



L-Università
ta' Malta

ESE 2023 Summer Training:

Automation and Prototyping





L-Università
ta' Malta

ESE 2023 Summer Training:

Day 3:

Final setup and validation

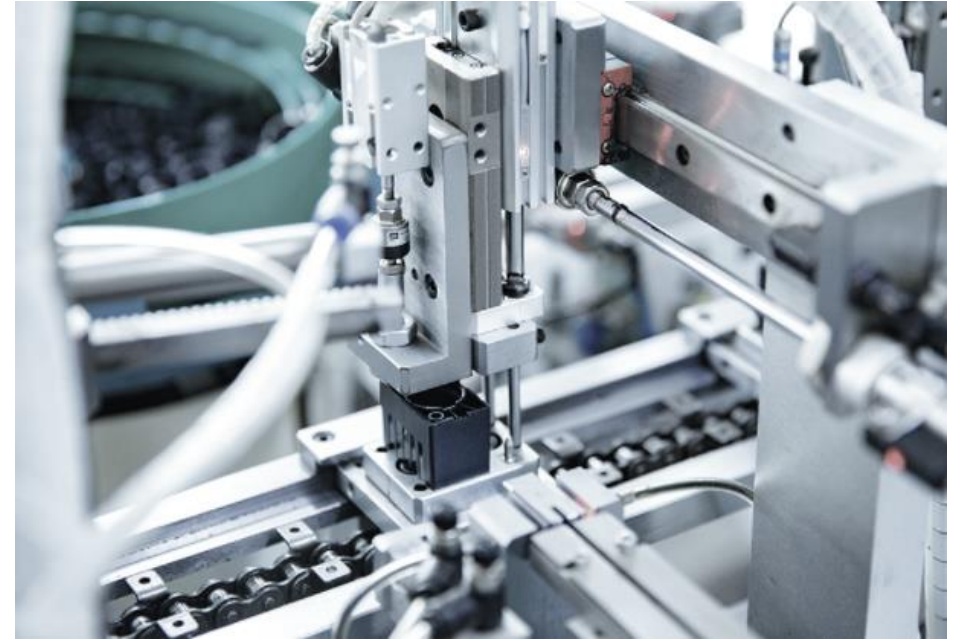


Day 3:

What we will be covering:

Defining requirements and follow-up.

- Installing the interface board
- Setting up the Arduino IDE
- Wiring-up the whole system
- Running the system



Project Description:

To build a pneumatically-operated system that will perform a simple OFF/ON/OFF functionality test of a momentary type, push button switch.



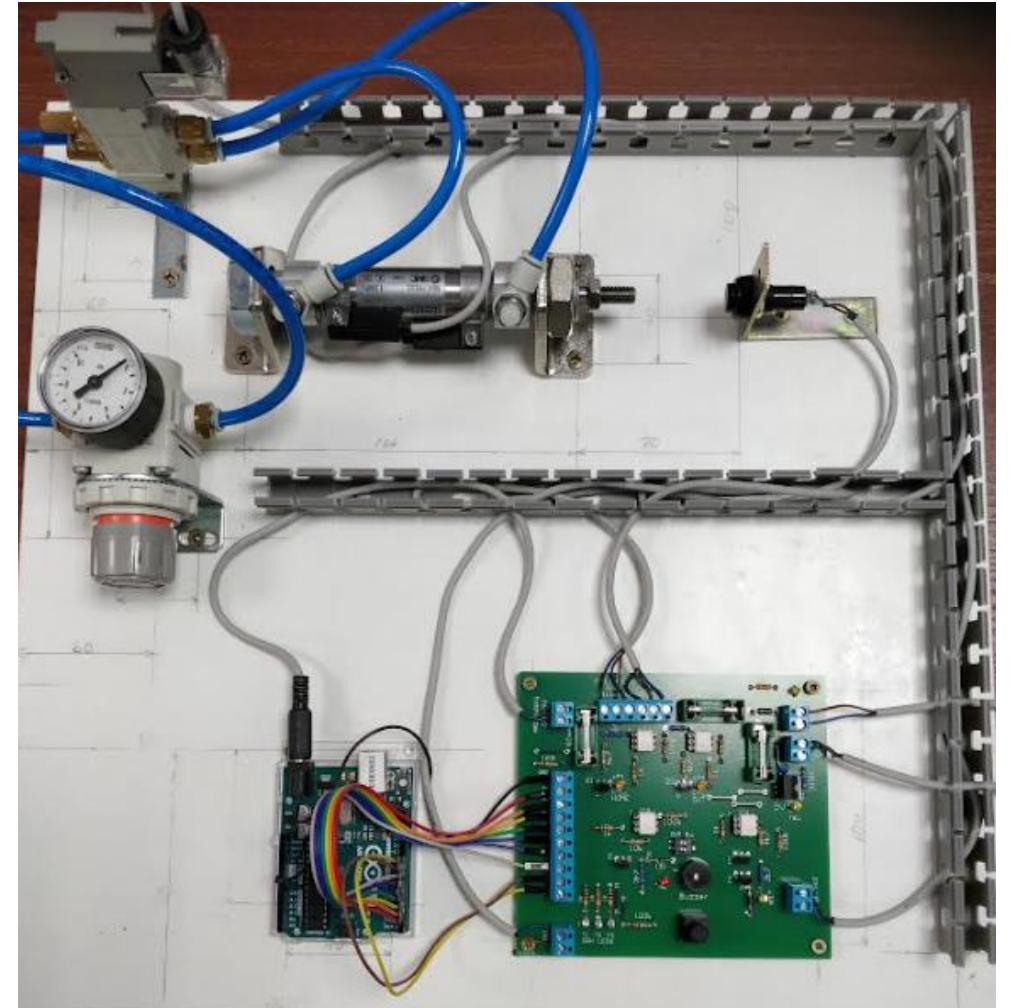
If our switch is good, then:

1. There should be an open circuit between its terminals when it is not activated
2. There should be continuity when the switch is pressed.
3. When released, there should once again be an open circuit between the terminals.

Setup

The next step is to physically mount the interface board on our test setup and to wire up the board to the:

- Cylinder sensors
- Valve solenoid
- DUT (Device Under Test) – this is the switch mounted on the bracket that we will be testing.
- The Arduino
- Two separate power supplies a 12V dc and a 24V supplies



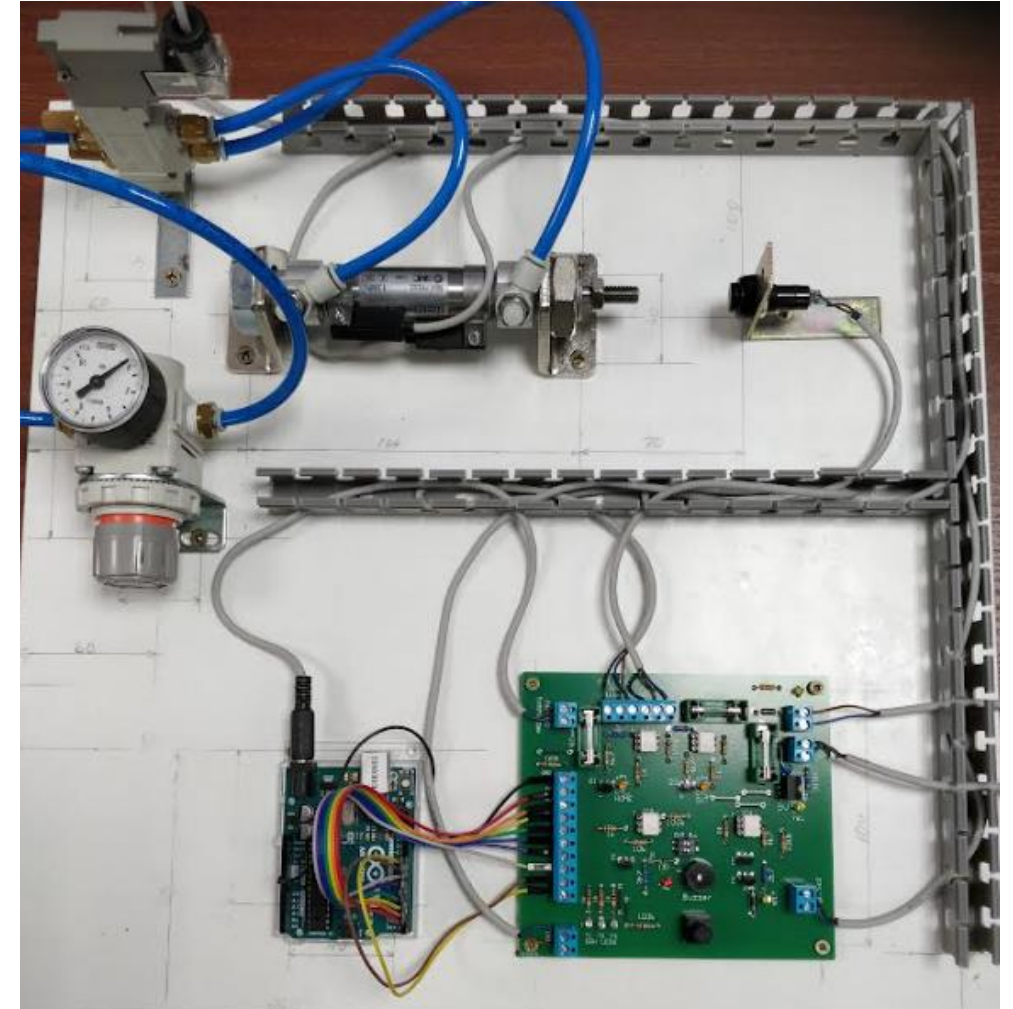
Setup (cont.)

Arduino inputs and outputs - this is the 10 way terminal block.

Terminal 1 on the interface board, goes to any one of the Arduino ground pins.

Terminal 2, (interface board), goes to Terminal 2 on the Arduino

Terminal 3 goes to terminal 3 on the Arduino and so on, right up to terminal 9.



Arduino Programming Software (IDE)

Before we can start using our Arduino, we need to download its software and we can do that by going to the official Arduino website at:

<https://www.arduino.cc/en/software>

The download procedure should be quite straightforward.

Once you have downloaded the software you can open up the IDE, (Integrated Development Environment), which we will consider in the next slide.

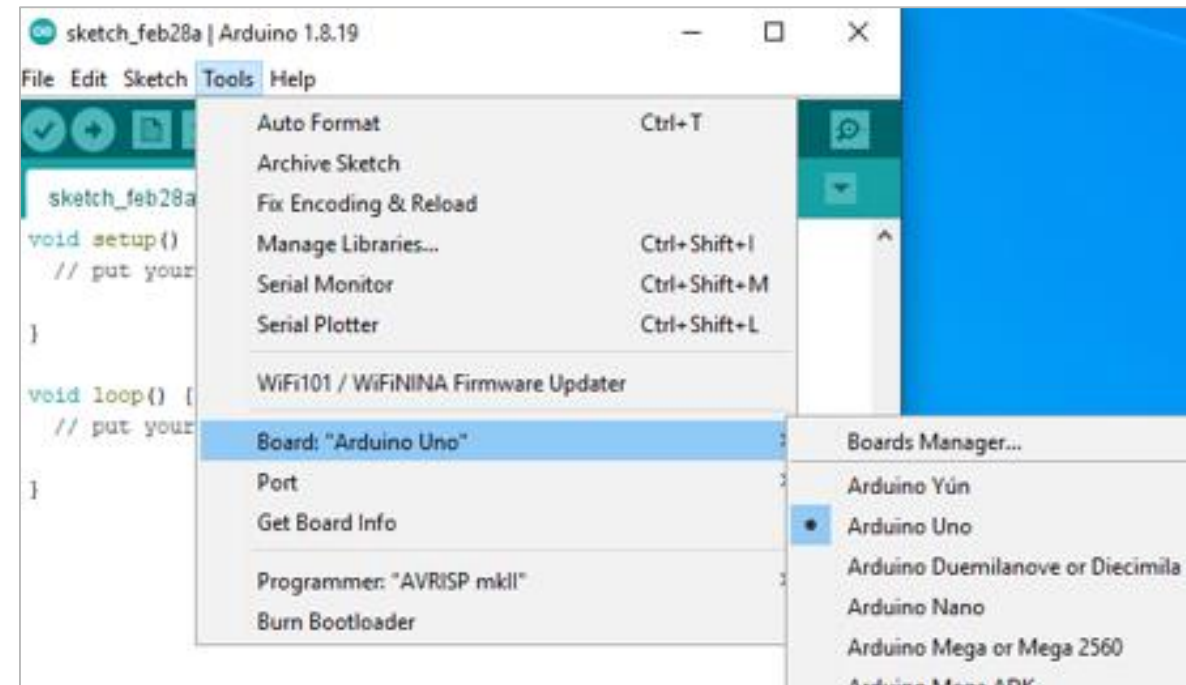
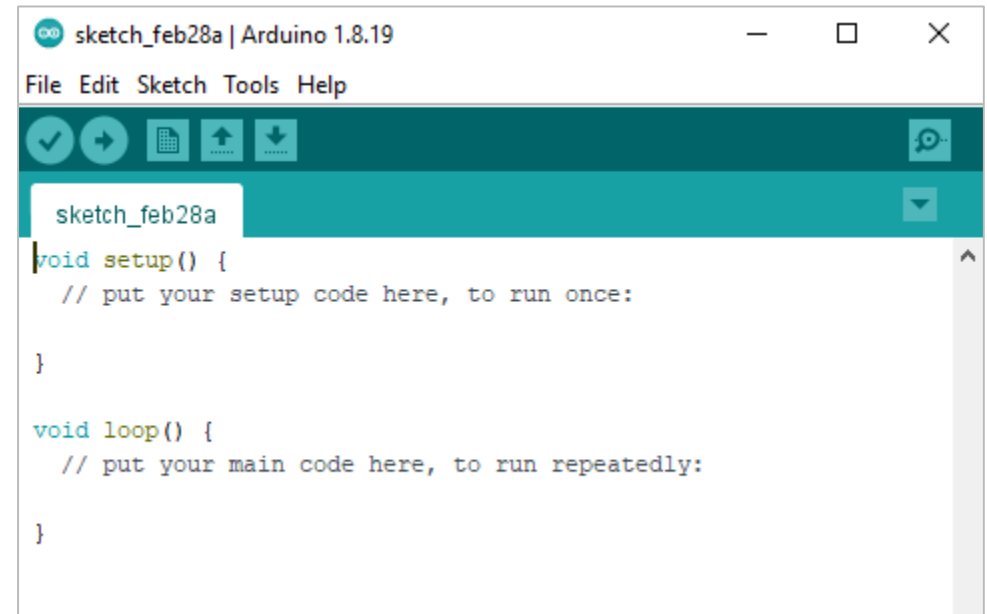
Once you have downloaded the software, the screen shown at top right, should show up.

The installation can now be completed in two easy steps:

Step 1

First, we define what type of Arduino we are using.

Tools > Board > Arduino > Arduino Uno



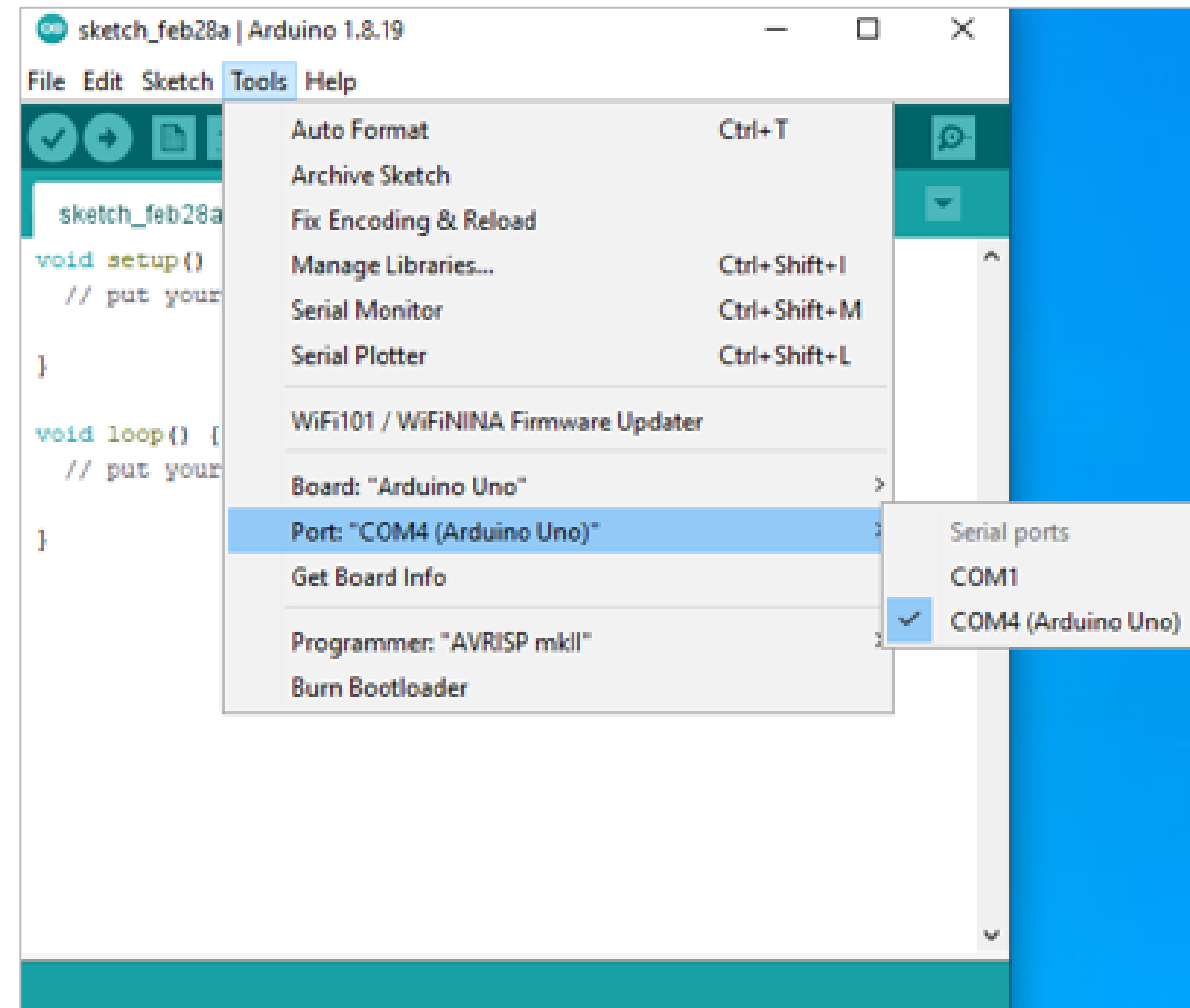
Step 2:

Define the port we will be using:

Tools > Port

Select the Com port which reads:
(Arduino Uno)

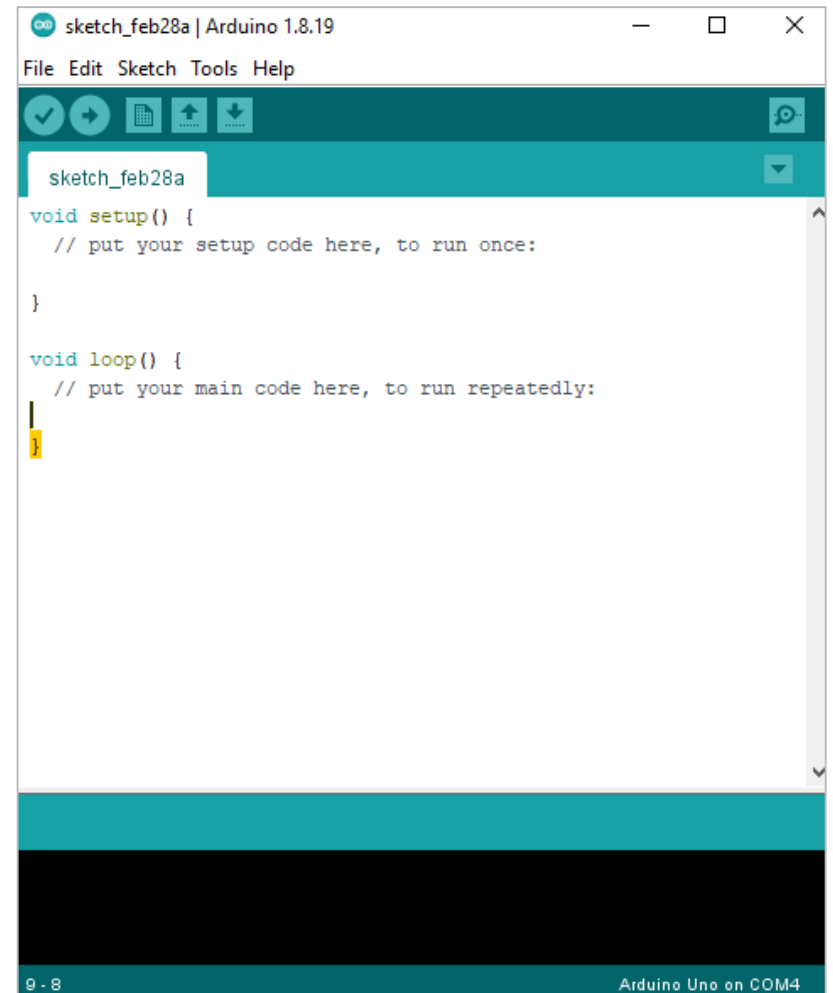
..and we're good to go!



At this point we are ready to write our very first program.

The great thing about Arduino is that the IDE itself contains several program examples that make it much easier for the beginner to become familiar with the skills needed to program this device.

In addition, there are thousands of programs on the web that provide solutions or ideas for almost any application.



Let's start by flashing an LED!

Go to:

File > Examples > 01. Basics > Blink

Here we will be blinking an LED which is on the Arduino board itself, so we will not need to do any wiring-up for this example.

As you might have noticed, the syntax is C-based.

There are two main functions that we need to take careful note of:

Void setup()

Void loop()



```
Blink | Arduino 1.8.19
File Edit Sketch Tools Help

Blink

/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
  */

// 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
}

1 Arduino Uno on COM4
```

Void setup()

The I.O. Pins of the Arduino are configurable. For example, we can define say, pin 4, to be an input or an output.

In the `Void setup()`, we will defining in the `pinMode` area, who is going to be an INPUT and who's going to be an OUTPUT.

Note that:

`(LED_BUILTIN, OUTPUT);` means that the LED, (marked L next to the TX and TR LEDs), will be our output.



```
Blink | Arduino 1.8.19
File Edit Sketch Tools Help

Blink

/*
 * Blink
 *
 * Turns an LED on for one second, then off for one second, repeatedly.
 *
 * Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
 * it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
 * the correct LED pin independent of which board is used.
 * If you want to know what pin the on-board LED is connected to on your Arduino
 * model, check the Technical Specs of your board at:
 * https://www.arduino.cc/en/Main/Products
 *
 * modified 8 May 2014
 * by Scott Fitzgerald
 * modified 2 Sep 2016
 * by Arturo Guadalupi
 * modified 8 Sep 2016
 * by Colby Newman
 *
 * This example code is in the public domain.
 *
 * https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
 */

// 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
}
```

1

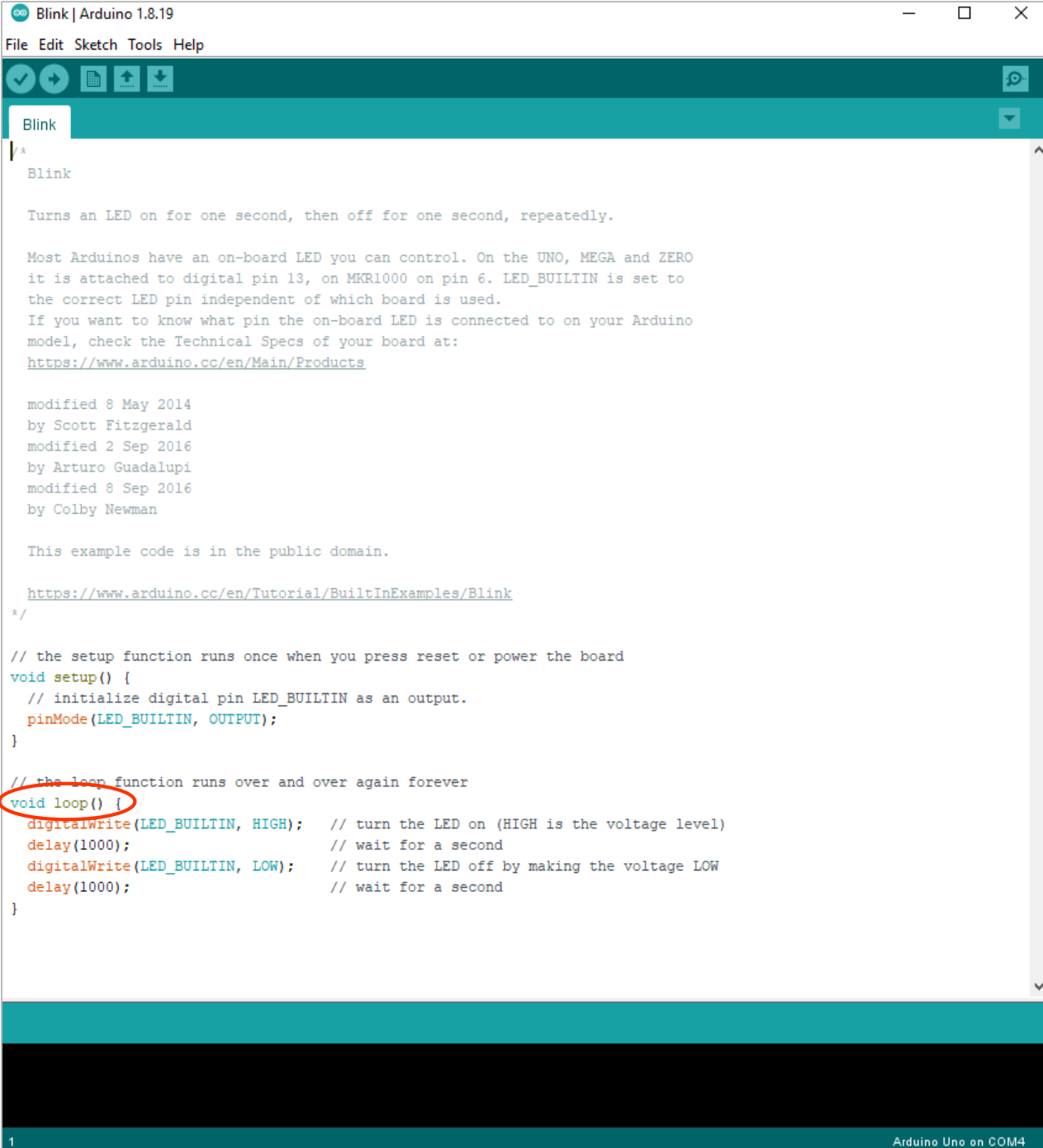
Arduino Uno on COM4

Void loop()

The **void loop** area is where we write our program which will, run a loop of commands for as long as the Arduino is in run mode. Here, we define under which conditions an output is to be activated. Let's take a look at this program:

digitalWrite(LED_BUILTIN, HIGH); Here, the Arduino is **writing** so there's going to be an **output** of some sort: Actually, we will be turning the LED on.

delay(1000); means that the program will sit and wait, and do nothing, for one second, (1000mS), and during this time, the LED will remain ON.



```
Blink

Turns an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
the correct LED pin independent of which board is used.
If you want to know what pin the on-board LED is connected to on your Arduino
model, check the Technical Specs of your board at:
https://www.arduino.cc/en/Main/Products

modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
modified 8 Sep 2016
by Colby Newman

This example code is in the public domain.

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// 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
}
```

1 Arduino Uno on COM4

Void loop() ..continued

digitalWrite(LED_BUILTIN, LOW); means that we will be turning the LED off.

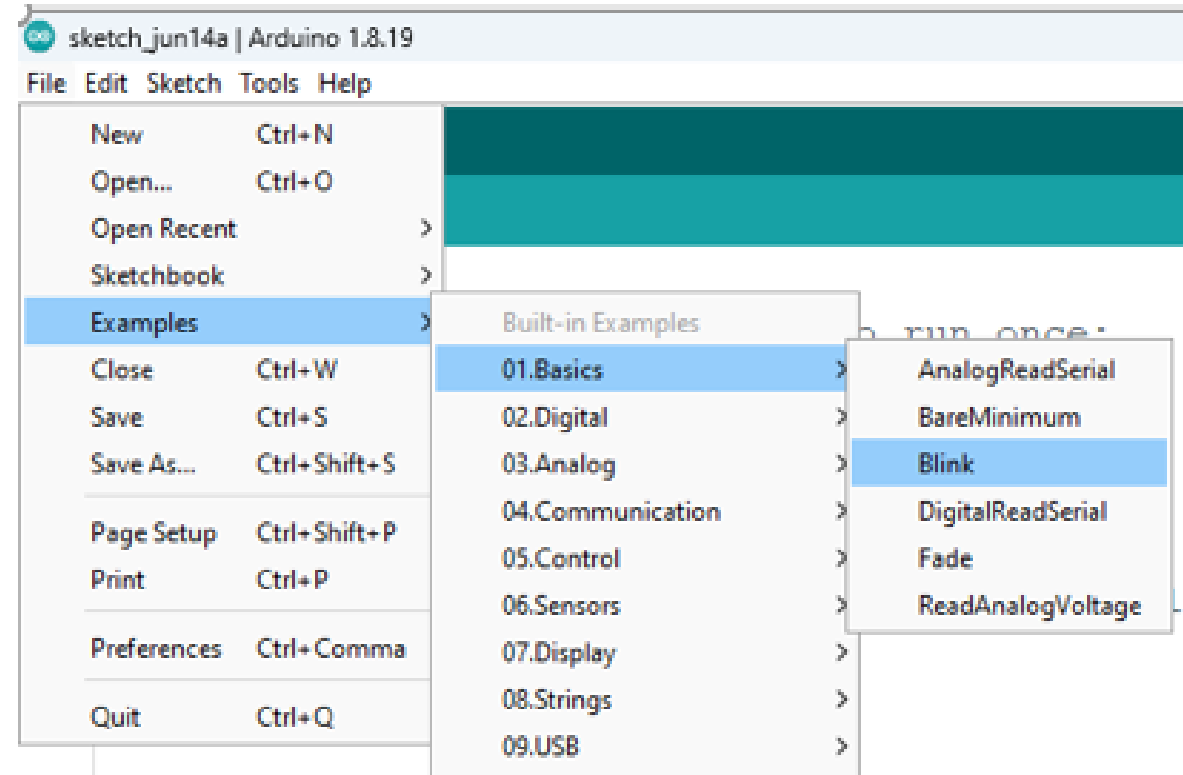
delay(1000); means that this time, we will keep the LED off for 1000mS or 1 second.

After that, the program will return to the first line and keep on looping endlessly.

```
// 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
}
```

Even if you are new to programming, there are so many programming ideas in the Examples file, that just by downloading and experimenting the examples, you will quickly get familiar with the basics. Once you do, then it will not be too hard to go to the next level like driving an external LED.



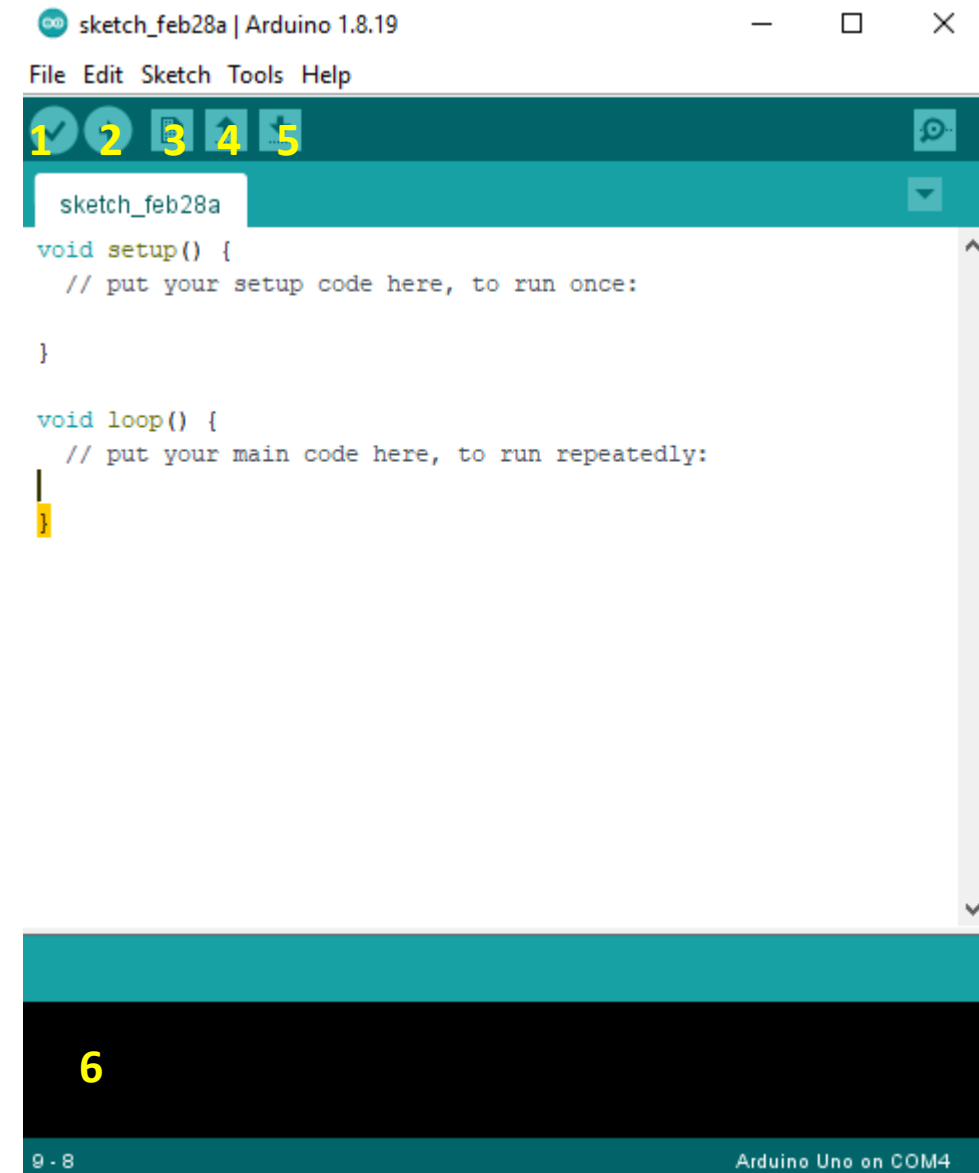
However, since the time we have available is very limited, we will leave that possibility for those of you who are interested in taking the initiative to learn more. On the web, you can find thousands of well-explained projects you can build.

Before we upload our LED blinker program, it's a good idea to make a brief reference to a few important buttons.

1. Compiles the code and points out any syntax mistakes in the black message area marked 6.
2. Uploads the program to the Arduino.
3. Opens a new blank sketch, (programs in Arduino are called sketches).
4. Opens up earlier sketches.
5. Saves the current sketch.

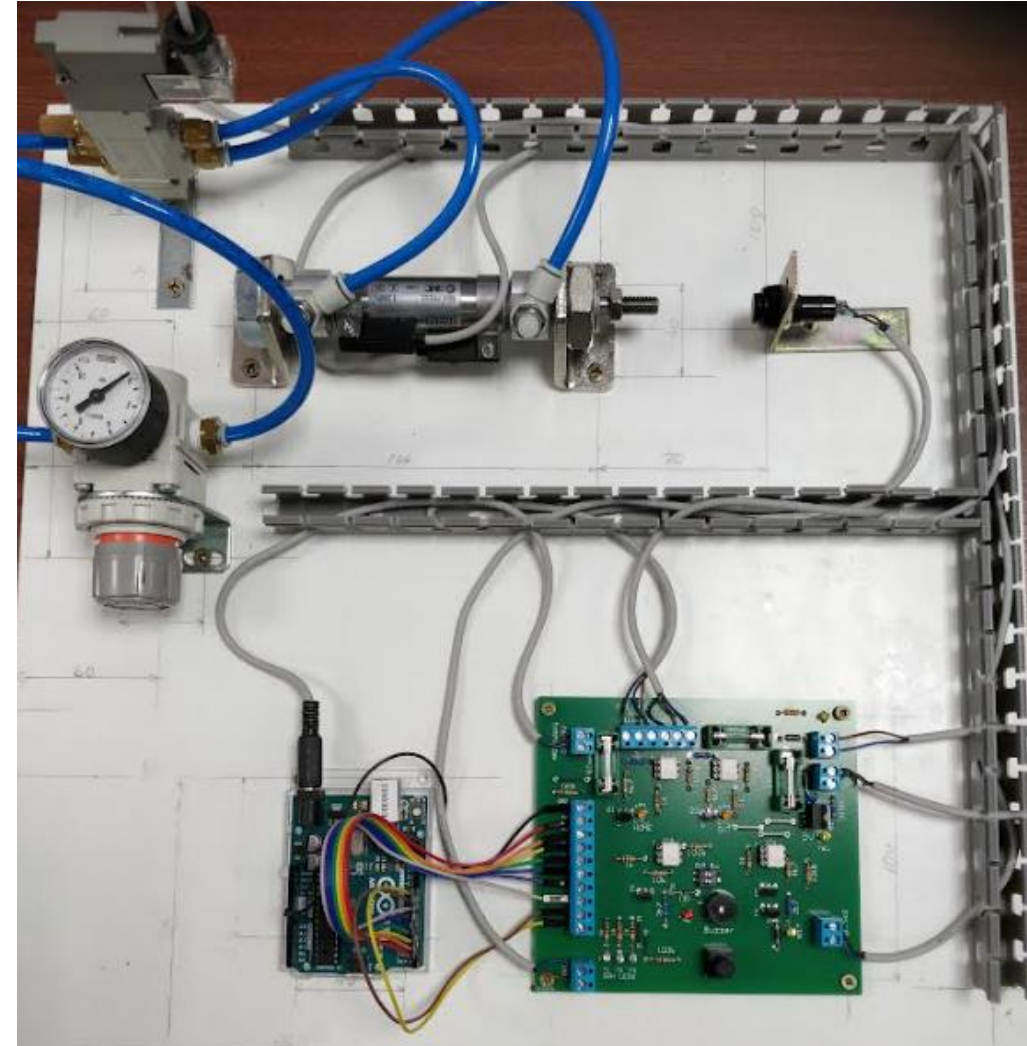
So, all we need to do is click on button 2 and we should notice the two adjacent LEDs flashing while the program is uploading.

After that, the LED should start to flash on and off at one second intervals. You can vary the blink rates by changing the delay timings.



In order to arrive at our goal of setting up our system to test a switch, we would basically need to write some code for:

1. The **actuation cycle** - here we need to provide the instructions that will activate the solenoid which will in turn switch the valve ports so that the cylinder rod will move outwards. We also need to ensure that the rod will only be extended outwards for say a second or two. So we will need to have a timer in the program so that the signal from the Arduino to the solenoid will be time-controlled.



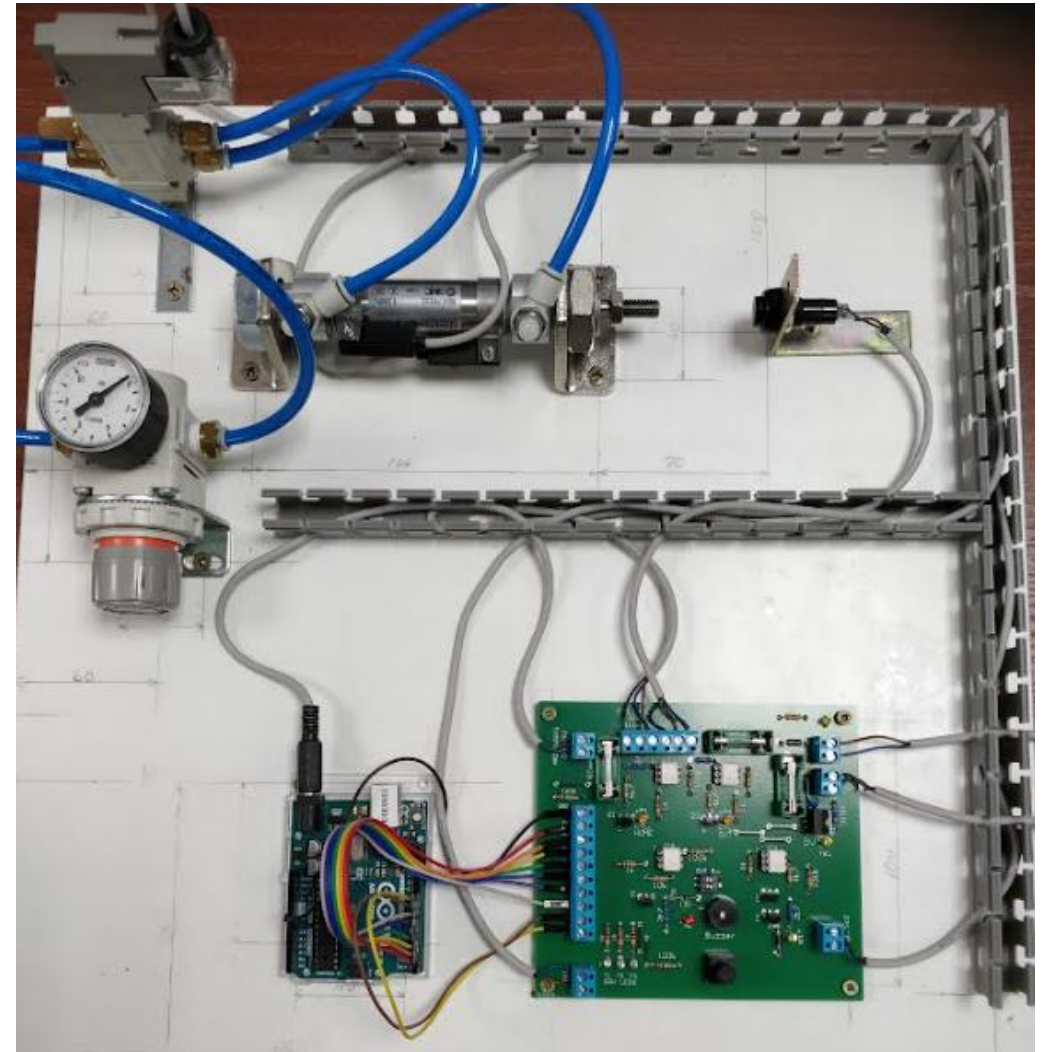
2. The **test routine** that will monitor the electrical state of the switch:

Before the button is pushed:
(1st test - check that switch is OFF)

While the button is pushed:
(2nd test - check that switch is ON)

When the button is released:
(3rd test – check that switch is OFF)

*You can upload the program from the pen drive,
which will be provided.*



Once you have done and checked all the wiring, you can test the whole setup.

So, once you press the start button on the interface board, then if the switch under test is ok, you should see:

- The first LED, (T1), light up, (no continuity between terminals of DUT).
- Cylinder rod move outwards, (DUT is now actuated and switch contacts should now be closed).
- The second LED, (T2), now lights up.
- Following a timeout, the cylinder rod should retract and the DUT is now open circuit again.
- The third LED, (T3), should now light up.
- After two or three seconds, the LEDs should go out as the program goes into reset mode, ready for the next test cycle.

