Girls into Electronics 2026
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? 2
The Grove Beginner Kit for Arduino 2
Arduino IDE – setting up and interface 4
- Using the Desktop Arduino IDE 4
- Using the Duino App website 5
- Explaining setup() and loop() 5
- Debugging and Errors 5

Exercises 6
- Exercise 1 – Blinking an LED 6
- Exercise 2 – Buzzing the Buzzer 8
- Tutorial - Serial Communication and Talking to your Computer 9
- Exercise 3 – The Accelerometer 10
- Exercise 4 – Combining it all 11

Open-Ended Design Challenge 15
Final Words and Further Resources 15
Installing the Plug & Play demo 16
About the UK Electronics Skills Foundation 18

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.

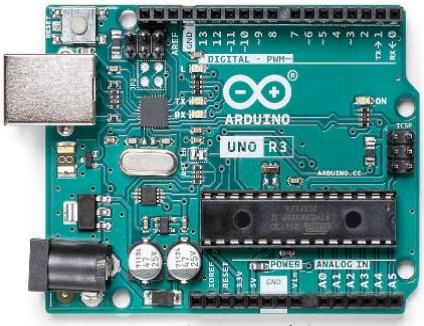


Figure 1 Arduino Uno

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.

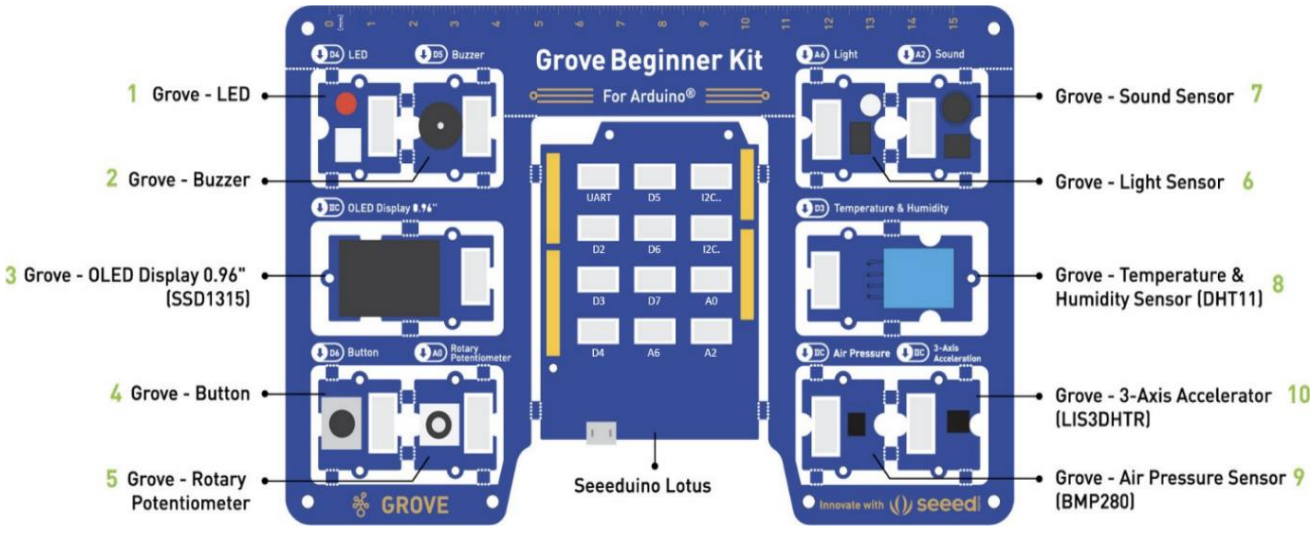


Figure 2 Grove Beginner Kit for Arduino showing all the peripherals (Adapted from bit.ly/2OSLrkZ)

Plug and Play Demo

The Grove Beginner Kit for Arduino comes with a plug and play demo. If you have a brand new kit then you can run the demo. If you don't have a new kit, you can skip this section and go onto the Arduino Software section to start to write your own code!

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.

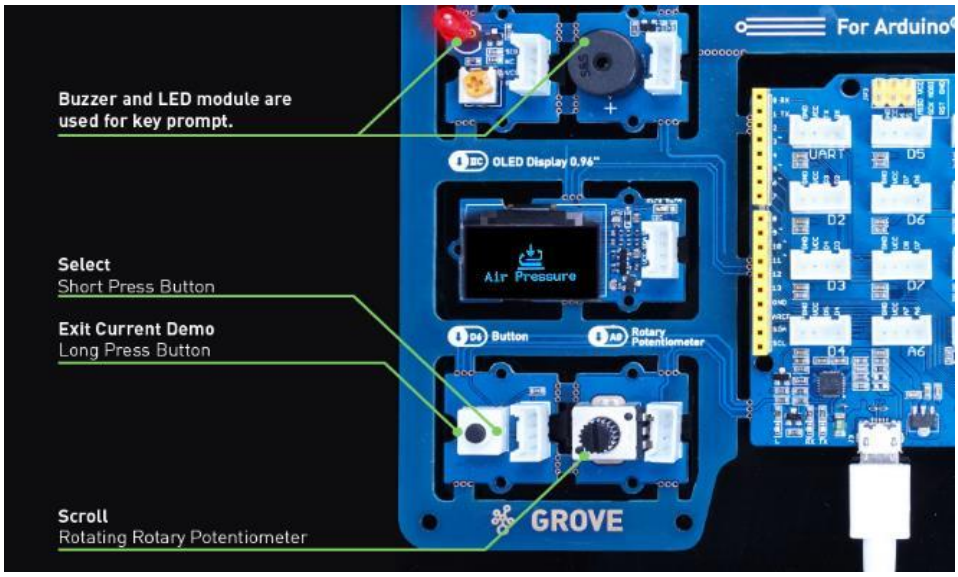


Figure 3 Controls for the demo

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 Software to code the Grove Beginner Kit for Arduino

There are two main software packages you can use to code the Grove Beginner Kit for Arduino.

We recommend using the **Duino.App**. This is a website so does not require any set up and should work on all operating systems.

The other software is the **Arduino Desktop IDE**(https://www.arduino.cc/en/software/). If you want to do a lot of coding with the Arduino then the **Arduino IDE** works better for more complicated projects. But this only works on Windows and Mac OS.

This guide will show you how to code in the Dunio.App, but all the code will work in any software designed for Arduinos.

Using the Duino App website

The Duino App can be used from most web browsers. Go to the website: https://duino.app/

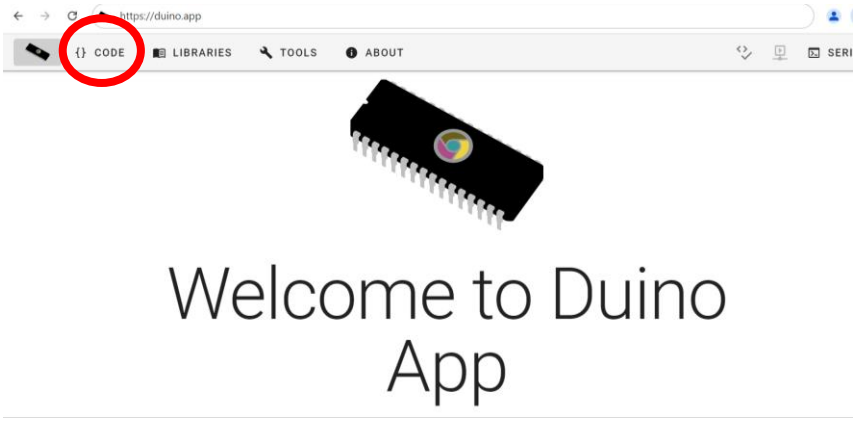


Figure 4 Duino App

To create a new project, click “Code” in the top bar of the webpage, then “Create New Project” as seen in Figure 5.

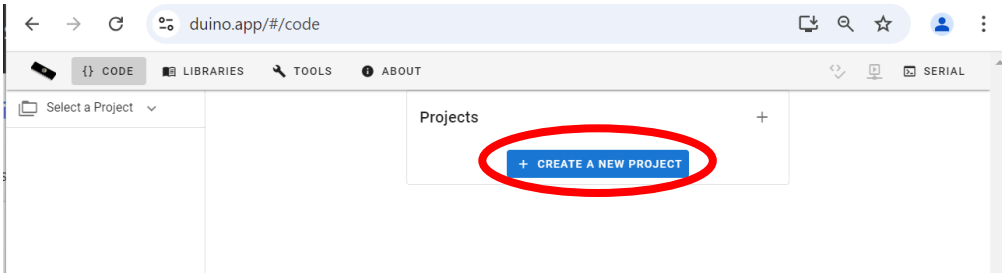


Figure 5 Duino App- Creating a new project

Then fill in a project name and click “Create”.

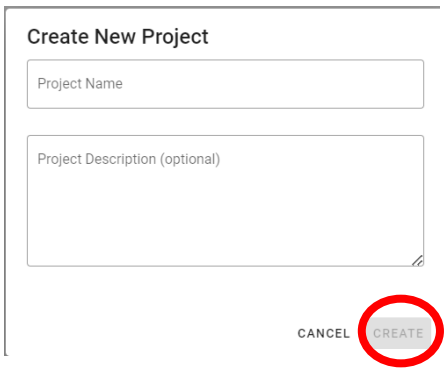


Figure 6 Duino App - Creating a new Project

Delete the standard code that appears, so you have a blank sketch ready to start.

Add in the following code to your sketch as shown in Figure 7 before starting the exercises. Figure 7 also shows all the important features that will be referred to in the following exercises.

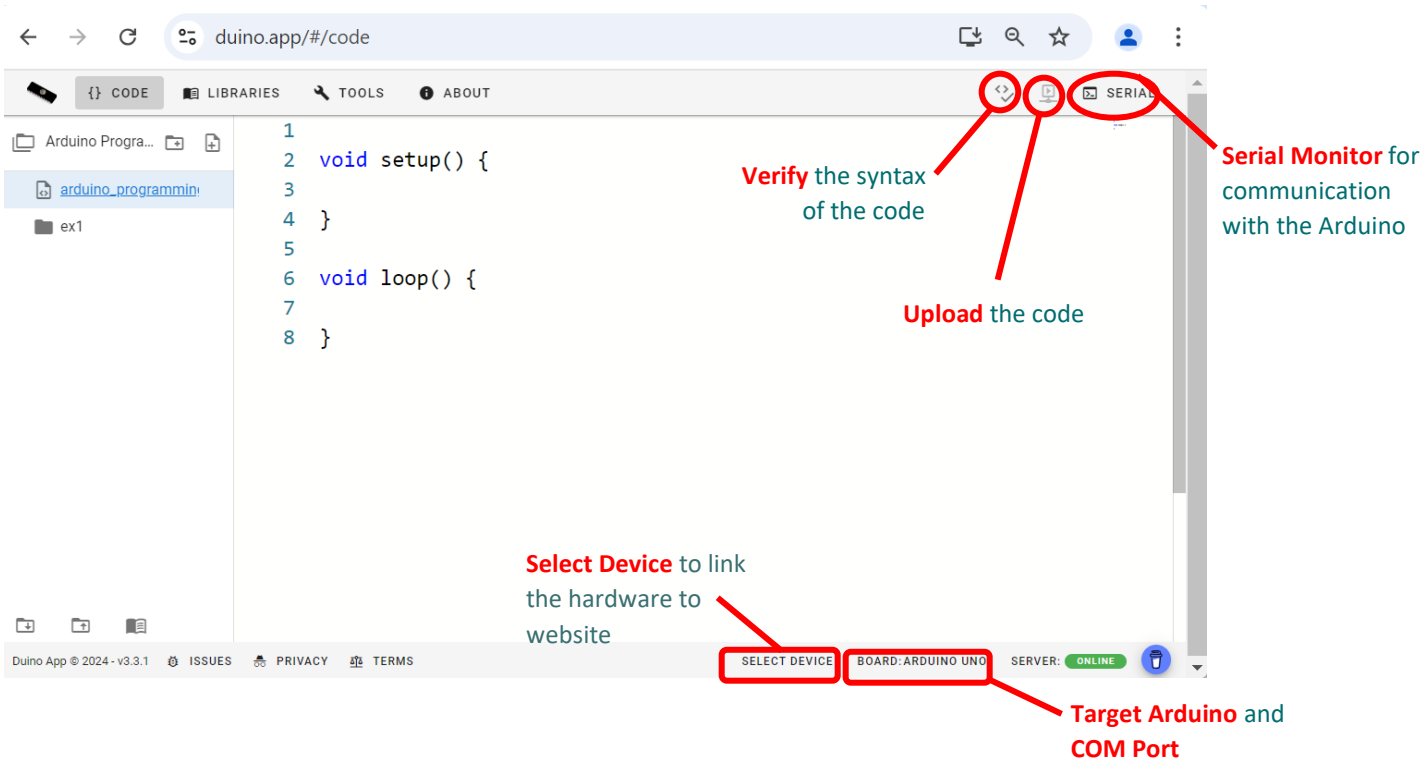


Figure 7 Duino App website interface

Explaining setup() and loop()

The setup() function describes what the Arduino needs to do once at the start of the program, 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 ***Check & Compile*** button in the IDE to check if you have written your code correctly and the ***Compile &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 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.

You are now ready to start the exercises!

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 (You don't need the "D" for the digital pins):

```
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. The LED should be defined as an Output. We can do this with a function called pinMode() as shown below:

```
void setup() {
  pinMode(ledPin, OUTPUT);  // Initialise 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 or a Low voltage.

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 Check & Compile 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.

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".

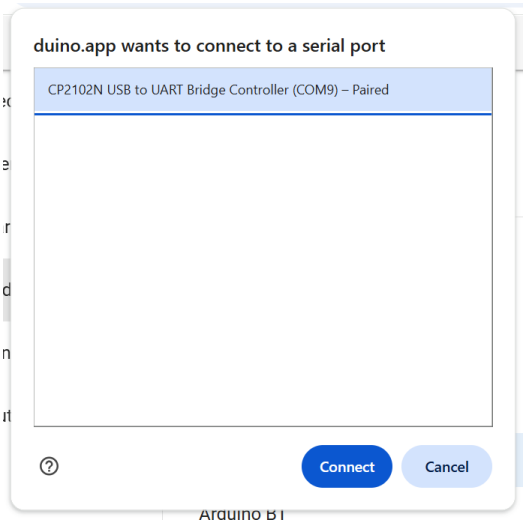


Figure 8 Duino App - connections window

Then, also in the bottom right of the page click on "Board" and choose "Arduino Uno".

Now upload the code to the Arduino by clicking the Compile & Upload button  in the top right-hand corner.

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

Your turn

1. Edit the code to change the amount of time the LED stays on.

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. You can either replace your previous code or create a new project.

Step 1

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:

```
int buzzerPin = 5;  //Define the name and pin number of the buzzer
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);  // Initialise 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 1 second (which we have to define as 1000 milliseconds), then turn the buzzer off for 1 second, 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 Check & Compile button and sort through any errors you encounter before uploading your code.

Your Turn

1. Edit the code to change the frequency and duration of the tone to explore the different sounds you can make.

Exercise 3 – Pressing button to Light up LED

In this exercise you will build on the code you wrote for exercise 1, changing it so the LED only lights up when you press the button. So, you will be using an input sensor – the button and an output- the LED. The button on the Grove kit is at pin D6.

Step 1

Before the setup() function, we will need to define the name and pin number of the LED, like we did in exercise 1, as well as defining the name and pin of the button and what state the button is in to start with. The states will be represented by "LOW" if off and "HIGH" if on.

```
int ledPin = 4; //Define the name and pin number of the LED
int buttonPin = 6; //Define the name and pin number of the button
int buttonState = LOW; //Define the state of the button to start with
```

Step 2

In the setup() function we need to declare whether the pin in question should be an Input or an Output. The LED will be an output again, but the button will be an input.

```
void setup() {
  pinMode(ledPin, OUTPUT); // Initialize the ledPin as an output.
  pinMode(buttonPin, INPUT); // Initialize the buttonPin as an input.
}
```

Step 3

Next, in the loop() function we will write code that checks if the button is on (HIGH) If it is then it will turn the LED on. Otherwise, it will turn the LED off. To do this, we will use an if-else statement.

```
void loop() {
  buttonState = digitalRead(buttonPin);//this checks the state of the button
  if (buttonState == HIGH) { //This is the condition you are checking
    digitalWrite(ledPin, HIGH); //The LED is turned on if the button is on
  } else {
    digitalWrite(ledPin, LOW); //The LED is turned off if the button is off
  }
}
```

Step 4

Once you have completed the above code click the Check & Compile button and sort through any errors you encounter before uploading your code. Press the button and see the LED turn on.

Your Turn

1. Edit the code so that the buzzer sounds when you press the button.
2. Edit the code so that the button controls both the LED and the buzzer.
3. Edit the code so that frequency of the buzzer changes each time you press the button.

Exercise 4 – Sound sensitive LED

In this exercise we will make the LED come on when it detects a noise. For this we will be using another input sensor, the sound sensor. On the Grove kit the sound sensor is pin A2. This means it is an analogue pin. We will need to refer to the pin as "A2".

Step 1

Before the setup() function, we will need to define the name and pin number of the LED, like we did in exercise 1, as well as defining the name and pin of the sound sensor.

```
int ledPin = 4; //Define the name and pin number of the LED
int soundPin = A2; //Define the name and pin number of the sound sensor
```

Step 2

In the setup() function we need to declare whether the pin in question should be an Input or an Output. The LED will be an output again, and the sound sensor will be an input.

```
void setup() {
  pinMode(ledPin, OUTPUT); // Initialize the ledPin as an output.
  pinMode(soundPin, INPUT); // Initialize the soundPin as an input.
}
```

Step 3

Next, in the loop() function we will write code that checks if the sound is over a certain threshold. If it is then it will turn the LED on. Otherwise, it will turn the LED off. To do this, we will use an if-else statement.

```
void loop() {
    int soundState = analogRead(soundPin); //This checks the sound level
    if (soundState > 400) { //This test if sound value is above 400
    digitalWrite(ledPin, HIGH); //The LED is turned on if the button is on
  } else {
    digitalWrite(ledPin, LOW); //The LED is turned off if the button is off
  }
}
```

Step 4

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

Your Turn

1. Edit the code to change the sensitivity of the sound sensor so it only detects certain levels of sound. Can it detect when you are speaking?
2. Edit the code so that the light sensor controls whether the buzzer sounds. When the light sensor detects a bright light like a mobile torch light, then the buzzer sounds.

About the UK Electronics Skills Foundation

Through engagement with schools, universities and industry, it is our mission to encourage more young people to study Electronics and to pursue careers in the sector.

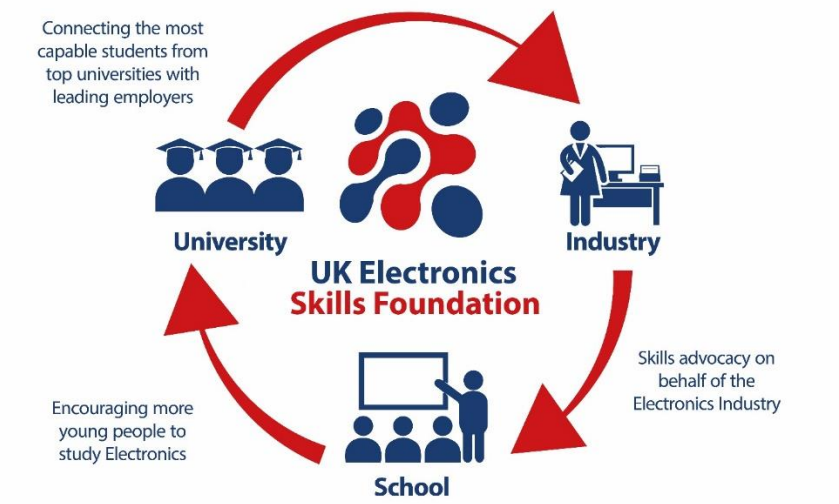
Electronics Engineers make a critical contribution to virtually all areas of life, from AI and digitalisation to green energy, healthcare, communications, manufacturing and more. The UK has a long heritage of technological innovation and has a worldclass Electronics sector. We are adept at creating and growing successful companies that design, make and sell products and services across all technological fields - and these companies will play an important role in solving some of the biggest challenges facing society today.

However, the demand for capable, employable Electronics Engineers and designers is currently outstripping supply. The only way we can sustainably support our industry to grow, is to increase the number of young people pursuing a career in the sector, and equipping them with Electronics skills for the future.

As an independent, UK based charity, we're working to address the skills gap in Electronics by:

- raising awareness,
- promoting interest in young people,
- supporting the development of those who choose Electronics, and
- building relationships to ensure a thriving sector.

Our mission can only be achieved by working collaboratively, and to date, we have worked with more than 100 employers from across the industry, 30 of the UK's leading universities, and over 900 schools.



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

Find out more: **ukesf.org**