Sheffield Hallam University
Department of Engineering
BEng (Hons) Electrical and Electronic Engineering
BEng (Hons) Mechanical Engineering

Activity ID		Activity Title			Laboratory Room No.	Level
Lab 101		An introduction to Arduino			4302	6
Semester	**Duration [hrs]**	**Group Size**	**Max Total Students**	**Date of approval**	**Lead Academic**	
1	2	1	25	09-24	Alex Shenfield	

Equipment (per student/group)

Number	Item
1	PC
1	Arduino kit

Learning Outcomes

	Learning Outcome
3	Select and apply appropriate embedded networking technologies for a given problem
4	Design, implement and test embedded networked devices

Risk Assessment

General risk assessment for Sheaf 4302 applies (see blackboard for more details).

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. Although the Arduino platform started out with the Arduino Uno (based on the commonly used ATMega328 chip), other, more capable, boards have since been released. In this series of labs you will be using the Arduino MKR WiFi 1010 board – a Cortex M0+ based Arduino board with built in WiFi module.

The purpose of this introductory lab session is to introduce the Arduino MKR WiFi 1010 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/

Task 1

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

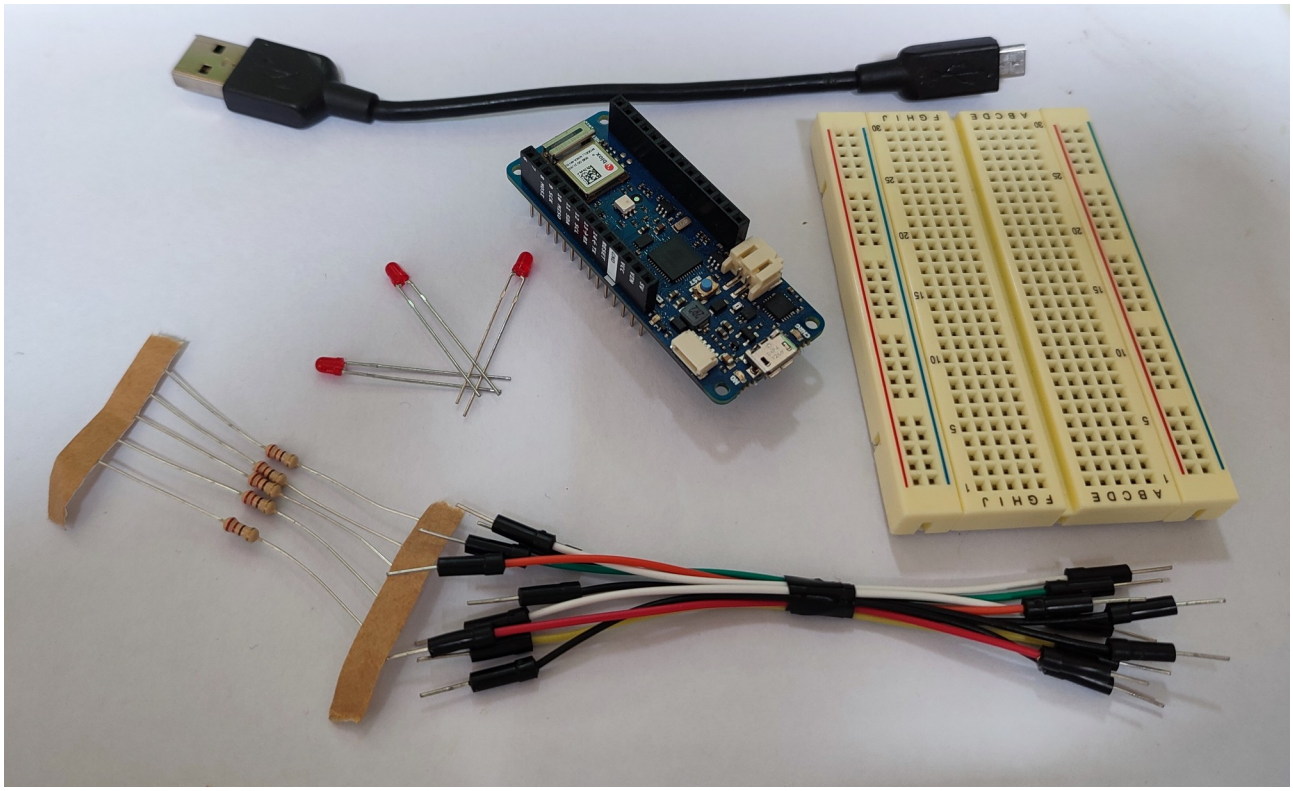

Figure 1 – The necessary equipment for this lab (don't worry if you have an LED of a different colour)

2. Open the Arduino IDE through AppsAnywhere on the SHU managed desktop PCs (or by double clicking the Arduino application in the Arduino folder or searching on the Start Menu on home computers).

3. We then need to choose the Arduino MKR WiFi 1010 board in the board select menu (see Figure 2).

Figure 2 – Selecting the appropriate Arduino board

4. This then tells you that the board support package is not installed and asks if you want to install it (see Figure 3). Click "YES" here and it will install everything you need to compile and upload code to the Arduino MKR WiFi 1010 boards. Installing this on my home machine takes approximately 3 minutes – though it might take longer if you need to do it on the SHU managed desktop.

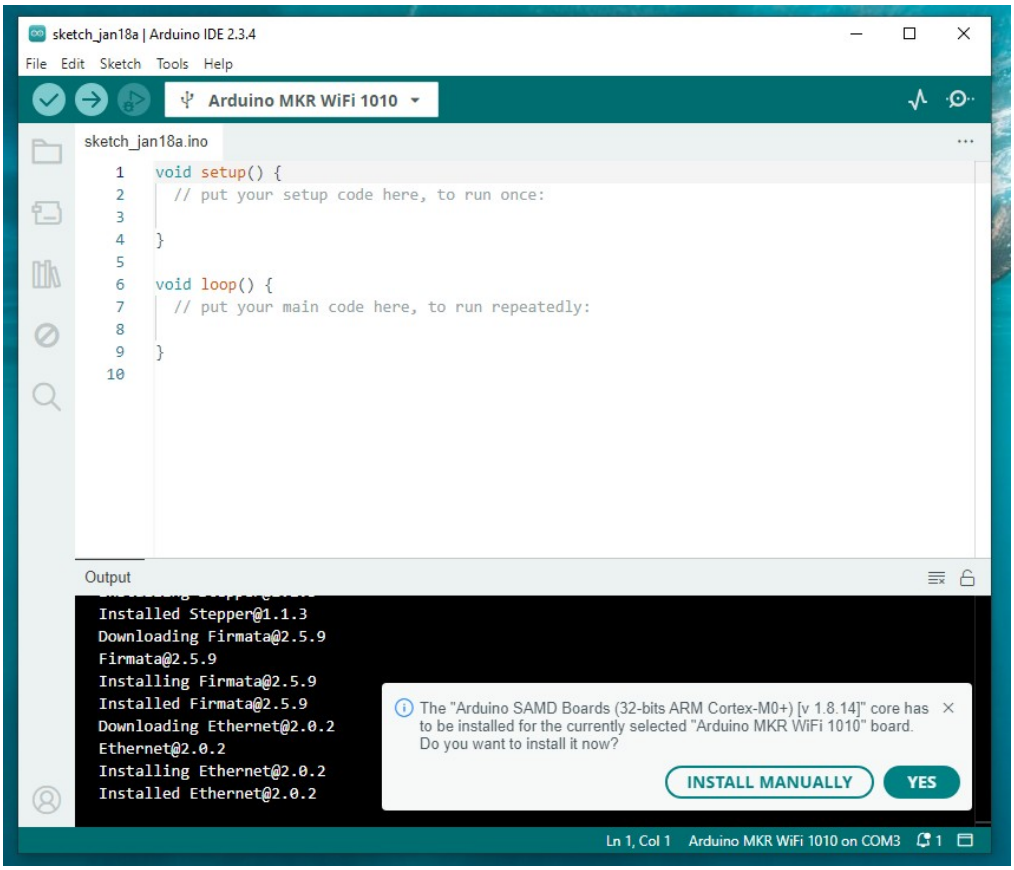

Figure 3 – Installing the appropriate board support packages

5. 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 6 for the led (this is the built in led on the MKR
 * series of boards).
 *
 * author: prof. alex shenfield
 * date:   2024-11-12
 */

// INITIAL VARIABLES

// the led is connected to pin 6
const int led_pin = 6;

// CODE

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

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

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

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

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

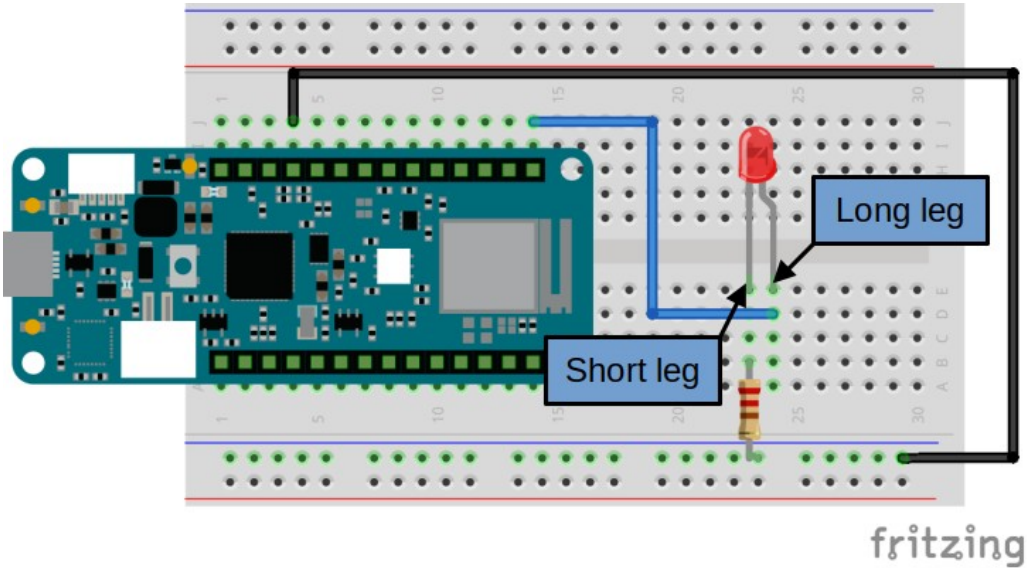

Figure 4 – Basic “blink” circuit

A detailed explanation ...

We are now going to step through the 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 6 for the led (this is the built in led on the MKR
 * series of boards).
 *
 * 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 6
const int led_pin = 6;
```

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 6). 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 before the statement (denoted this time by //). This is just to remind us what the statement does.

```
// this method runs once (when the sketch starts)
void setup()
{
  // sets the digital pin as output
  pinMode(led_pin, 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(led_pin, HIGH);
  delay(1000);
  digitalWrite(led_pin, 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 3.3V (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 was doing while the LED blinked. The program might then 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 loop(), 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 6, 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). This program turns an led on for 1 second and then off for
 * 1 second, and so on ...
 *
 * unlike the previous example, this version uses timing variables to blink the
 * led without using a delay statement. this frees up your code to do other
 * things (such as reading from sensors, handling network communication, etc.)
 *
 * this program uses pin 6 for the led (this is the built in led on the MKR
 * series of boards).
 *
 * author: prof. alex shenfield
 * date:   2024-11-12
 */

// the led is connected to pin 6
const int led_pin = 6;

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

// CODE

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

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

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

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

    // set the led to the new led state
    digitalWrite(led_pin, led_state);
  }
}
```

This pattern is known as a **state-machine** (because we are storing some variables relating to the **state** of the LED – i.e. whether the LED is currently on or off and when it last changed – and some variables relating to the **transition** between states – i.e. the on interval and off interval).

Now try changing the on / off interval times and adding a new state machine (with different transition times) for an additional LED. This is something that would be extremely difficult to do if we were using **delay()**!

Check list

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

Feedback