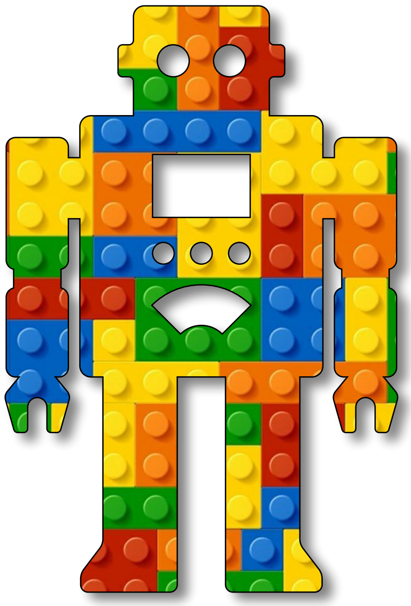
Intelligent Machines Module A Unit #1 the hardware



Contents

Important information

Important Notice: The board we will be using in this series of modules and units is the Arduino Nano 33 BLE. It is the newer version 2. You can still get hold of version 1, but they are not readily available anymore. The difference between the two is the IMU. This is where it measures acceleration, rotation, and the magnetic field. The version 1 uses the LSM9DS1 module, but version 2 uses the BMI270 and BMM150 modules. They do the same job. Also, get the one with pins already soldered.

Devices: These are often microcontrollers that are embedded in a smart device. A common example may be something like a smart speaker. An example of a microcontroller is the Arduino Nano 33 BLE. Some of these devices can connect to Wi-Fi and transmit data from sensors.

Sensors: You can add a very wide variety of sensors. Whether motion, sound, light, etc.

Connectivity: This often forms a network of devices that are connected together through Wi-Fi or Bluetooth. They can communicate together and form a mesh network, or communicate independently to a central collection.

Data Processing: This is where the data collected by the devices is processed and analysed. This can be done on the devices themselves, in the cloud, or on a combination of both. This may also be linked to a machine learning algorithm which can act on the data immediately.

User Interface: A device can be connected directly through cables or Bluetooth. This makes the device very versatile, and the Arduino Nano 33 BLE is brilliant for this.

Processing: This is where Arduino Nano 33 BLE boards start to earn their keep. They have enough processing power which allows you to run, albeit relatively small, neural networks on them. Although they are nothing compared to LLM or big data models, they can still have a place. They have an extremely low energy requirement.

The ESP32-S3 boards do have considerably more memory than the Arduino boards, but for this exercise, I have found the Arduino Nano boards to be a bit more robust in using the Arduino Code Editor (IDE).

More than a humanoid robot

Definition: If you think of the word robotics, what does it make you think of? A killer humanoid machine, a robotic arm in a factory, even a robotic dog called Spot? Invariably, it is something that moves, completes tasks, and may or may not be humanoid, although a number can interact with you as if they were human.

More than a robot: The term robotics is more than just another word for robots; it encompasses not just motors but also sensors and displays. Another term that we could use is Physical Computing. Everything that is more complicated than a torch (flashlight) has a microchip in it and has therefore some code embedded in it.

The heart of a robot: At the heart of robotics is the microcontroller. It is the hardware that the code runs on. It is usually what looks like a very small black tile on a printed circuit board. There are, as you might expect, many different types. Each having varying functionality but in essence, do almost the same thing. They control very small electrical impulses through tiny switches or gates.

Purpose: We won't get too deep here because we are more focused on how to use these chips. For that, we have a wide variety of microcontrollers to choose from. We can wire up our motors, sensors, and displays to make them serve a particular purpose; your car, microwave, washing machine, TV all use them. They monitor what they are doing, feeding back information, allowing you some control to programme them, choose a set programme, or switch channels, for example.

Hard code: The machines (TV, cooker, etc.) need human input through pressing buttons, turning dials, or even speaking to them. These simple components are connected to the microcontroller, and that is where the coding comes in and responds to the press of a button and the turning of a dial.

Code that learns: What if instead of being a simple machine, we give it a brain and train it to perform tasks that might be more challenging to hard code? Machines that have learned or can learn, develop, and improve. That is what we are doing with intelligent machines.

Which micro-controller?

Too much choice: I alluded to the fact that there is a bewildering variety of microcontrollers. They all have their merits; some cost a few pounds/dollars/euros, but others seem to be considerably more expensive, although to be honest, nothing is ridiculously expensive.

TARDIS-like: The problem is not the choice but what we are going to do with the microcontroller; we are going to put a neural network inside the chip. Although it is going to be a very small neural network model, it will still demand a lot of memory. Most microcontrollers, by their very nature, have very small memory chips.

Introducing the Nano family: The microcontroller is usually the size of your index finger (I know people's index fingers vary in size, but you get the general idea), so you can't expect to run Windows on it. The good news is that there are some microcontrollers that do have enough memory, and one I particularly recommend is the Arduino Nano 33 BLE.

Other boards: If you demand much more memory, then I suggest an ESP32-S3 (Arduino do a Nano version of the ESP32-S3), but also XIAO do a tiny board the size of your thumbnail. Another board you could use is the Arduino Nano 33 BLE Sense v2, which has a large array of sensors but is twice the price. Something like an Arduino Uno just doesn't have the memory space.

What’s on the board?

What does it have?: The Arduino Nano 33 BLE has a trio of sensors: an accelerometer, a gyroscope, and a magnetometer. It also has a built-in LED and RGB LED, which we can use. It also has a lot of pins to which we can connect different components, such as other sensors, displays, and motors.

Bluetooth: It has Bluetooth (but no Wi-Fi), which we can make use of by connecting several of these devices together in a mesh network.

So what?: Wearable tech and smart appliances make use of the latest developments in microchip technology. Some appliances have elements of AI in them, for instance, wake words for smart speakers.

Small is beautiful: Although AI (or machine learning) is predominantly the domain of larger machines that can handle a large model (trained Deep Neural Network), it is possible to install a small truncated Deep Neural Network on a small microcontroller if it has enough memory space.

Elegant solutions: This gives the opportunity to develop a range of applications and gadgets. But first, we need to have some understanding of how they work, and the Arduino Nano 33 BLE is a good board to start with. Not only does it have enough capacity, it also has lots of pins we can connect components to.

It is a starting point: In the end, you are unlikely to use it for anything commercial. There are other chips and form factors you would likely use being manufactured on a circuit board, but that is for another day.

Bigger is not necessarily better: Most talk about AI focusses on Large Language Models, which are costly in terms of GPUs (data centres), power (cooling and running). We are going in the opposite direction; these use very little power and have to be efficient. They will lack the breadth of an LLM, but that doesn’t mean they are redundant. I predict we will see more development in this area, especially if the interest in LLMs wanes or the AI bubble bursts.

`You can still be a player`: Although only big tech companies can handle LLMs, they cost billions just to keep running. Embedded AI is accessible to anyone as long as you don't want to take on ChatGPT. Saying that these microcontrollers can be linked to an LLM or SLM (Small Language Model) and harness their power if you so wish, we may come onto that later.

How to get started

Ecosystem: The first thing is to understand the ecosystem. We are using an off-the-shelf board. We are going to use the Arduino IDE to programme the microcontroller. I will assume for this that you have completed the section on p5.js/ml5.js of this AI tutorial. We will be using that knowledge later on.

First things first: First, we need to connect our Arduino and learn how to programme it, looking at all its features. If you are familiar with programming a microcontroller, such as the Arduino Uno, then this is familiar territory. You could skip some of module A, but there may be things that are new to you, and in any case, it won't hurt to practice. Although the next couple of units will go into more detail about the hardware and software you will be using, I will give you a brief summary here.

Hardware: Mostly, you will need an Arduino Nano 33 BLE with soldered pins, which is important, a half-size breadboard, and a micro USB cable. As you get near to the AI bit, you will be introduced to buttons and how you can connect them to the Nano.

Software: To programme the board, we need to use an IDE of some sort. It is what we use to upload the code. We do have a choice. We could use the cloud version of the IDE, which is good for a number of reasons. However, I won't be using it, simply because the free version only allows you 25 uploads per day. Instead, we will use the downloadable version of the IDE (version 2.x) and use that. You have unlimited uploads! The downside is that you have to manually update boards and libraries as well as the IDE version (if there are any updates). You are informed when there are updates.

Computers: You can use any Mac, Windows PC, laptop, or Raspberry Pi or Linux operating system.

Chromebook woes: If you are using a Chromebook, you could use the Arduino Cloud, but you will have a problem (at the time of writing). The Arduino Nano 33 BLE is not compatible with the Arduino Cloud on the Chromebook. Hopefully, it will be in the future.

Hardware you will need

Which board: The main bit of kit you will need is a board. There are quite a few to choose from that are either made by Arduino or other companies. Using Arduino Nano 33 BLE boards is a very popular option as they are cheap (≈£25/€27) and easily available.

Tight budget: If you are on a very tight budget, you could also go for the XIAO ESP32-S3, which is the size of your thumbnail.

Minimal: I have endeavoured to keep the amount of electronics to a minimum in this tutorial and focus more on the code and the machine learning in particular. However, the whole point of microelectronics is that you use components and create some wonderful piece of technology or gadget.

Components: To start to develop a good understanding of how you can use the Arduino Nano 33 BLE board, I will introduce you to a small number of components. This will give you a sense of how to integrate them into a project.

Hardware bits and pieces: So here is a list of hardware (components) you will need. They are easily sourced on eBay, Amazon, or specialist stockists like PiHut or Pimoroni.

- Arduino Nano 33 BLE (with headers)
- Micro USB cable (to connect it to your computer)
- Breadboard (need a half-size)
- Buttons (push, momentary, tactile)
- Jumper leads (male-to-male sort to connect the components on the breadboard)

Arduino Nano 33 BLE

Soldered pins: Important, get the Arduino Nano 33 BLE with pins already soldered on unless you are a dab hand at soldering. Unlike the Uno, which operates on 5 volts, the Arduino Nano boards work on 3.3 volts. Some components are 5 volts only, some are dual-use, and some are designed for 3.3 volts only. Always check the specification.

Automation: With the built-in Bluetooth module, the Arduino Nano 33 BLE can communicate with other devices and sensors. This makes it suitable for a wide range of applications, such as home automation, remote monitoring, data logging, and more.

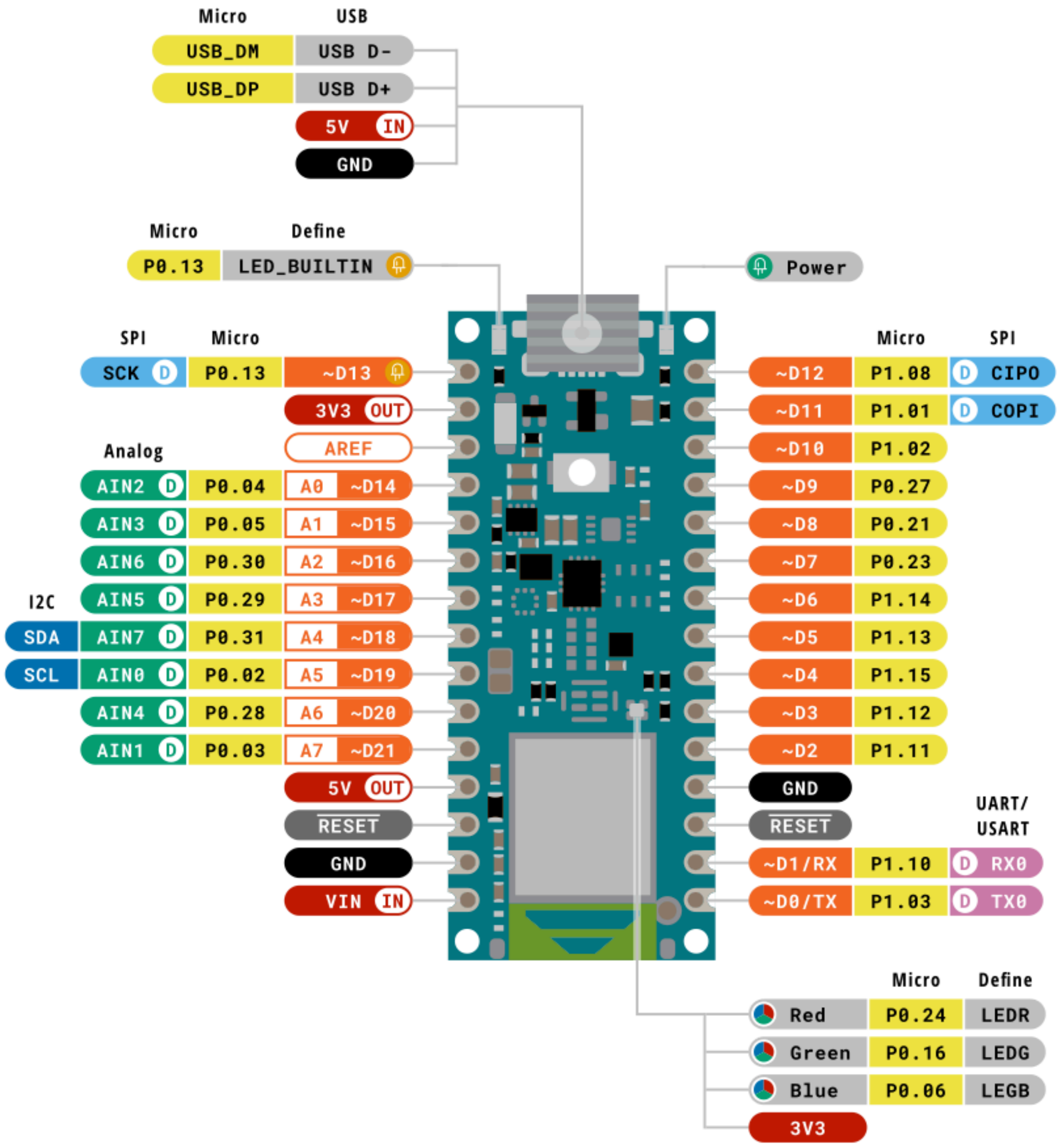
The IDE: The board's compatibility with the Arduino Integrated Development Environment (IDE) also makes it easy for beginners and hobbyists to get started with programming and electronics. The IDE provides a simple and intuitive interface for writing and uploading code to the board, as well as a large community of users and resources for learning and troubleshooting.

Digital and Analog: The Arduino Nano 33 BLE has a series of pins running along each edge of the board. Roughly one side is for analog components, and the other side is for digital components. Where the prefix A is for analog and the prefix D is for digital, followed by a pin number. There are other pins which have various functions; the ones we will be making use of will be the GND or ground pins.

IMU: We have access to the accelerometer, gyroscope, and magnetometer. This is something we will make use of later. If you have a board (such as the ESP32), then you will have to buy a separate module. This is not a great problem but is something you will have to navigate. Hence why I have gone for the Nano 33 BLE.

Pin diagram: To work out which pin is what, you will need to use the pin out diagram. The pins are numbered on the front (very tiny). The analogue pins can also be used as digital pins if needed, see fig.1.

Figure 1: pin out



Get to know your board

Here are some images of the board.

Figure 2: oblique view

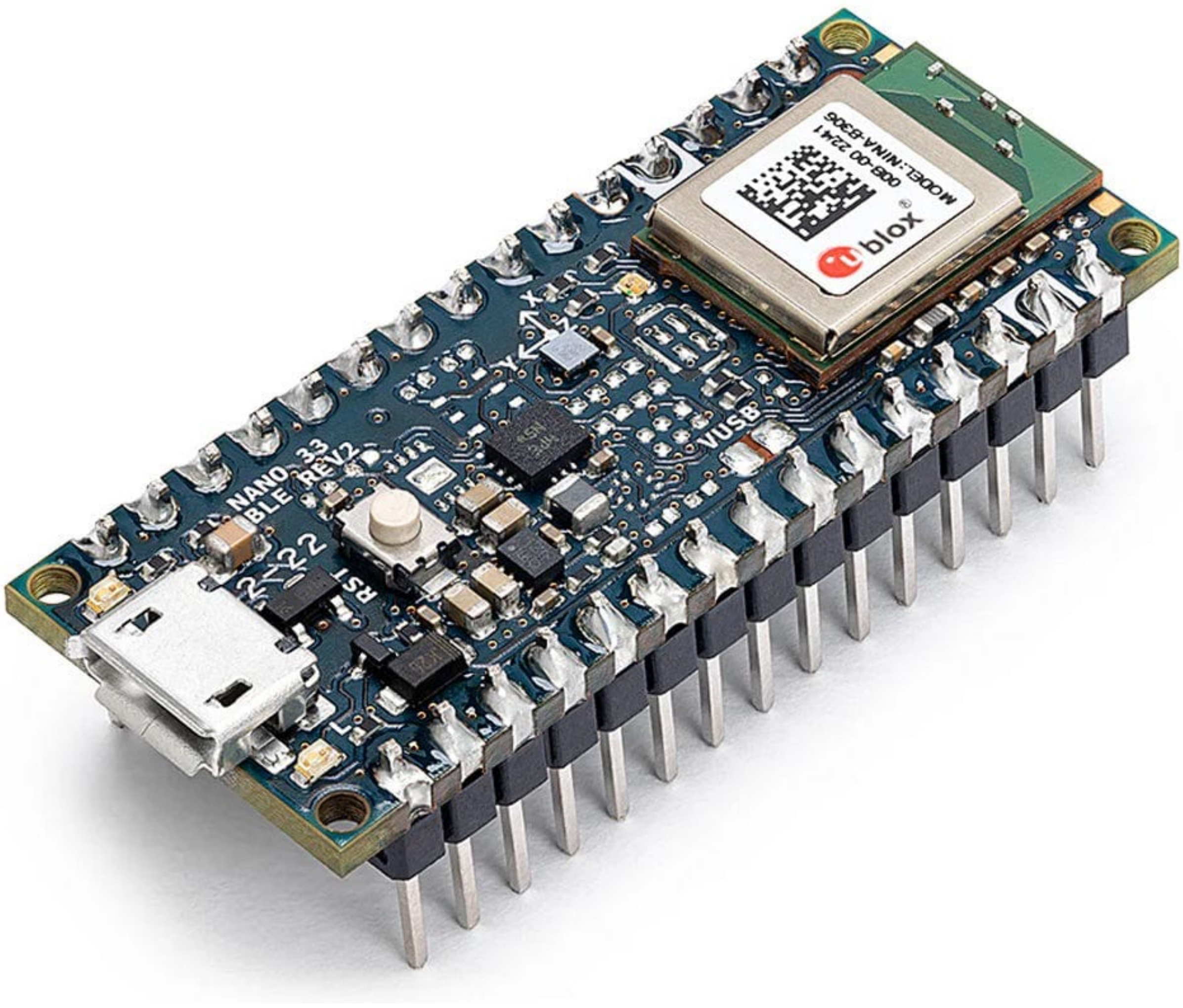


Figure 3: top view

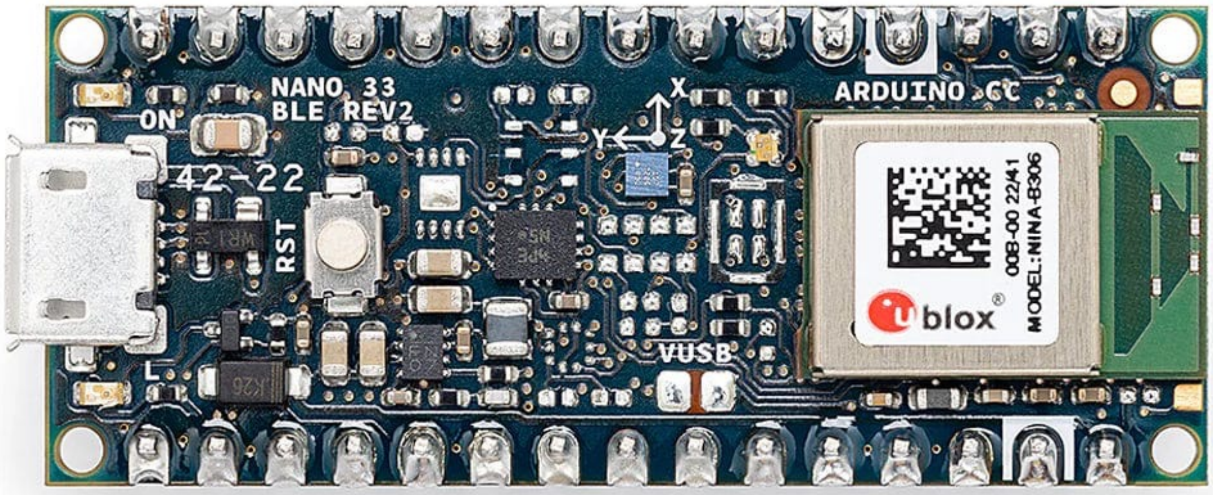
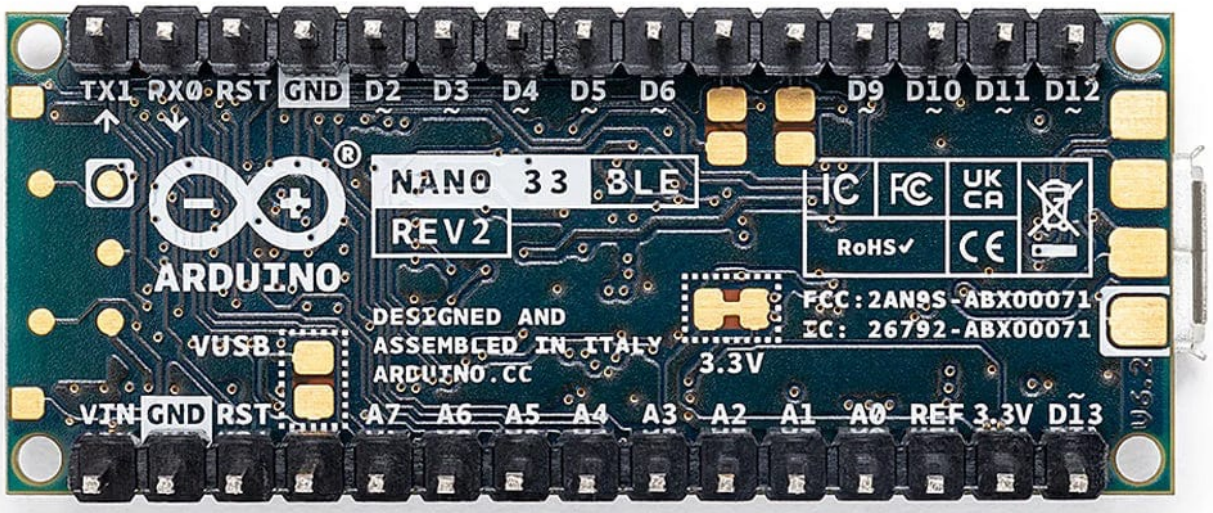
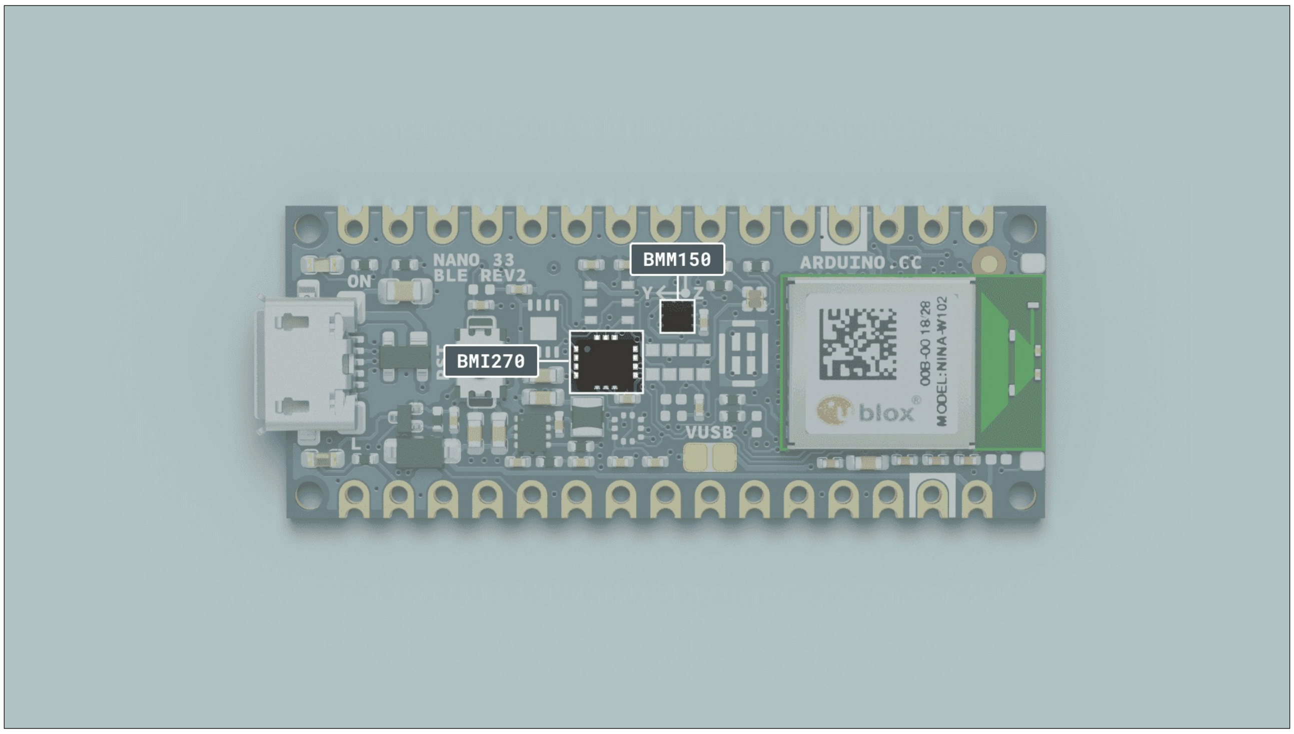


Figure 4: underneath



Below is an illustration of the IMU on the board. There are two sensors: the BMI270, which is the accelerometer and gyroscope. The other sensor, BMM150, which is a magnetometer that measures the magnetic field. More on those later.

Figure 5a

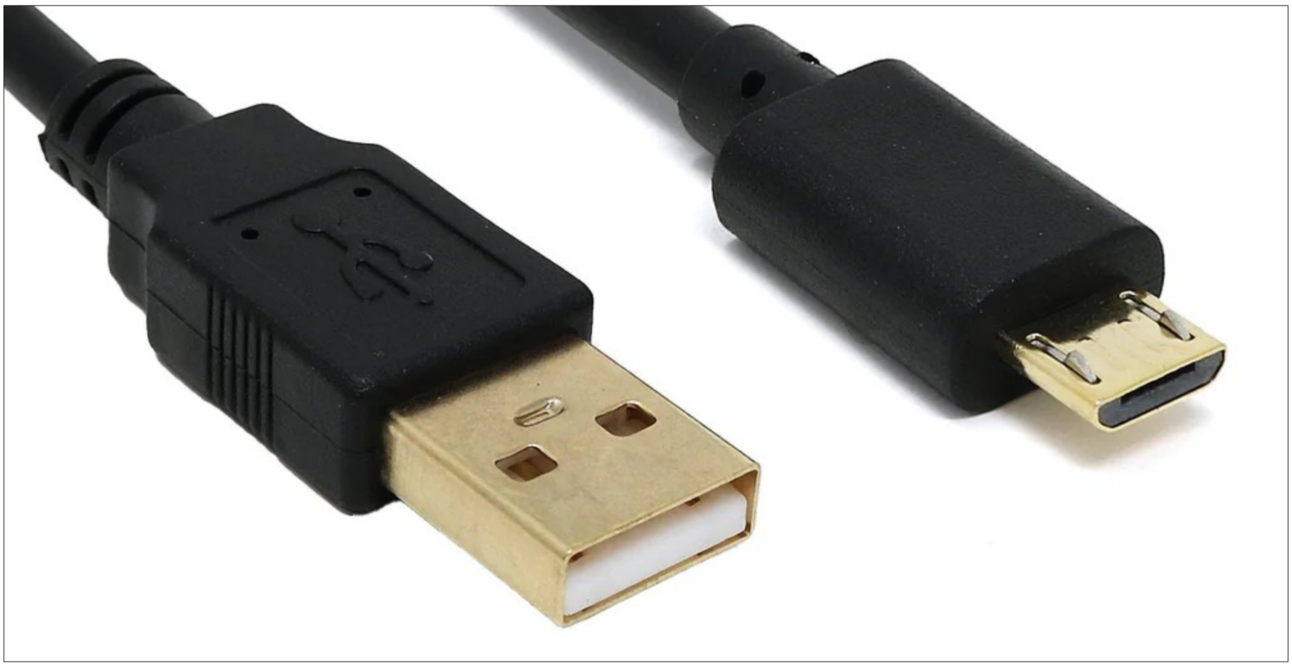


USB cable

To connect your Arduino Nano 33 BLE to your computer and to power it, you will need a micro USB cable (shown below). Make sure you get one that can do both of the following:

1. To power the Arduino Nano 33 BLE, from a 5-volt source such as a normal phone charger or battery pack.
2. It sends data to and from the computer and the Arduino Nano 33 BLE, such as uploading your code.

Figure 6: micro USB



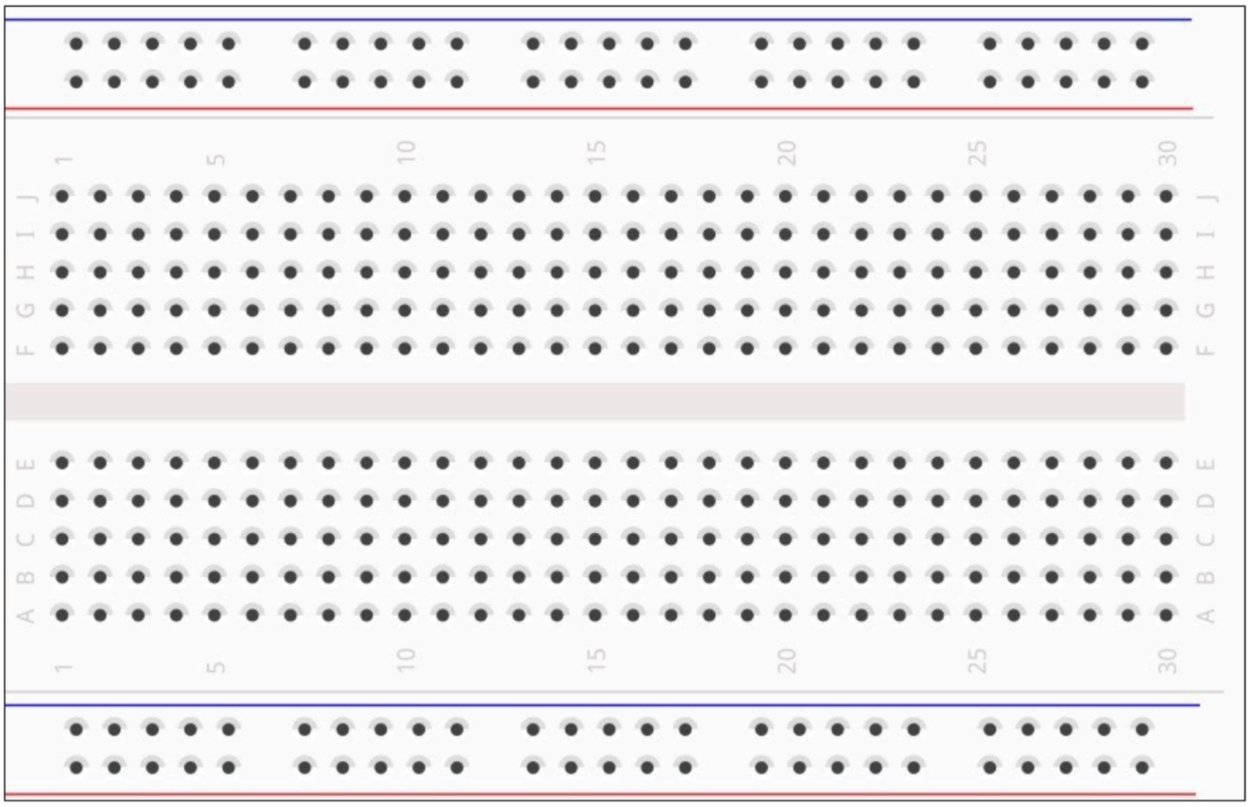
Breadboard

The breadboard is simply a great way to connect components without having to solder anything. There are strips of metal underneath that connect the holes together.

There are many sizes and varieties of breadboards. This one is a half-size breadboard; it has 400 pin points. They can be larger (full size) and smaller (mini). The half-size is perfect for our purposes and most projects. Whichever one you get, they all work on the same principle.

I place mine on a coaster, not essential but useful if you were to add components that don't fit onto a breadboard.

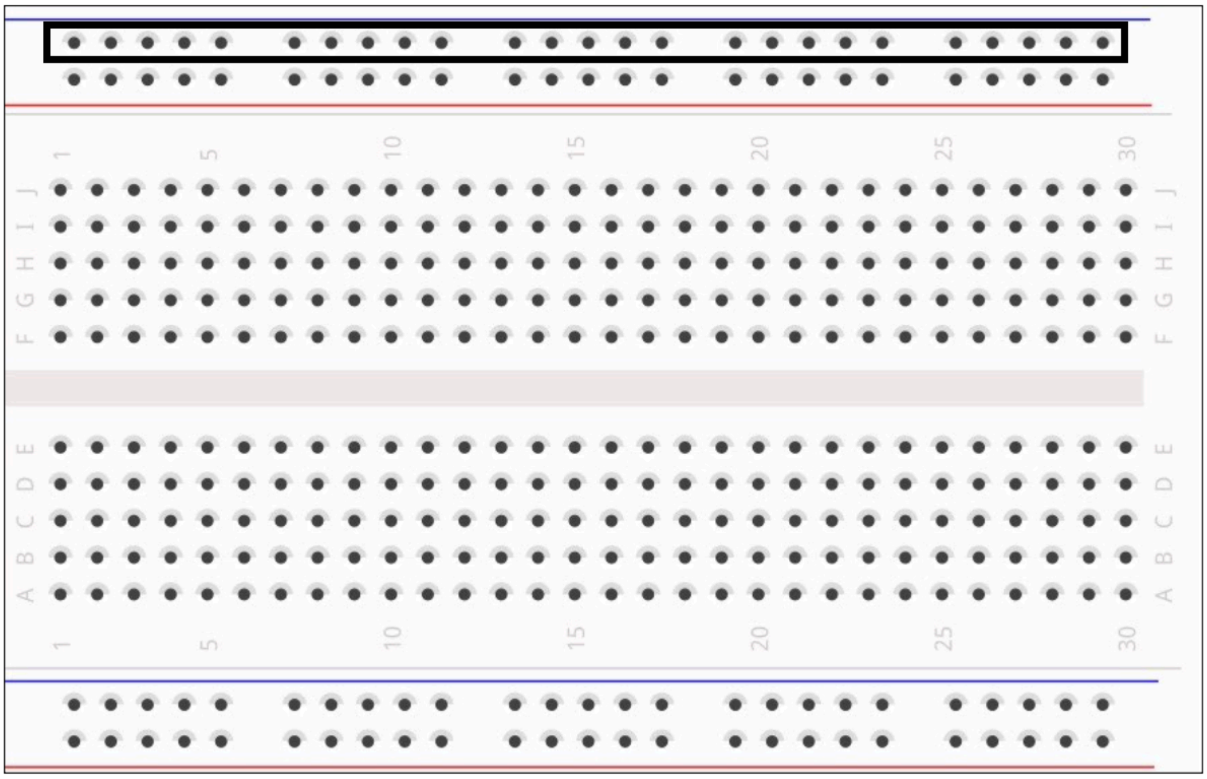
Figure 7: Half-Sized Breadboard



Using the Breadboard

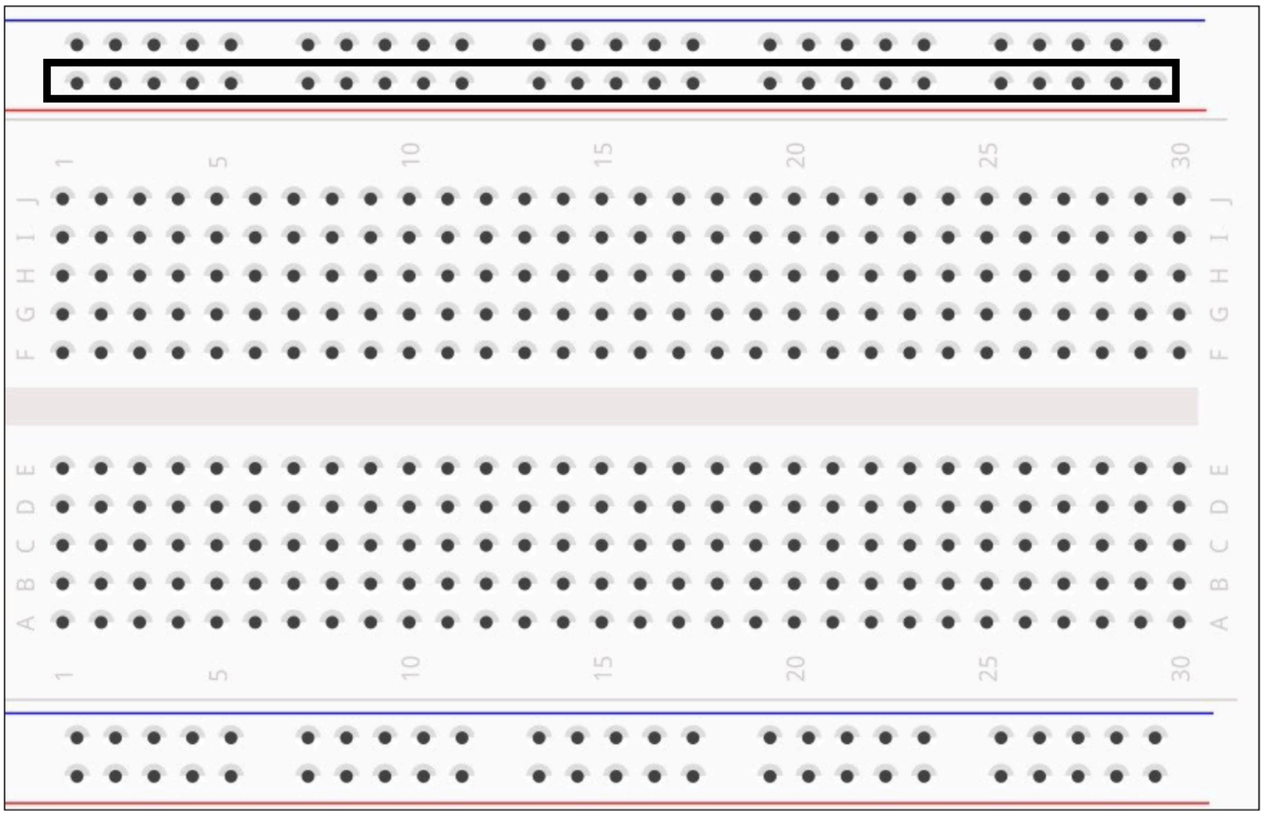
The ground (GND) rail is connected along the whole length of the breadboard. This means that if you have a number of components all wanting to connect to the ground (GND), you can plug them all into the same one. This is the one near the blue line.

Figure 8: connecting along the length of the negative/ground rail



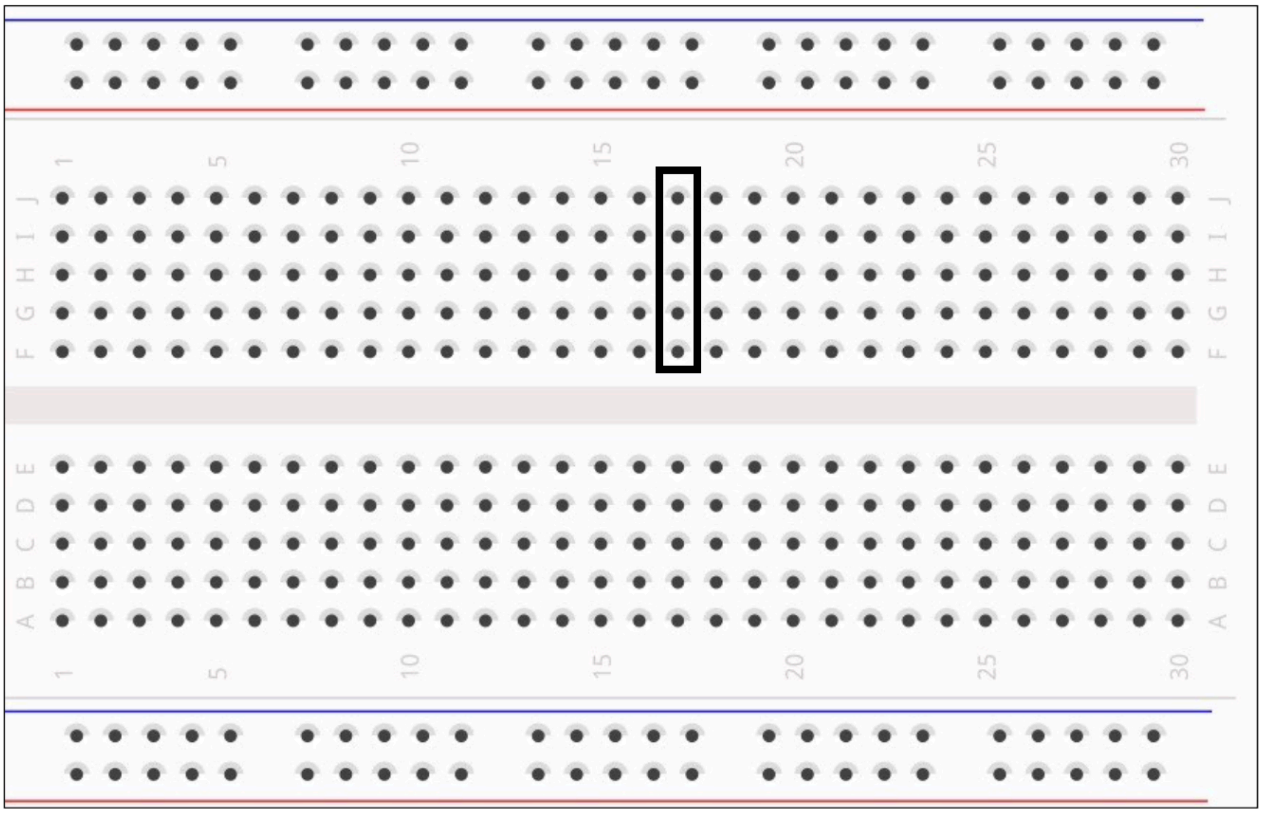
The positive (VCC) rail is connected across the entire length of the breadboard, as is the ground (GND). If you have lots of components that need a 5-volt supply (VCC), such as a servo, then you can use this to supply all of them. Convention is to follow the red line.

Figure 9: connecting along the length of the positive rail



The holes are connected across in rows of five, so that anything in that row is connected electrically. Each row of five holes is separate from all the other sets of five holes.

Figure 10: five connected pin holes



Adding the Arduino to the Breadboard

One of the reasons for getting the Arduino Nano 33 BLE with headers (pins) already soldered on is so that we can add it to the breadboard. Take care when doing this so that you do not bend the pins. Notice that it does not sit centrally, so it has to be offset to the left or right; it doesn't really matter, but I suggest doing what I have done because we use the digital pins side more than the analogue side.

Also notice that when it is connected (and powered), a little greenish (power) LED tells you that it is on.

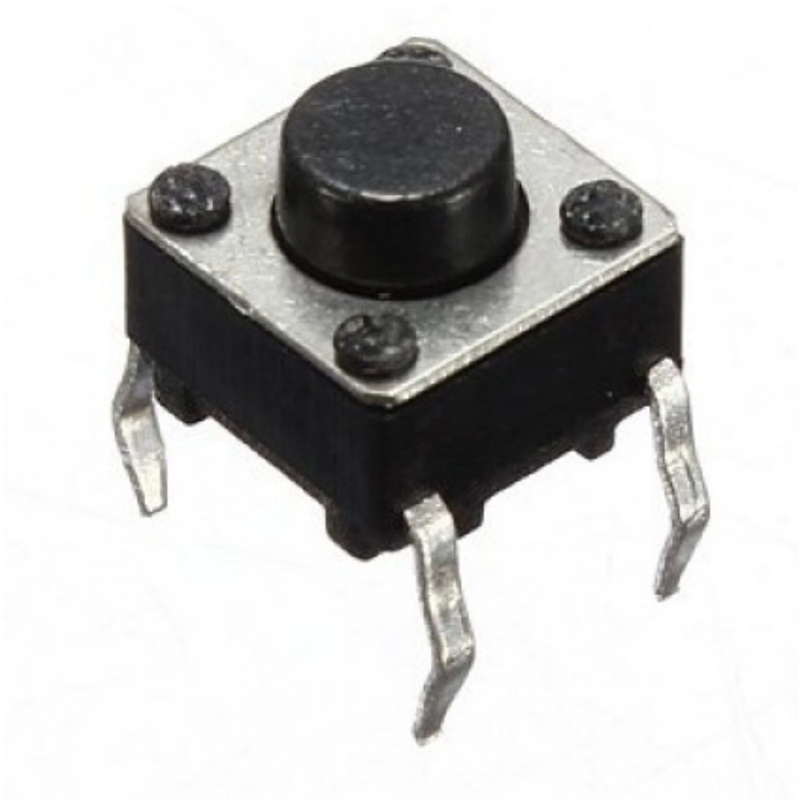
Button

The button is a tactile, momentary type button, so it only completes the circuit while it is pressed. They come in a variety of sizes (and colours), and mainly they fit on the breadboard, although button modules don't.

One of the issues is that they will need a resistor in series so that the current isn't too high for the pin and would damage the board (this is also the case with LEDs). You can simply place a resistor in series or use a button module that has a resistor already built in. A third option is to use an internal resistor already within the board itself called a pull-up resistor.

For simplicity, we will use a version of `fig.11` and use the pull-up feature of the Arduino Nano 33 BLE.

Figure 11: button



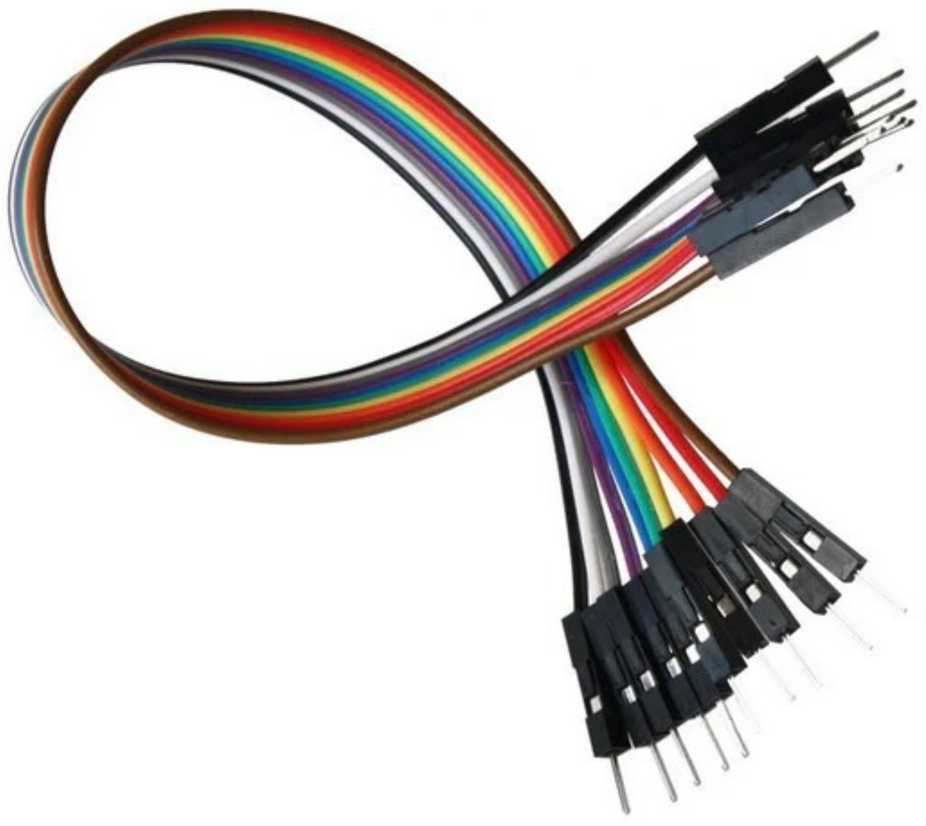
Jumper wires

These are wires widely used in electronics to connect components on a breadboard. They have pins sticking out or holes at the ends of the wires. Get ones that are as long as possible, or better still, get a mixture of lengths. There are three main types:

- Male-to-Male connectors (see Fig.12)
- Male-to-Female connectors
- Female-to-Female connectors

What you will need are Male-to-Male, and they usually come in packs of ten, which is plenty. You might also want to get Female-to-Male and Female-to-Female while you are at it.

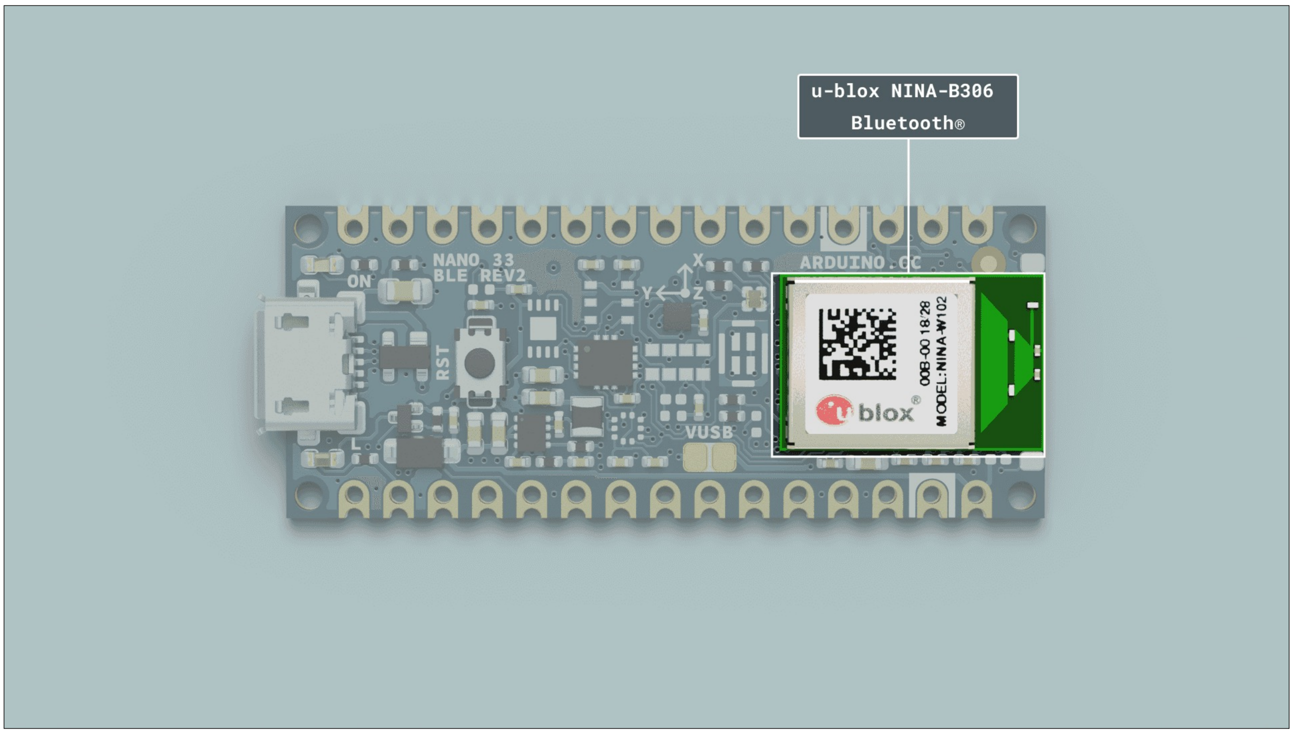
Figure 12: male-to-male jumper leads



Bluetooth

The Arduino Nano 33 BLE also has Bluetooth, which means it can communicate with other Bluetooth devices, sending data to each other. There are two types of Bluetooth:

Figure 13: bluetooth module



Bluetooth Classic: Which can send lots of data but is energy-hungry, so only useful if powered directly.

Bluetooth Low Energy: Which is, as the name suggests, low energy because it is reserved for small amounts of data and therefore is more suitable for battery-powered devices.

ESP-NOW (*ESP boards only*): This is another protocol that allows you to connect several ESP devices together. It has very low latency.

More than one: The board in Bluetooth mode is either a peripheral device or a central device, depending on whether it is sending data or receiving it. For obvious reasons, we will need at least two devices for that demonstration. They don't have to be exactly the same device.

Connecting to your computer

Your first step is to put the Arduino Nano 33 BLE on the breadboard as shown below. Use the USB cable to connect the microcontroller to your computer. The power LED should light up to tell you there is power going to the microcontroller and that you are now ready to upload your sketch. Before you can do that, we need software, and that is where the next unit explains all.

Figure 14: connecting your Arduino Nano 33 BLE