Girls into Electronics 2025 Student Guide

This guide will introduce you to the Arduino microcontroller platform, the Grove Beginner Kit for Arduino and basic programming skills. This guide is designed to give you the skills and confidence needed to undertake your own Electronics projects.

Contents

Introduction – What is Arduino?

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board called a microcontroller and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

There are many Arduino microcontrollers, but they all work in a similar way. The most widely used Arduino board is the Arduino Uno shown in *Figure 1*. We will use the Grove Beginner Kit for Arduino.

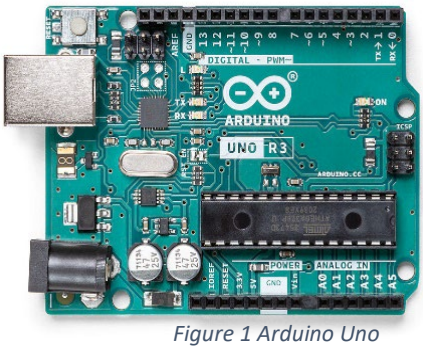


Figure 1 Arduino Uno

The Grove Beginner Kit for Arduino

This kit has an Arduino board and a set of sensors, with some input and output devices integrated into it.

If your box is sealed, it is recommended to use scissors to cut the seals on the front and the two side compartments. The side compartments contain a micro-USB cable (to power the board) on one side and 6 Grove cables (to link peripherals) on the other.

It is best, initially, to leave the board in the box.

Figure 2 labels all the attached devices. The microcontroller is the middle section that links to all the devices around it. This is a modified Arduino Uno microcontroller.

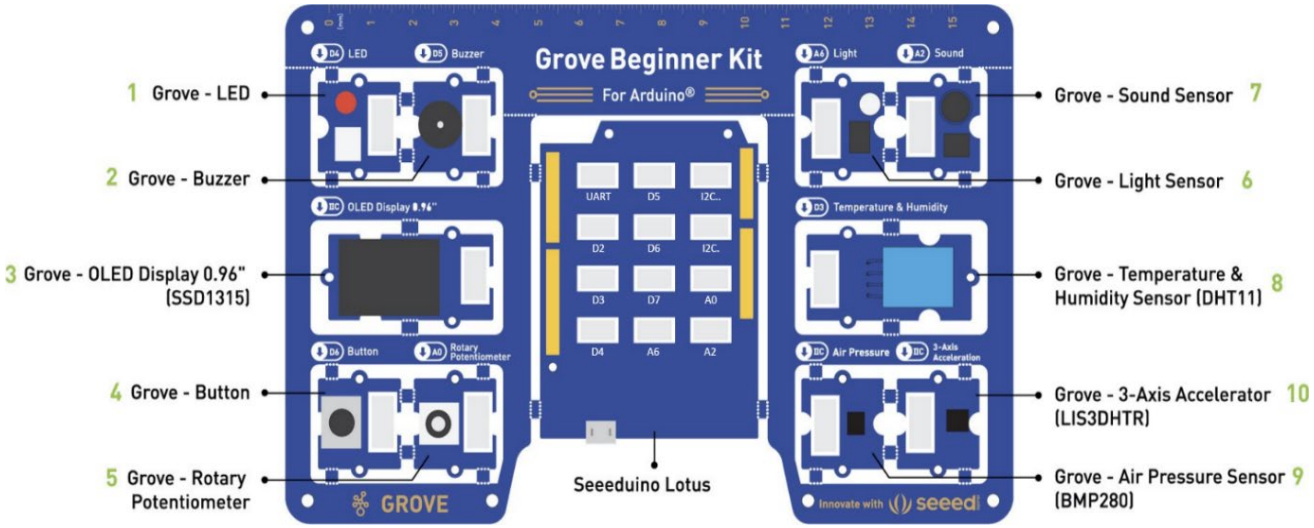


Figure 2 Grove Beginner Kit for Arduino showing all the peripherals (Adapted from bit.ly/2OSLrkZ)

Plug and Play Demo

The Grove Beginner Kit for Arduino comes with a plug and play demo.

*NOTE: If your board no longer has this demo, it can be re installed. The instructions for this are on p 15 (**Installing and Uploading the Plug and Play Demo**)*

To run the demo, plug the board into your computer using the USB cable found in one of the side compartments. The OLED display should light up with one of the sensor demos on display.

To move between the sensor demos use the button and the rotary potentiometer. *Figure 3* shows the controls for the demo.

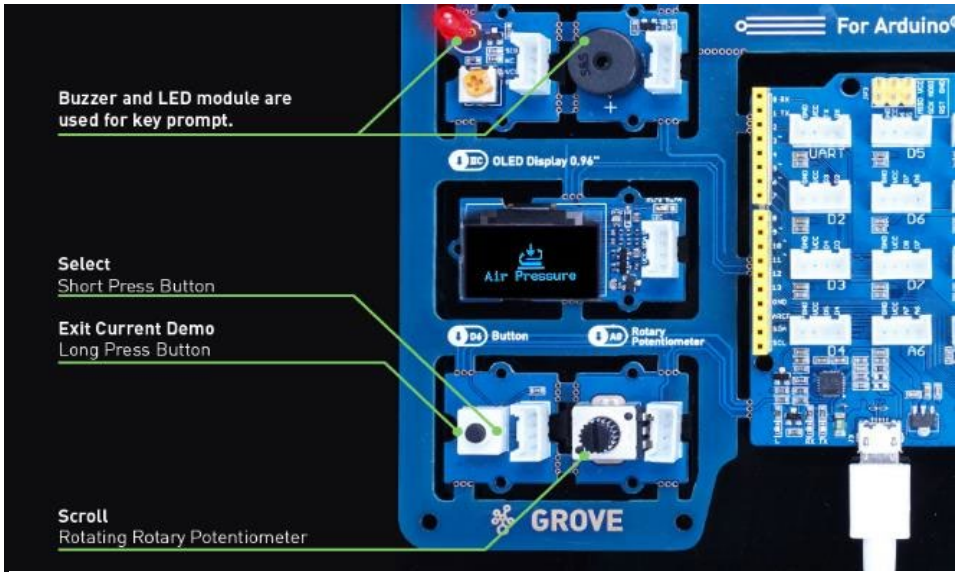


Figure 3 Controls for the demo

The sensors in the demo are:

- **Acceleration** : move the board in 3 axes to show this working
- **Air Pressure**
- **Temperature & Humidity** : place a hand over the sensors to change these readings
- **Sound** : clap over the sensor
- **Light** : use a phone torch on the sensor

Try out all the demo features to understand how the sensors work.

Arduino IDE – setting up and interface

Important Note: Please complete the steps to install the appropriate Arduino IDE for your operating system detailed in the "Software Installation" document before starting.

The Arduino IDE is where the code is written and uploaded to the Arduino board.

You have two options:

- Using the Desktop Arduino IDE
- Using the Duino App website

Using the Desktop Arduino IDE

Create a new sketch (File → New Sketch). Figure 4 shows a new sketch and labels the important features of the IDE.

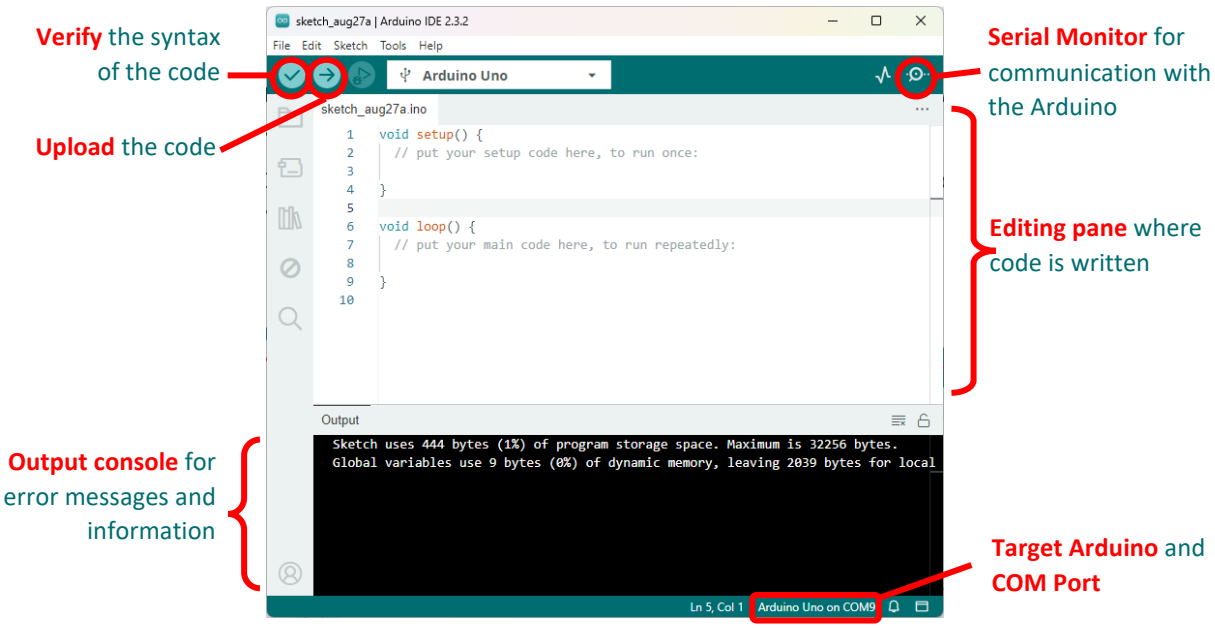


Figure 4 Desktop Arduino IDE Interface

Using the Duino App website

Add in the following code to your sketch as shown in **Error! Reference source not found.** before starting the exercises. **Error! Reference source not found.** also shows all the important features that will be referred to in the following exercises.

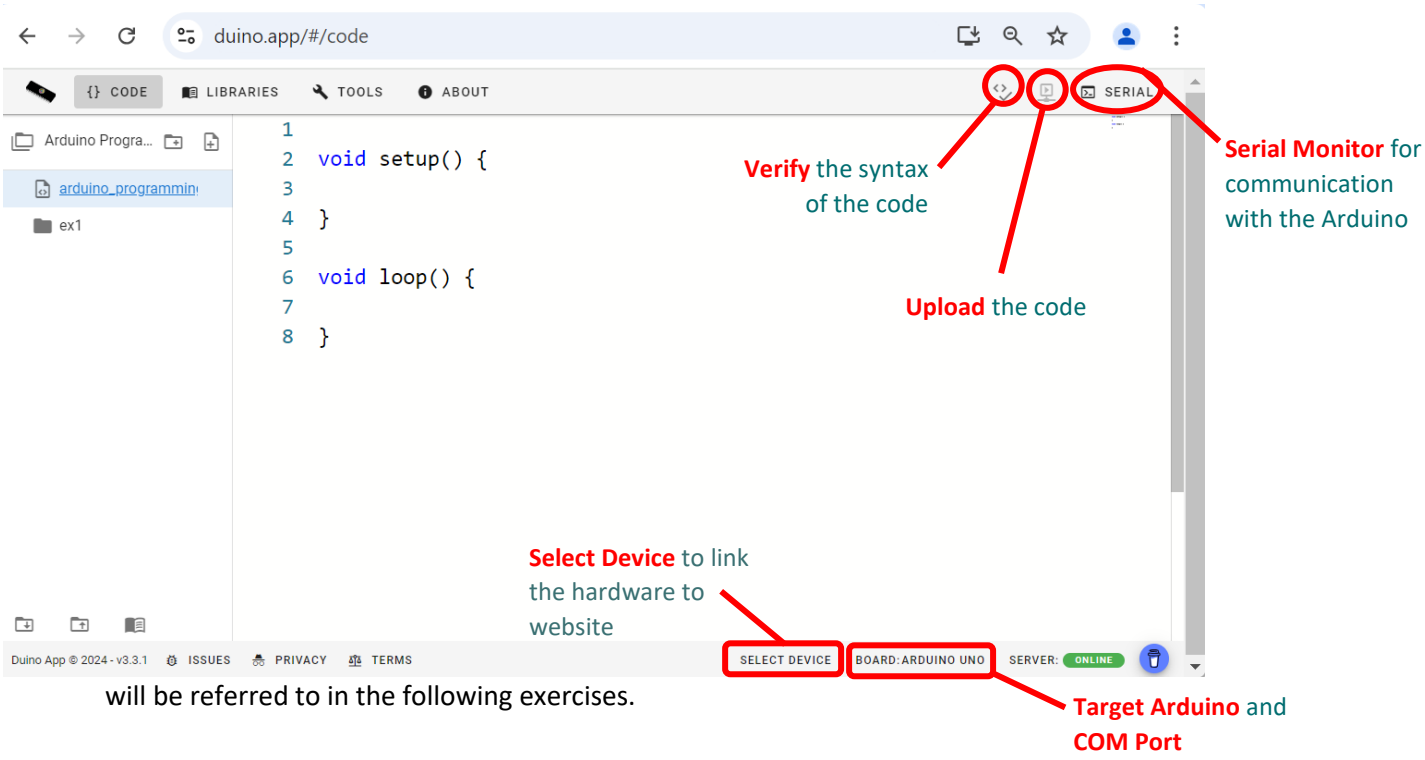


Figure 5 Duino App website interface

Explaining setup() and loop()

The `setup()` function describes what the Arduino needs to do once, for example, initialise a sensor or start a serial communication protocol to communicate with a computer or peripheral.

The `loop()` function describes the things you want the Arduino to do repeatedly, in a loop. For example, measure temperature, output a sound on the buzzer or display something on a screen. This is where most of your code will go and can be considered the main program of the Arduino.

Debugging and Errors

You can use the *verify* button in the IDE to check if you have written your code correctly and the *upload* button to upload the finished code to your board. If there are *errors* in your code, the IDE will warn you and stop you from uploading it. You will instead need to work out the cause(s) of the errors and "debug" your code. Some common causes of errors/bugs are: spelling errors, missing semi-colon(s) (;) at the end of code statements or undefined variables but there are many more. Debugging can take time and patience is key!

Note: errors will be printed at the bottom of your IDE, usually in red or orange to highlight that there is an issue. However, the IDE does not always point you to the exact line of the code where the error exists, and you may have to look through your code several times to find the source of the error.

Exercises

Exercise 1 – Blinking an LED

Blinking an LED is a simple program that blinks a light on and off and will introduce you to the development environment. It is also a good test to check your connection to the hardware as well as the hardware itself.

Step 1

In your empty sketch, *before* the `setup()` function, define the name and pin number of the LED you will use (in our case, the Grove kits' red LED is connected to pin D4, and is written on the top left of the Grove kit board). Therefore, we will define our LED pin to be digital pin number 4:

```
const int ledPin = 4;  //Define the name and pin number of the LED to blink
```

Step 2

In the `setup()` function (the code that runs only once) we need to declare whether the pin in question should be an Input or an Output. In our case, we are going to drive the LED from the Grove kit, which means our pin should be defined as an Output. We can do this with a function called `pinMode()` as shown below:

```
void setup() {
  pinMode(ledPin, OUTPUT);  // Initialize the ledPin as an Output.
}
```

Step 3

Next, in the `loop()` function we will define what we want our LED to do. This function will run through the code in a loop, forever. Since we want to blink the LED we need to decide when it should turn on and off and for how long it should stay in each state. We can implement this using a function called `digitalWrite()` which either turns the LED ON by outputting a High voltage to our LED pin (i.e. 5V or a 1 state) or a Low voltage (i.e. 0V or 0 state). To keep the LED in a state for some period of time we use the `delay()` function. This function tells the program to wait for a certain number of milliseconds before moving on to the next line of code. These two functions result in the following code:

```
void loop() {
  digitalWrite(ledPin, HIGH);  // Turn the LED on (High, 1, or 5V).
  delay(250);                  // Wait for 250 milliseconds.
  digitalWrite(ledPin, LOW);   // Turn the LED off (Low, 0, or 0V).
  delay(250);                  // Wait for 250 milliseconds.
}
```

Step 4

Once you have completed the above code click the verify button and sort through any errors you encounter. Once you are clear of errors, you're ready to upload your code.

Plug in the Arduino board to the USB port.

If using the **Arduino IDE**, click the drop-down menu in the top bar as shown in Figure 6. From here choose "Select other board and port…".

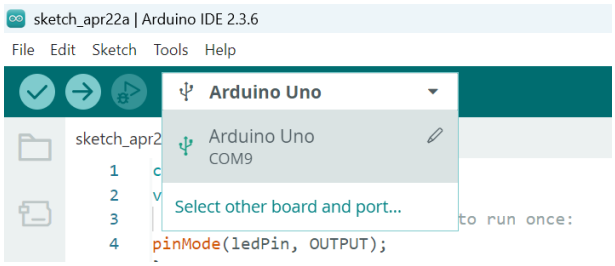


Figure 6 Arduino IDE – drop down menu

In the window that pops up make sure “Arduino Uno” is selected in the left hand box and the right COM port is selected in the right hand port as in Figure 7.

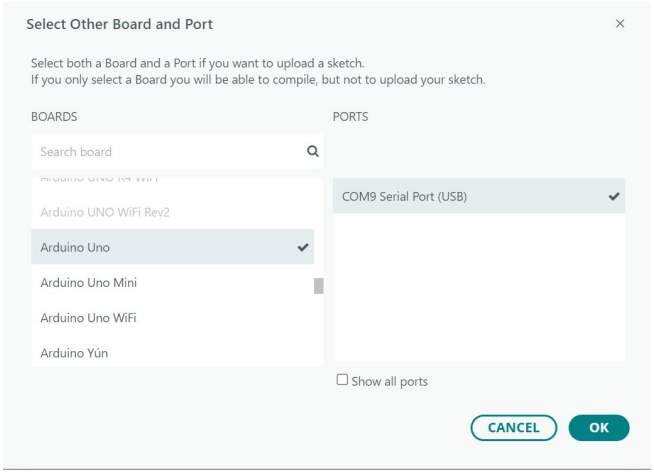


Figure 7 Arduino IDE - Select board and port

OR if using **the Duino App**, click on “select device” at the bottom of the page. This will bring up a window with the connections. Click on the connection as shown in Figure 8 and then click “Connect”.

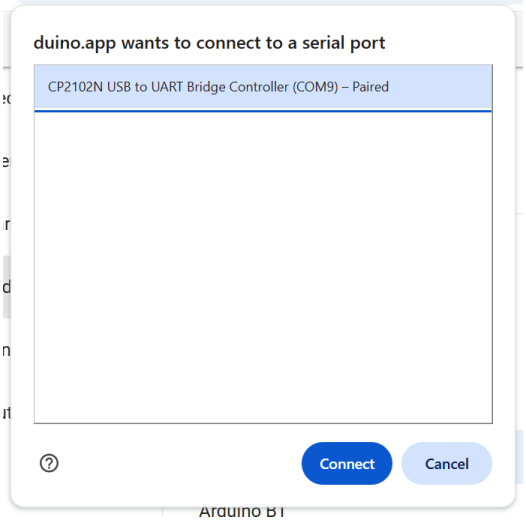


Figure 8 Duino App - connections window

At the bottom right of the page, click on “Board” and choose “Arduino Uno”.

Upload the code to the Arduino by clicking the icon in the top right-hand corner.

If you see your LED blink, congratulations! You’ve written your first working Arduino code.

Exercise 2 – Buzzing the Buzzer

In this exercise you will learn how to use the buzzer on the Grove kit (next to the red LED). This code will play a tone on the buzzer that lasts one second and then pauses for one second before repeating.

Step 1

Launch a new sketch. Before the `setup()` function, define the name and pin number the buzzer is connected to (D5 on the Grove kit) and the frequency of the tone you wish to play:

```
const int buzzerPin = 5;  //Define the name and pin number of the buzzer
const int toneFrequency = 200;  //Frequency of tone in Hz
```

Step 2

In the `setup()` function declare whether the buzzer is an input or an output. To play a tone on the buzzer, the pin needs to be defined as an output:

```
void setup() {
  pinMode(buzzerPin, OUTPUT);  // Initialize the buzzerPin as an output.
}
```

Step 3

In the `loop()` function, define the actions for the buzzer, starting with a tone of 200 Hz for 1s, then turn the buzzer off for 1s, and repeat. To play a tone on the buzzer the Arduino language provides two functions: `tone()` and `noTone()`. They both need the pin number that the buzzer is connected to as arguments, but the `tone()` function also needs the frequency of the tone to be played:

```
void loop() {
  tone(buzzerPin, toneFrequency);  // Play a tone of 200 Hz on the buzzer
  delay(1000);                     // Wait for 1000 milliseconds or 1s.
  noTone(buzzerPin);               // Turn the buzzer off.
  delay(1000);                     // Wait for 1000 milliseconds or 1s.
}
```

Step 4

Once you have completed the above code click the verify button and sort through any errors you encounter before uploading your code.

Try changing the frequency and duration of the tone to explore the different sounds you can make.

Tutorial - Serial Communication and Talking to your Computer

Serial communication works by converting information to a stream of bits, which are then sent between two devices over one or more wires, allowing interaction with more complex devices such as a laptop or sensors such as a digital accelerometer.

The Serial family of functions in Arduino allow use of the UART communication protocol to communicate with a computer.

Serial protocol functions :-

`Serial.begin(9600)`- Initialises different peripherals and communication protocols. 9600 is the 'baud rate' of the connection. It defines how fast the data is to be sent.

`Serial.print()` – Sends the text in the parentheses to the computer.

`Serial.println()` – Sends the text in the parentheses to the computer and also starts a new line.

To test this functionality, type the below code into a new sketch and upload it.

```
void setup() {
  Serial.begin(9600);               // Begin the serial communication.
  Serial.println("Hello, World!");  // Send a message to the computer.
  //You can add more print statements here
}

void loop() {} // For now, our loop() function does nothing.
```

If using the Arduino IDE, go to **Tools > Serial Monitor** (or click the looking glass icon in the top-right). OR if using Duino, after uploading the code, click on the monitor tab and change the baud to 9600 (in bottom right hand corner).

This will open a new window showing the message sent from the board (see Figure 9).

Try sending different messages using both Serial.print() and Serial.println().

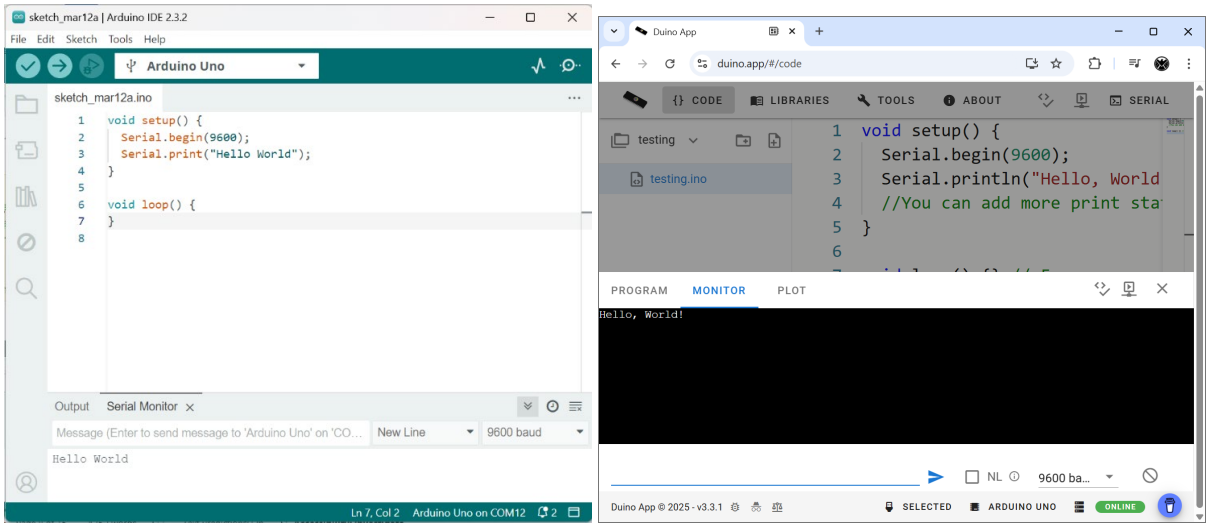


Figure 9 A preview of the Serial Monitor tool showing the "Hello, World!" message that was sent over USB from the Arduino

Exercise 3 – The Accelerometer

In this exercise you will learn to how to read data from a 3-axis accelerometer and plot it as a graph on the Arduino Serial Plotter. An accelerometer is a device that measures the rate of change of velocity of a body. For example, an accelerometer at rest on the surface of Earth will measure the acceleration due to Earth's gravity as 1 g (gravitational force equivalent) or approximately 9.81 m/s^2. Using the accelerometer on the Grove kit, we can measure acceleration in three axes.

Step 1

Before the `setup()` function, include the UKESF library (which will handle some of the accelerometer functions) and define an accelerometer variable called "myAccelerometer":

Note: Make sure you followed the instructions in the software installation instructions to install the UKESF library for the software you are using.

```
#include <UkesfSixthFormers.h> // Include the UKESF library of functions

Accelerometer myAccelerometer;  // Create an instance of the accelerometer
```

Step 2

Initialise serial communication and the accelerometer:

```
void setup() {
  Serial.begin(9600);     // Begin the serial communication.
  myAccelerometer.begin();  // Begin the accelerometer.
}
```

Step 3

In the `loop()` function, read the accelerometer values and print them to the serial monitor. Note that floating point variables or `float` must be used because the acceleration is given as a decimal number.

```
void loop() {
  float x = myAccelerometer.readX();  // Read x-axis acceleration
  float y = myAccelerometer.readY();  // Read y-axis acceleration
  float z = myAccelerometer.readZ();  // Read z-axis acceleration
  Serial.print(x);
  Serial.print(" ");
  Serial.print(y);
  Serial.print(" ");
  Serial.println(z);
  delay(10); // Delay the program by 10ms for stability when looping
}
```

Step 4

There are two ways to view the data from the accelerometer. The first is using the serial monitor and the second is the *Serial Plotter*. You can access the plotter from **Tools > Serial Plotter.** This is a useful tool that plots the data in graph form. Try tilting and turning your board to identify each of the three axes of the accelerometer.

Exercise 4 – Combining it all

In this exercise you will combine what you have learned in the previous three exercises into one program, implementing a tilt sensor with a warning light and sound. Specifically, the program will:

- Read data from the accelerometer,
- Calculate the pitch and roll (angles) of your board in degrees, based on that data,
- If the roll (or pitch) exceeds 20 degrees, light the LED and sound the buzzer.

The examples of code given below include only the comments of what the program should include. You will fill them in, using the syntax and pin numbers given **exercises 1-3.**

Step 1

Include the UKESF library to access the accelerometer and define the pin numbers and variables the program will use later. Fill in the missing lines of code before each comment below:

```
// Include the UKESF library of functions

// Define the name and pin number of the LED
// Define the name and pin number of the buzzer
// Frequency of tone in Hz
// Create an instance of the accelerometer
```

Step 2

Initialise the LED, buzzer, serial communication and accelerometer.

```
void setup() {
  // Initialize the ledPin as an output.
  // Initialize the buzzerPin as an output.
  // Begin the serial communication.
  // Begin the accelerometer.
}
```

Step 3

Implement the main program in the `loop()` function:

1) Read the accelerometer data
2) Calculate the roll and pitch
3) Light the LED and sound the buzzer IF the roll is > 20 degrees.

Approach each of these tasks in turn.

1) Read the accelerometer data

Look back to exercise 3 for the code used to read data from the accelerometer and print it to the serial monitor. Although the program does not require printing to Serial, it is a good idea to do so for debugging purposes, using the serial monitor after uploading our code to make sure the data from the accelerometer is as we would expect.

```
void loop() {
  // Read x-axis acceleration
  // Read y-axis acceleration
  // Read z-axis acceleration

  // Print the value to serial for debugging help
}
```

2) Calculating roll and pitch from accelerometer data.

Pitch, roll and yaw are rotational forces (illustrated in Figure 10) and deriving them as angles from accelerometer data is complicated. The links below give good explanations of how a 3-axis accelerometer can be used to derive pitch and roll.

- DF Robot tutorial with quick derivation: https://wiki.dfrobot.com/How_to_Use_a_Three-Axis_Accelerometer_for_Tilt_Sensing
- Application note from NXP with more rigorous derivation: https://www.nxp.com/files-static/sensors/doc/app_note/AN3461.pdf

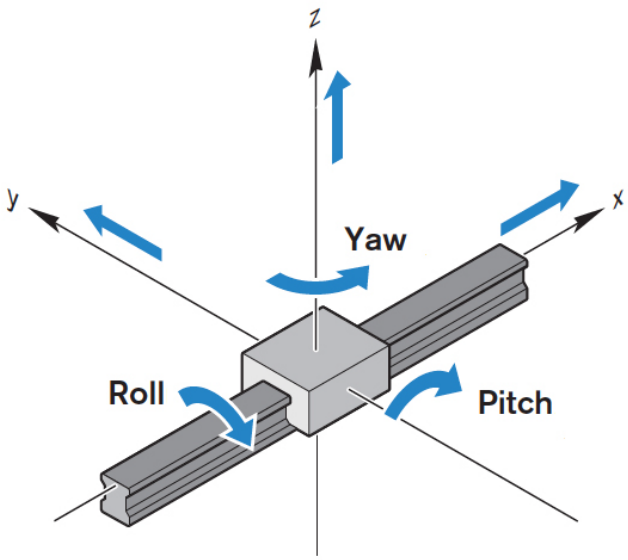


Figure 10 An illustration of pitch, roll and yaw of a linear system, in this case a rigid beam.

It is sensible to read through the information given in the provided links to get an understanding of on the derivation of roll and pitch angles. The final equations are given below and require the use of pi, which in Arduino is written as “M_PI”.

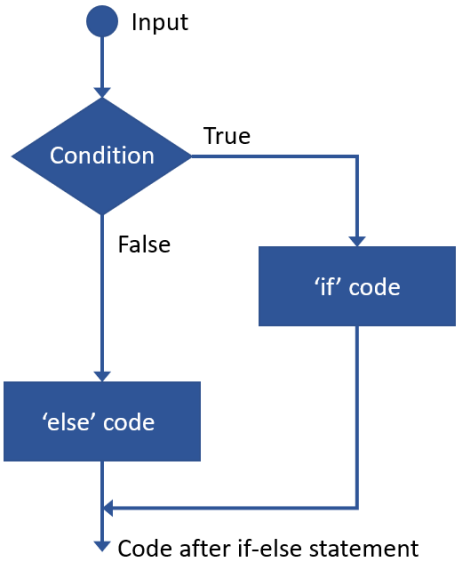
```
void loop() {
  // Read x-axis acceleration
  // Read y-axis acceleration
  // Read z-axis acceleration

  // Print the value to serial for debugging help

  //Calculate roll and pitch based on accelerometer data
  //See provided links for derivation.
  float roll  = (atan2(-y, z)*180.0)/M_PI;
  float pitch = (atan2(x, sqrt(y*y + z*z))*180.0)/M_PI;
}
```

3) Light the LED and sound the buzzer if the roll is > 20 degrees.

To turn the LED on and sound the buzzer if the roll angle is > 20 degrees in either direction, the absolute value function `abs()` must be used. To do this, use a conditional if-else statement to implement the logical sequence illustrated by the flow diagram in Figure 11. The if-else statement tests an input variable against a condition (i.e., is the angle > 20 degrees?) and takes the True or False path depending on the result. When a condition is True, only the code inside the curly brackets enclosing the 'if' statement is executed and when it is False, the 'else' statement is executed.




```
if(condition is True){
 // Execute this code (condition is True)
}
else{
 // Execute this code (condition is False)
}
```

Note! The `condition` in the if statement can be any logical test, for example:

To check for negative temperatures, you could write:

```
if(temp < 0)
```

To check for an angle of 45 degrees, you could write:

```
if(angle == 45)
```

Figure 11 Flow diagram of an if-else statement in Arduino and the corresponding code. Depending on if the condition is True or False different code gets executed and the rest is skipped.

Complete the code template below, based on your previous work.

```
void loop() {
  // Read x-axis acceleration
  // Read y-axis acceleration
  // Read z-axis acceleration

  // Print the value to Serial for debugging help

  //Calculate roll and pitch based on accelerometer data
  //See provided links for derivation.
  float roll  = (atan2(-y, z)*180.0)/M_PI;
  float pitch = (atan2(x, sqrt(y*y + z*z))*180.0)/M_PI;

  //if-else statement to check if the roll is > 20 deg
  //Note the use of the absolute value abs()
  if(abs(roll) > 20){
    // Turn the LED on (High, 1, or 5V).
    // Play a tone of 200 Hz on the buzzer
  }
  else{
    // Turn the LED off (Low, 0, or 0V).
    // Turn the buzzer off.
    }

  delay(10); // Delay the program by 10ms for stability when looping
}
```

Experiment by changing your code to light the LED and sound the buzzer if the pitch is > 20 degrees

Open-Ended Design Challenge

Well done on completing the introduction to the Grove kit and Arduino IDE! You now understand the basics and can explore your own designs using the board. Be creative and have fun!

A good way to find inspiration is to search for ideas online. There are thousands of previous Arduino projects on the web and often you can find something related to what you want to achieve. For example, if you would like to play a song using the buzzer, search for "Arduino play tune on buzzer" and you will find many examples. It can be tricky to discern which example is best suited to your needs or how someone else's code works. Often this is a process of trial and error. You could also combine parts of several different codes to make your own program. You could, for example:

- Play tunes using the buzzer. See this Arduino tutorial.

- Display custom text on the OLED screen based on sensor input. For example:
    - Display text on the screen for different orientations of the board using the accelerometer.

- Use the microphone to pick up sound and inspect the audio waveform using the Serial Plotter. Set a threshold to turn on the LED if the sound level is above the threshold.

- Use the push button to increment and display the number of times the button has been pressed on the OLED screen. Search online for "*Arduino number of button presses display*" for tips.

Final Words and Further Resources

Creativity and imagination lie at the heart of Electronics and there are many more resources to explore. As well as searching online for project ideas and, there is also a large community of forums and hubs where you can ask questions and get help if you need it. Here are some suggestions:

- The Arduino homepage.
- The Arduino project hub.
- Seeedstudio's guide for the Grove kit: Grove-Beginner-Kit-For-ArduinoPDF.
- Seeedstudio's article with a collection of tutorials: Arduino Project Roundup.
- UKESF videos for the Grove kit: Introduction and Level meter project.
- Arduino projects on Instructables: https://www.instructables.com/Arduino-Projects/
- Search for "*your project idea + Arduino*" and you'll find lots of resources!

Good luck with your future Electronics projects!

Installing the Plug & Play demo

To use the original plug and play demo, follow the instructions below to download and install the zip file.

1) Go to: https://wiki.seeedstudio.com/Grove-Beginner-Kit-For-Arduino/#resources

2) Click on “5. Initial Arduino Firmware Demo” to download a zip file of the code and libraries needed to run the initial demo code.

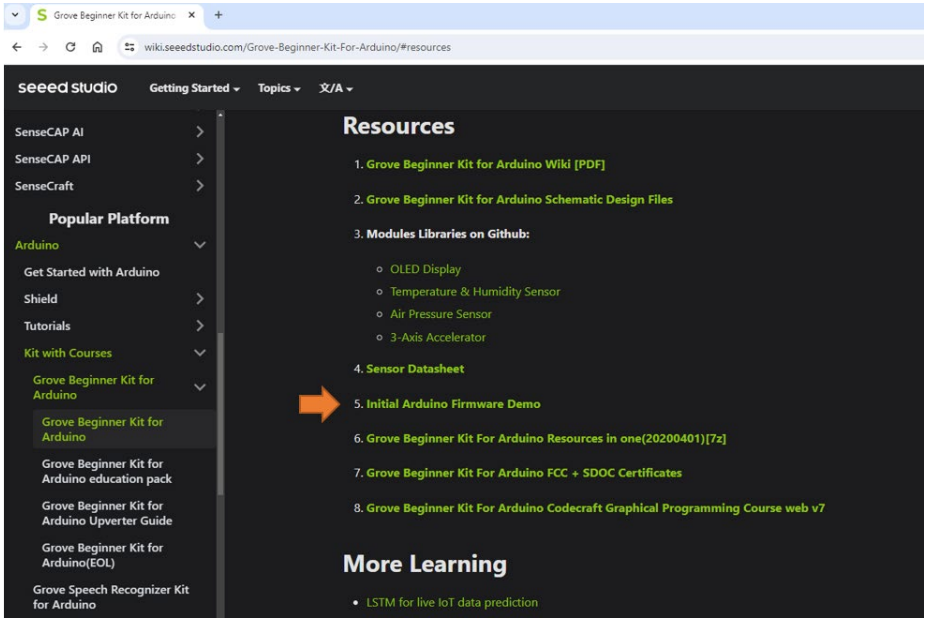


Figure 12 On the Seeed Arduino Grove kit wiki click the “Initial Arduino Firmware Demo” link to download the zip file which contains the initial code and libraries.

3) Open your Arduino IDE and go to File → Preferences to check your sketchbook location

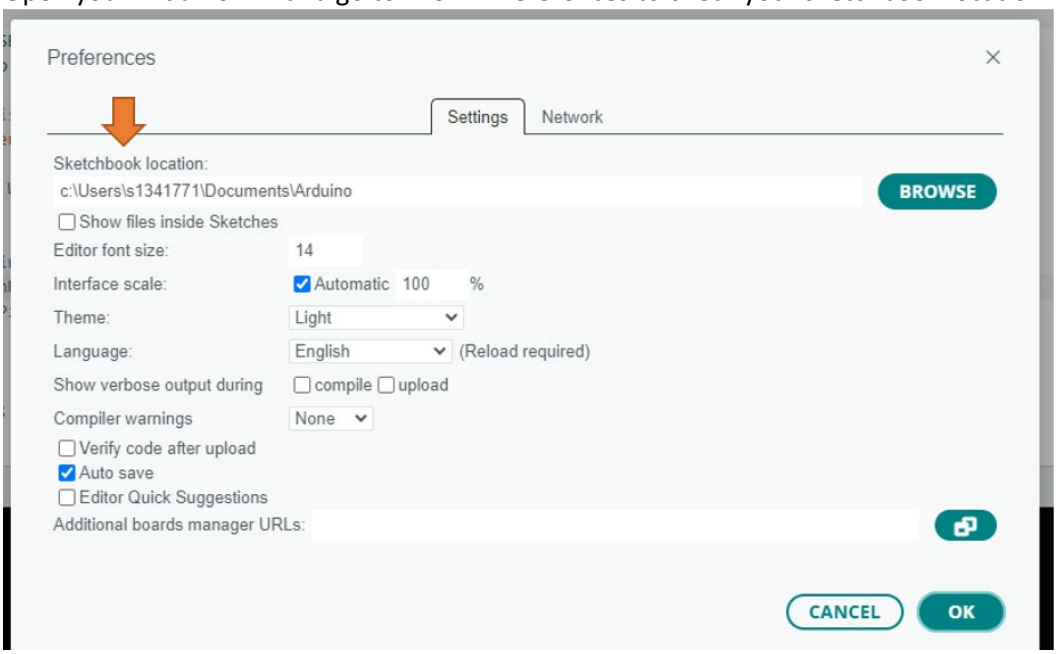


Figure 13 Sketchbook location: In your Arduino IDE, File ⍰ Preferences, check you Sketchbook location (you will need to unzip the firmware zip file here).

4) Open the Sketchbook folder in a file explorer window.
5) Move the downloaded zip file from step 3 into this sketchbook folder and unzip it there. Once done, you should have a folder called “Grove_Starter_Kit”.
    a. If the “Grove_Starter_Kit” folder is still inside another folder, move it out so that your sketchbook location looks like Figure 11.

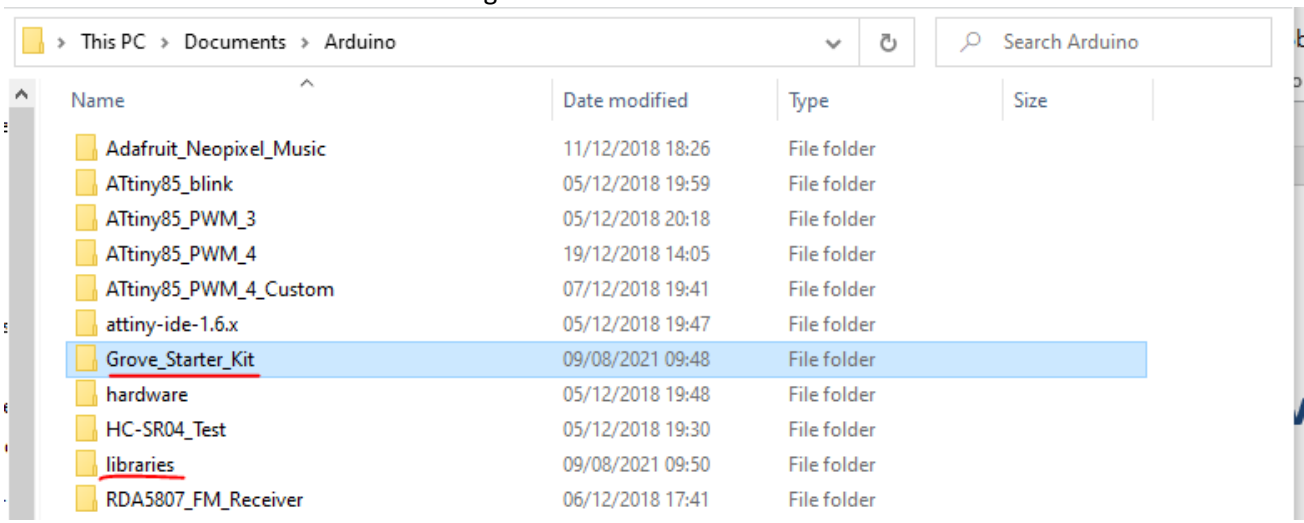


Figure 14 Sketchbook folder: In your Sketchbook folder you should now have the unzipped folder called “Grove_Starter_Kit” and a “libraries” folder (which is where all your Arduino libraries like the UKESF-Sixth-Formers library lives).

6) Your sketchbook folder also includes a “libraries” folder from before, as highlighted in Figure 11.
7) From the unzipped “Grove_Starter_Kit” folder, copy the highlighted folders in Figure 12. across to your Sketchbook “libraries” folder (Note that you may already have a “U8g2” library folder from before, in which case you can skip that one).

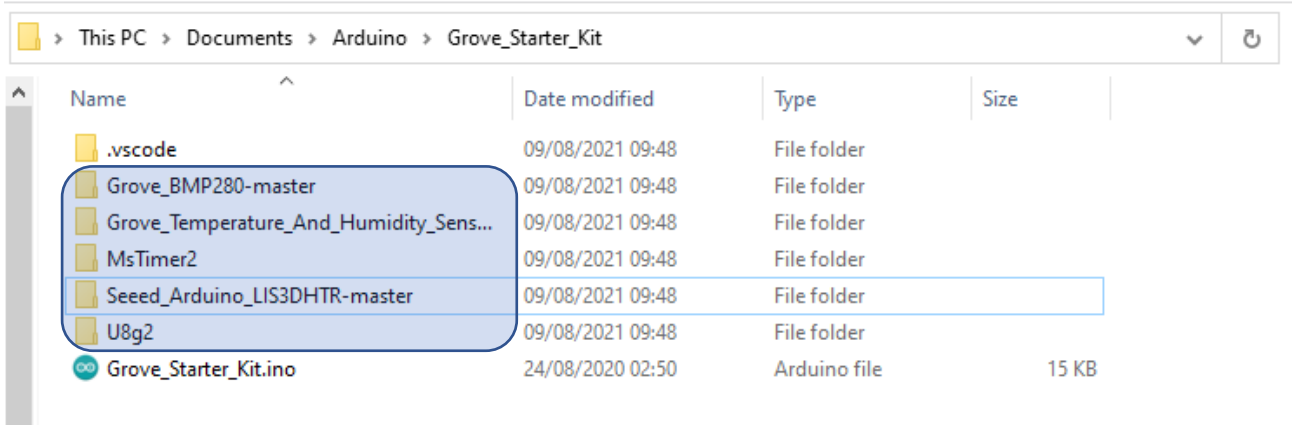


Figure 15 Copy the libraries across: included in the “Grove_Starter_Kit” folder are five libraries we need to copy into the Arduino Sketchbook “libraries” folder (Note: you may already have a U8g2 library from before, in which case you can skip that one or overwrite

8) After you have copied the folders across, you can double click the “Grove_Starter_Kit.ino” file to open the Arduino code for the initial example and upload it to your Grove Kit board.

If you get any errors when trying to upload, please refer to the UKESF Guide and check that you have chosen the correct COM port, Arduino Uno board type and AVRISP MKII programmer, also check that the libraries you copied over from the zipped folder to your libraries folder were successful.

You can now edit and upload the original Example code for the board.

About the UK Electronics Skills Foundation

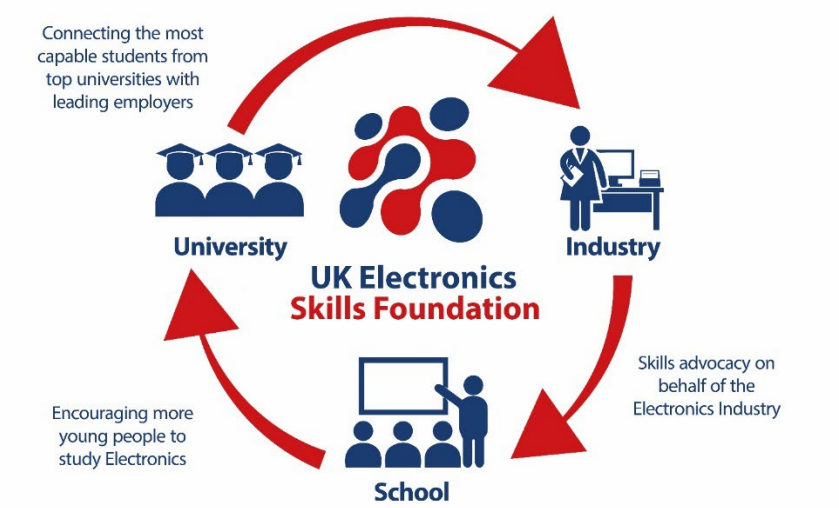
The UKESF exists to encourage the pursuit of careers in electronics and support industry in developing excellence in the sector. We do this through:

- Encouraging more young people to study Electronics by providing resources and experiences to develop their interest
- Connecting the most capable students from leading universities with employers and supporting their professional development to equip them with work-ready skills and experience
- Skills advocacy on behalf of the Electronics industry, including apprenticeships and research
- Collaborating with industry partners and stakeholders to build relationships and secure the future pipeline with a community of Electronics Engineers.

We are an independent charitable foundation, established in 2010, at the nexus of an extensive network of partners and collaborators, including 30 universities and around 75 companies.

Registered charity number: SC043940

www.ukesf.org



"Moving beyond talk about the skills shortage to take positive action is what the UKESF is all about."

Stew Edmondson, CEO, UKESF