted that **loop()** - like **setup()** - is another special method. In this case **loop()** contains the main body of the program and, as the name suggests, is executed again and again until the Arduino is turned off (or another program is loaded on to it).

Exercises ...

Now make some more alterations to your Blink sketch.

a) Change the number in the **delay()** method call to 500 and compile and save the sketch. Does the LED blink faster or slower?

b) Modify the code so that the LED is on for 100 ms and off for 900 ms.

Task 2

Sometimes you need to do two things at once. For example you might want to blink an LED (or some other time-sensitive function) while reading a button press or other input. In this case, you can't use delay(), or you'd stop everything else the program while the LED blinked. The program might miss the button press if it happens during the delay(). This sketch demonstrates how to blink the LED without using delay(). It keeps track of the last time the Arduino turned the LED on or off. Then, each time through the loop() method, it checks if a long enough interval has passed. If it has, it toggles the LED on or off (depending on it's previous state).

Firstly we will set up the same LED circuit as previously (see Figure 4, previously).

Now we can use the program in code listing 2 to implement the blink sketch without a delay function.

Code Listing 2: Blink without a delay

```
/*
 * 2_blink_without_delay.ino
 *
 * this is a simple Arduino program to blink an led (the electronics
 * equivalent of hello world) but without using the delay statement.
 * this program turns an led on for 1 second and then off for 1
 * second, and so on ...
 *
 * this program uses pin 13 for the led
 *
 * author: prof. alex shenfield
 * date:   2024/11/12
 */
```

```
// the led is connected to pin 13
const int ledPin = 13;

// timing variables ...
long previousTime = 0;   // the last time the led was updated
long interval = 1000;    // the blink interval
int  ledState = LOW;     // initial state of led (off)

// CODE

// this method runs once (when the sketch starts)
void setup()
{
  // sets the digital pin as output
  pinMode(ledPin, OUTPUT);
}

// this methods loops continuously
void loop()
{
  // get the current time this time round the loop
  unsigned long currentTime = millis();

  // if the set time interval has elapsed ...
  if (currentTime - previousTime > interval)
  {
    // save the time
    previousTime = currentTime;

    // blink the led (by flipping the led state)
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }

    // set the led to the new led state
    digitalWrite(ledPin, ledState);
  }
}
```

Compile and upload this code to your Arduino board. You should see the Arduino blinking the LED in much the same way as the previous task – the difference is that now the Arduino can do other stuff too! We will see how this is useful in the next set of lab exercises.

Check list

Task 1 – Program	
Task 1 – Exercise a)	
Task 1 – Exercise b)	
Task 2 – Program	

Feedback/ reflections