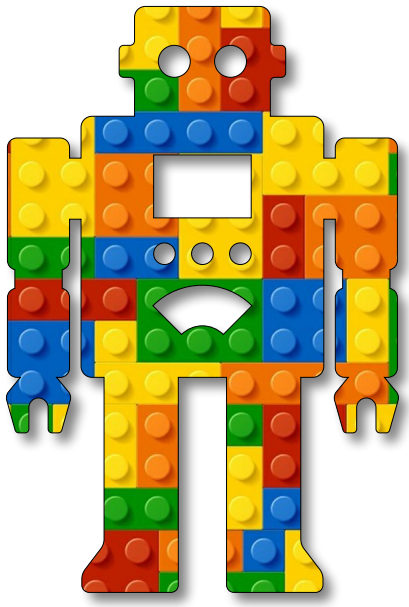
Intelligent Machines Module E Unit #1 servo



Content

Module E Unit #1 servo

Introduction to the servo

The servo is a particular type of motor. It usually moves through a limited number of degrees. In this case, it is 180°. You can get similar ones that are continuous.

What you will need

In addition to the Arduino Nano 33 BLE and breadboard, you will need the following:

1 x breadboard power supply
1 x 9-volt battery
1 x 9V battery clip with 5.5mm/2.1mm plug
1 x TowerPro Servo Motor - SG92R Micro
4 x male-to-male jumper leads

The breadboard power supply

The power supply for the breadboard provides 5V or 3.3V along the side rails. There are a number of different types and methods of supplying power. We need to supply separate power to the servo because the Arduino Nano 33 BLE won't be able to handle the extra power needed if the servo is loaded (meets resistance). It could damage the Arduino.

The power supply module shown in the image below is a popular and very cheap one. When you place it on the breadboard, it is important that you share the ground with the board's ground.

You select whether you want 5V or 3.3V (or neither) by lifting the plastic slider and placing it over the relevant pins. The slider is shown in yellow on both sides of the power module. The 9V battery is connected by inserting the jack plug (black). The USB can be connected to the Arduino to power it (5V).

Figure 1: breadboard power supply

The 9v battery and connector

The power supply to the above module is a 9v battery with the appropriate cable. See fig.2 and fig.3.

Figure 2: 9v battery cable

Figure 3: 9v battery

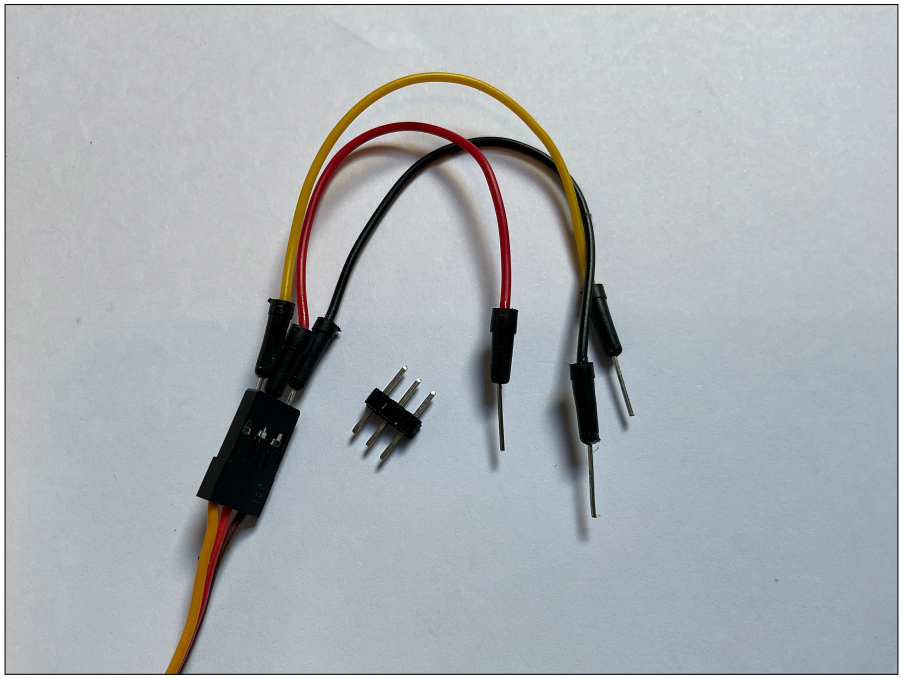
The servo motor that is cheap and plentiful is the TowerPro Servo Motor - SG92R Micro or the TowerPro Servo Motor - SG90 Digital. They are both quite similar. The SG92R is a slight upgrade.

Both take a ≈5v supply from the rails on the breadboard. They have three leads. A red lead which is the 5v power supply, a brown lead which is the ground (GND), and an orange lead which will be attached to digital pin 2 in our examples.

Figure 4: servo motor

To make life a little simpler, you might want to get hold of a strip of pins that you can break off any number to use. I use them in this case on the breadboard to make connections between the servo and the components/breadboard. Either method is fine.

Figure 5: two ways to connect servo to Arduino



The circuit diagram

The circuit diagram is shown below. Notice the common ground.

Servo wires	Arduino Pins
Red	+ve (5V) red rail on breadboard
Brown	-ve (GND) blue rail **and** GND Arduino
Yellow	2 (digital pin 2)

Figure 6: circuit diagram

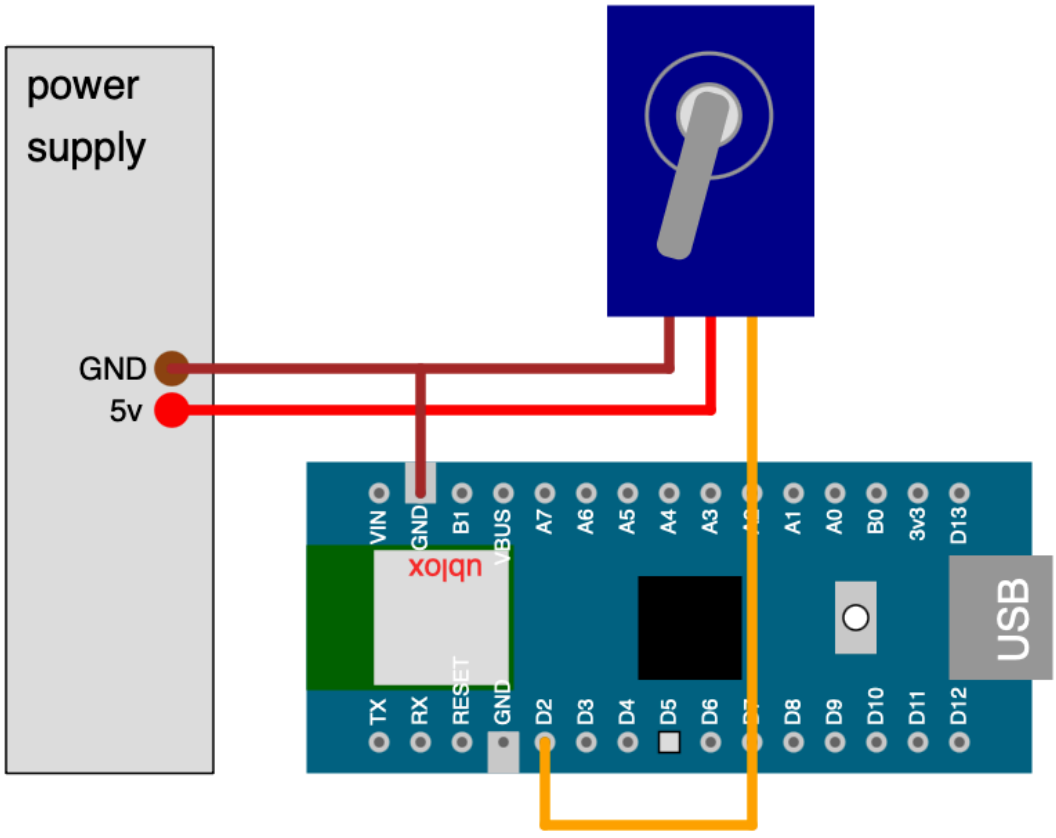
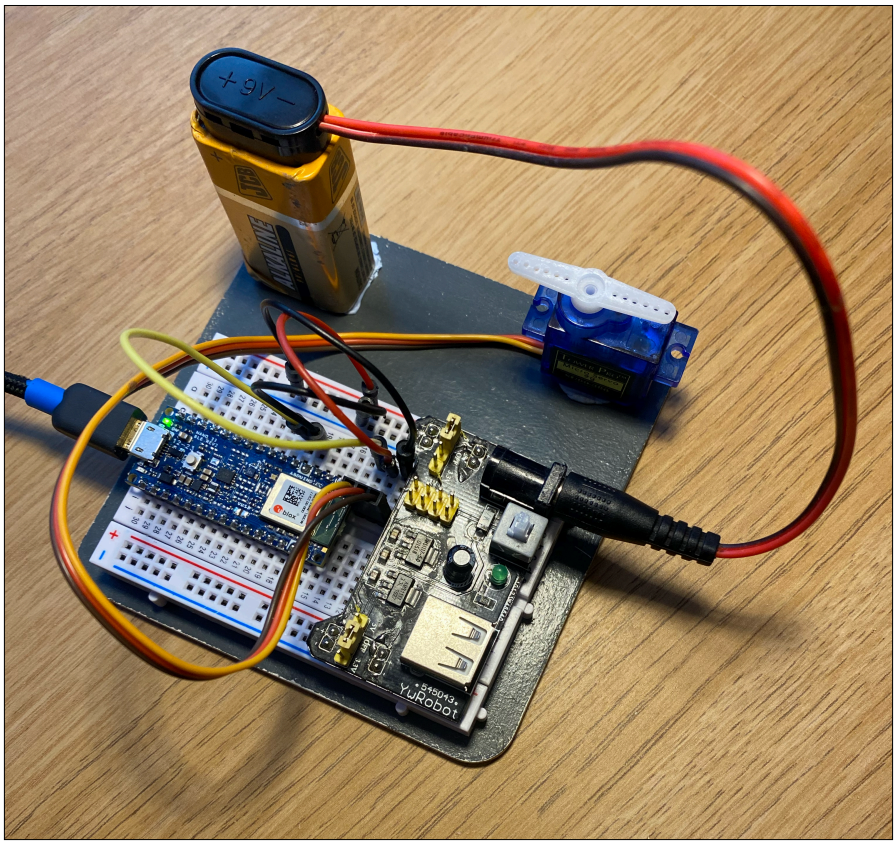


Figure 7: fully connected



Installing libraries

This is a critical part of learning to code with the Arduino and with using other coding languages like Python. `Libraries` are like little modules that contain a sketch or programme that does a certain function to control a complex component. It saves you having to write a lot of complicated code.

Some libraries are pre-installed, for instance, the `servo library`. Others you will need to install manually. This is very simple and quick to do. If you don't, you will soon get an error message, and no damage is done.

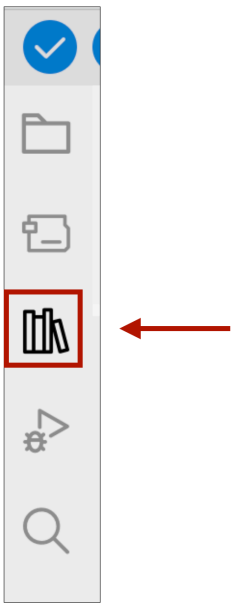
In the code, you will have a line like so...

```
#include <Servo.h>
```

This means that you are including a library called `Servo.h`. Sometimes you will only have to install one library, and at other times, you will install many. You can even write your own library, but that is beyond the scope of this book.

1 Click on the symbol that looks like a stack of books (this is where the libraries are, hence like a library shelf).

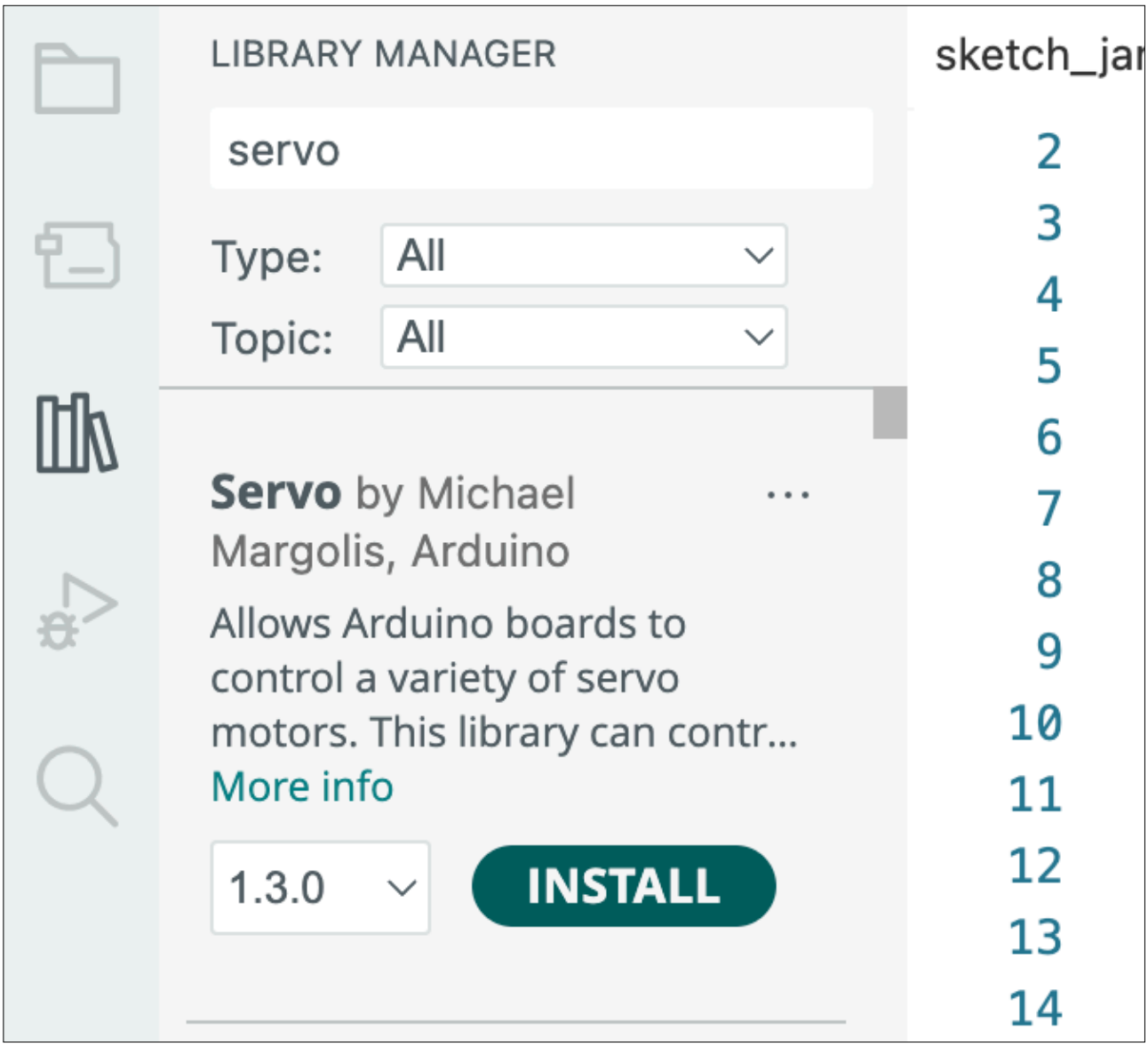
Figure 8: library



2️⃣ Type in the word `servo` in the box at the top and press Return. You should get a long list of possible libraries to install. The one that I recommend is highlighted here.

It is the one by Michael Margolis, which seems to be Arduino's own. Click INSTALL (latest version).

Figure 9: servo library



Sketch E1.1 position zero

Our first servo sketch. This positions the servo at an angle of 0°.

Arduino sketch

```
#include <Servo.h>
Servo myServo;

void setup()
{
  myServo.attach(2);
}

void loop()
{
  myServo.write(0);
  delay(15);
}
```

🗒️ Notes

❗ You may find that the servo doesn't behave well. Mine didn't. So you could set it to 10° instead if it starts to rotate of its own accord.

🌻 Challenge

Try other angles.

🛠️ Code Explanation

`#include <Servo.h>`	The servo library
`Servo myServo;`	We give the servo a name myServo, it can be any name and will need to be if you are operating more than one
`myServo.attach(2);`	The servo is attached to digital pin 2
`myServo.write(0);`	The angle is written to the servo in degrees
`delay(15);`	Gives the servo time to get to new position

Sketch E1.2 now 90 degrees

Next, we change the angle to 90°.

Arduino sketch

```
#include <Servo.h>
Servo myServo;

void setup()
{
  myServo.attach(2);
}

void loop()
{
  myServo.write(90);
  delay(15);
}
```

Notes

Fairly obvious, just to make the point. It should move round to a new position after you have uploaded.

Sketch E1.3 finally 180 degrees

Just to top it off.

Arduino sketch

```
#include <Servo.h>
Servo myServo;

void setup()
{
  myServo.attach(2);
}

void loop()
{
  myServo.write(180);
  delay(15);
}
```

🗒️ Notes

And it should move to its final resting place.

🌻 Challenge

Have a think about how you would rotate it continuously from 0° to 180° and back again. The next sketch will show you.

Sketch E1.4 repeated rotation

We create two `for()` loops. The first one checks for the position of the servo arm. If it is zero, it increments `1°` at a time until it reaches `180°`. The second `for()` loop decreases the angle by `-1°` from `180°` back to `0°`. This is repeated.

Arduino sketch

```
#include <Servo.h>
Servo myServo;
int pos = 0;

void setup()
{
  myServo.attach(2);
}

void loop()
{
  for (pos = 0; pos < 180; pos += 1)
  {
    myServo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1)
  {
    myServo.write(pos);
    delay(15);
  }
}
```

🗒️ Notes

I have not broken it down into separate sketches because by now you should be able to understand how the `for()` loops work, the conditional statements, and the incremental steps.

🌻 Challenge

Try rotating it from and to different angles.

🛠️ Code Explanation

`for (pos = 0; pos < 180; pos += 1)`	We cycle through from 0 to 180 in steps of 1° each iteration

Sketch E1.5 variation on a theme

Just to complete the picture, we could also do the following and watch it on the serial plotter.

Arduino sketch

```
#include <Servo.h>
Servo myServo;
int pos = 0;

void setup()
{
  myServo.attach(2);
  Serial.begin(9600);
}

void loop()
{
  for (pos = 0; pos < 180; pos++)
  {
    myServo.write(pos);
    delay(15);
    Serial.print(0);
    Serial.print(" ");
    Serial.print(180);
    Serial.print(" ");
    Serial.println(pos);
  }
  for (pos = 180; pos > 0; pos--)
  {
    myServo.write(pos);
    delay(15);
    Serial.print(0);
    Serial.print(" ");
    Serial.print(180);
    Serial.print(" ");
    Serial.println(pos);
  }
```

```
}
```

🗒️ Notes

We have an extra two reference points to keep the plotter from going a bit berserk.

🌻 Challenge

Could you make the servo rotate according to a sine wave?

Figure E1.5