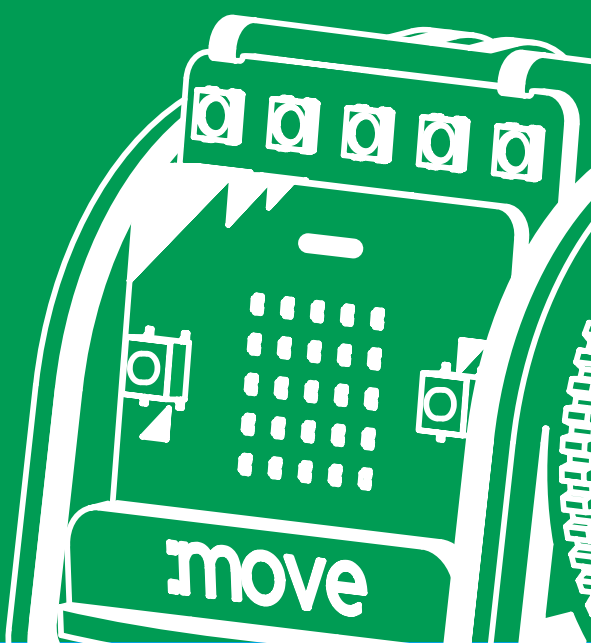
Kitronik

FOR BBC micro:bit

:MOVE mini



V1.1

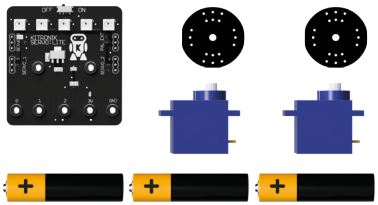
The **Kitronik :MOVE mini for the BBC micro:bit** provides an introduction to robotics. The :MOVE mini is a 2 wheeled robot, suitable for both remote control and autonomous operation. A range of add-on boards can expand the capabilities to include more advanced functionality. The included :MOVE Servo:Lite board can also be used in conjunction with a BBC micro:bit to build other movement based projects.

Kitronik have created custom blocks for the MakeCode coding environment, these make it ultra simple to code your :MOVE mini. Give it a try by adding the 'Servo:Lite' blocks from the Extensions tab in MakeCode!

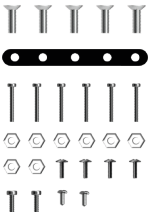
CONTENTS

SUPPLIED WITH THIS KIT

ELECTRONICS



FIXINGS



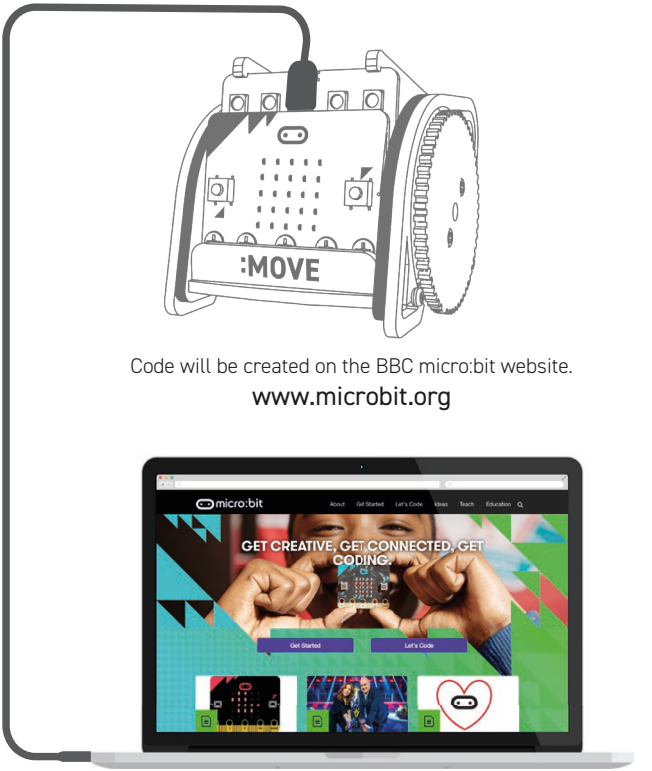
CHASSIS PANELS

2x

1x

GETTING CONNECTED AND FINDING THE PROGRAMMING ENVIRONMENT

Using a USB to micro-USB type B cable, connect the BBC micro:bit to a computer.



Code will be created on the BBC micro:bit website.

www.microbit.org

THE BBC micro:bit SOFTWARE

The BBC micro:bit is programmed using a web based (internet access required) programming environment which is found at **www.microbit.org.**

To save programs to access at a later date you will need to save the .hex files you create (these files will be explained later in this booklet). To reload a program the relevant file needs to be dragged on to the editor screen. If this is the first time you have used your BBC micro:bit then please refer to our getting started guide at **www.kitronik.co.uk/microbit.**

All of the experiments in this guide are based around the **Microsoft MakeCode** and **Microsoft MakeCode JavaScript** editors. The Microsoft MakeCode Block Editor is a very easy to use graphical editor. The Microsoft MakeCode JavaScript editor is a text based programming language which is ideal for slightly more complex programs. It is possible to convert a Block program into JavaScript. This offers an easy way of progressing from Block programming to JavaScript. Other editor options include the Python Editor. Refer to **www.kitronik.co.uk/microbit** for tutorials based on this.

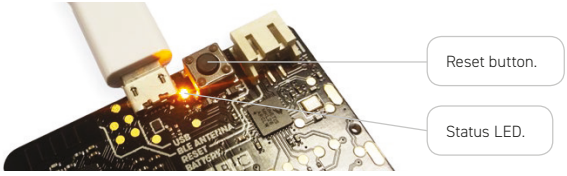
GETTING A PROGRAM ON TO THE BBC MICRO:BIT

It is very easy to transfer a finished program to the BBC micro:bit. First of all select 'download' . This is where the program is converted into a program the micro:bit can understand. This is known as a '.hex' file. If it has compiled successfully, it will return the message 'Download completed... Move the .hex file to the MICROBIT drive to transfer the code to your micro:bit.' The message 'Do you want to open or save microbit-script.hex from microbit.org' will appear. Select 'Save As' from the 'Save' drop down menu and save the hex file to a folder for BBC micro:bit .hex files.

Next plug a BBC micro:bit into the computer via USB. The BBC micro:bit will appear as a removable drive on the computer called 'MICROBIT'.

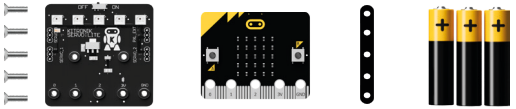
To download the .hex file to the BBC micro:bit 'Drag' the .hex file from the folder where it was saved and 'Drop' it onto the MICROBIT removable drive. A message will appear saying ' Copying 1 item..... to MICROBIT'. At the same time, the yellow LED on the back of the BBC micro:bit will flash.

After a few seconds, the download will complete and the BBC micro:bit should now be running the program.

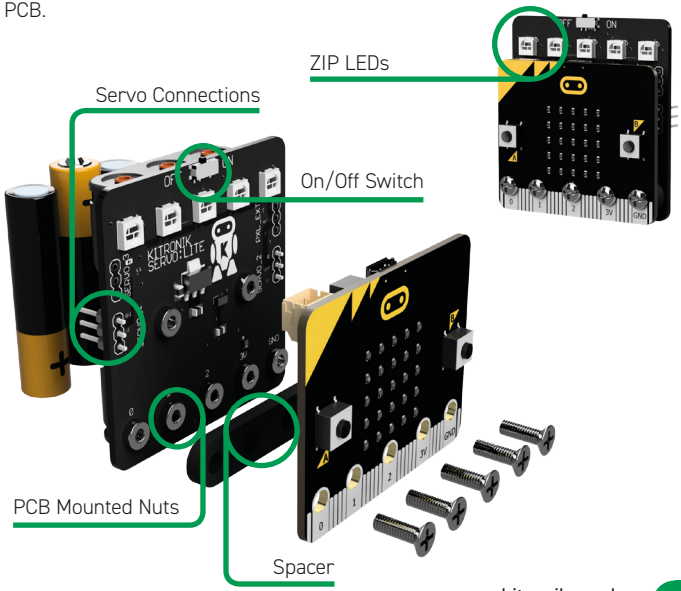


If it doesn't you may need to press the reset button, which is next to the status LED.

1 ASSEMBLING THE SERVO:LITE PCB



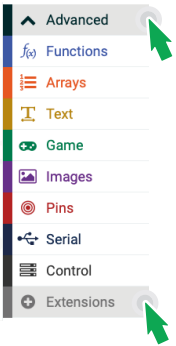
STEP 1: Use a small Phillips screwdriver to screw the five M3 machine screws through the micro:bit and spacer, into the nuts mounted on the PCB.



2 FLASH THE ZIP LEDs

STEP 1: Put batteries into the SERVO:LITE PCB, and turn it on.
STEP 2: Connect it to a computer using a micro-USB cable.
STEP 3: Bring up Javascript Blocks Editor (makecode.microbit.org).

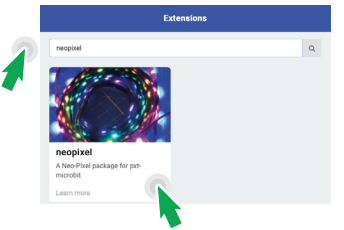
NOTE: Kitronik's ZIP LEDs are compatible with Adafruit's Neopixels.



STEP 4: In the toolbox towards the left of the screen, select the 'Advanced' section. Additional packages should appear below.

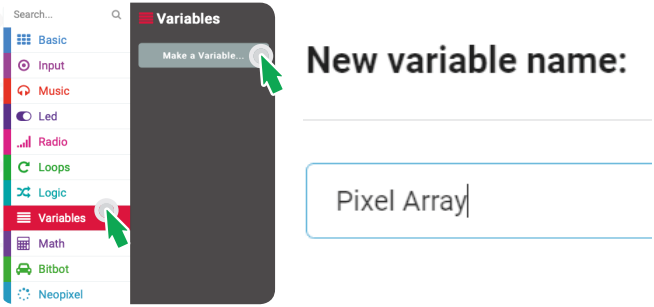
STEP 5: Select 'Extensions'.

STEP 6: In the search bar type 'neopixel', then select the 'neopixel' box.



NOTE: This will load a set of blocks compatible with Kitronik's ZIP LEDs, which makes them really easy to code!

STEP 7: Create a variable and name it 'Pixel Array'.



WHAT THIS MEANS

A variable is like a container which can store information. This could be a number, a word or a piece of information you want your program to remember.

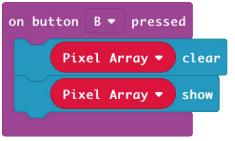
STEP 8: Create the following code.

WHAT THIS MEANS

This tells the micro:bit that Pin 0 is connected to 5, colour addressable LEDs.

WHAT THIS MEANS

When button A is pressed, light up all the pixels red.



WHAT THIS MEANS

When button B is pressed, clear the LED's represented by the variable 'Pixel Array'. This will turn them off.

STEP 9: Download!

The program will automatically run on the simulator. Click on the 'A' button on the simulator to see the LED pattern. Click on 'B' button to clear the pattern.

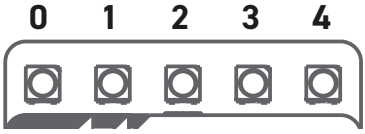
Download the program to generate the .hex file.
Save the .hex file ready to upload to the BBC micro:bit.

Plug the BBC micro:bit into a USB port then drag and drop the .hex file onto the BBC micro:bit window to upload the program.

STEP 10: Press A and get ready for the lights!

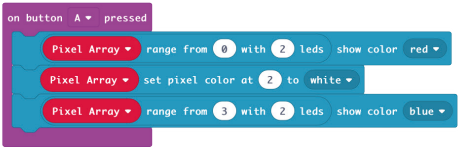
STEP 11: Try changing the code to make a different colour.

As well as addressing all the pixels at once it is possible to set them individually, or as a group. The first pixel always has an address of 0 (see diagram below).



STEP 12: Change the code under 'On button A pressed' to the following.

NOTE: You may need to click on 'More' under the 'neopixel' toolset to view extra blocks.



WHAT THIS MEANS

Set the first two pixels to red, the middle pixel to white and the last two to blue.

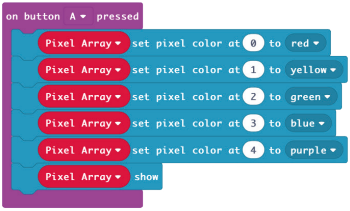
REMEMBER: To show a change you must use a block with 'show' in it.

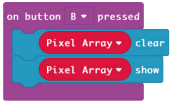
STEP 13:

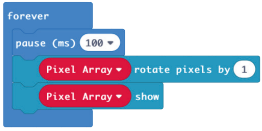
Download → Upload

STEP 14: Press A and watch the lights!

STEP 15: Create the code below.







WHAT THIS MEANS

This code shows a colour changing pattern when button A is pressed and stops when button B is pressed.

NOTE: The 'rotate pixels' block shifts each LED colour onto the next LED. When it reaches the end of the line, it goes back onto to the first LED.

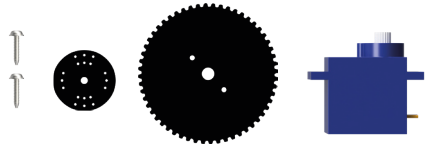
STEP 16:

Download → Upload

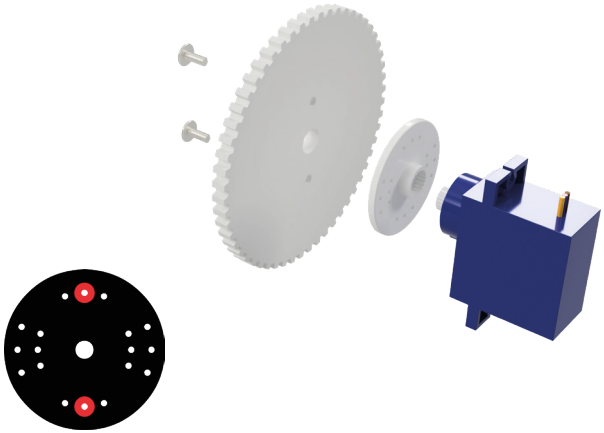
STEP 17: Press A and watch the light show!

3 BUILDING THE WHEELS

2x



STEP 1: Screw the 2 larger screws from the servo packet, through the wheel and into the appropriate holes in the servo attachment. The centre of the horn and the centre of the wheel should be aligned. Press fit the wheel onto the servo and repeat for the opposite side.



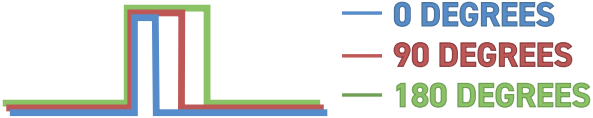
NOTE: The screws should go through the holes shown above in red.

4 TESTING THE SERVOS

SERVOS

The continuous rotation servos used in the :MOVE mini are controlled in the same manner as normal remote control servos. These servos are controlled by a repeating pulse, whose width commands the servo to turn to a position. For a normal servo, position is measured from the output shaft and used to determine what angle the servo should stop at.

NORMAL SERVO



CONTINOUS ROTATION

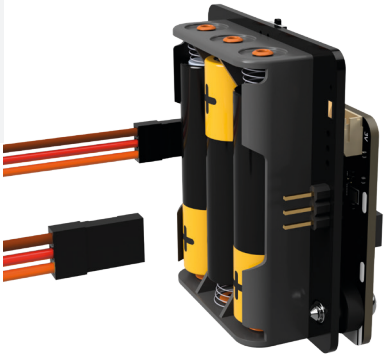
A continous rotation servo is slightly different. Instead of the signal telling the servo how ***far*** to move, it tells the servo how ***fast*** to move.

Because of component tolerances, we may need to set the centre point on the :MOVE mini servos to ensure it will stop when commanded. This is done with a trimmer, which is explained later on.

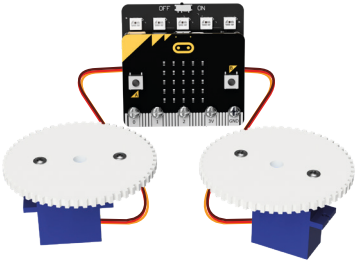
STEP 1: Plug the servos into the Servo:Lite board.

The :MOVE board connections are:

TOP : GROUND : BROWN
MIDDLE : POWER : RED
BOTTOM : SIGNAL : ORANGE

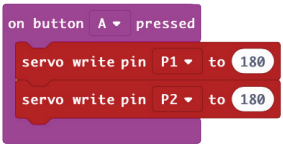


STEP 2: With both wheel servos plugged in, it is time to write some test code. Set out the servos like below. This will allow trimming/calibration of the servos. This means they will stop and travel at the same speed when commanded.

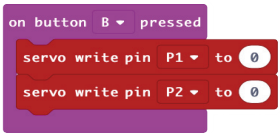


STEP 3: Bring up JavaScript Blocks Editor (makecode.microbit.org).
STEP 4: Create the following code.

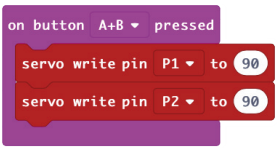
When button A is pressed, both servos should turn anti-clockwise (looking from the wheel side).



When button B is pressed both servos should turn clockwise (looking from the wheel side).



When buttons A + B are pressed the servos should stop turning.

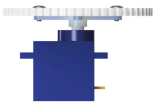


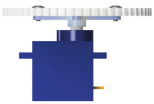
If they do not then the centre point trimmer will need adjustment. On the bottom of the servo, there is a small hole. This is used to access the trimmer.

STEP 5:

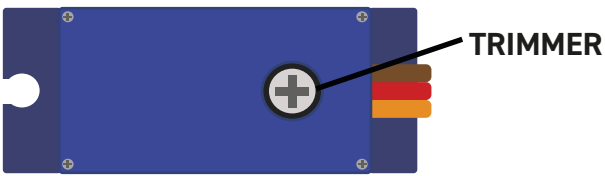
STEP 6: Test out the buttons.

5 CALIBRATING THE SERVOS





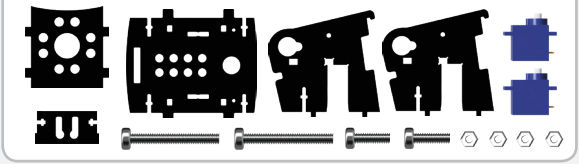
STEP 1: Press buttons A+B,
Then with a small screwdriver (through the hole) gently move the centre point trimmer until the servo completely stops. There should also be no sound coming from the servos.

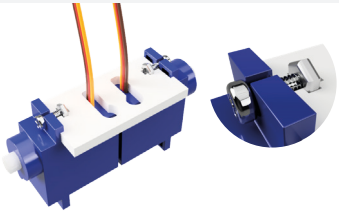


STEP 2: Once the servos are calibrated unplug them from the board and detach the wheels from the servos.

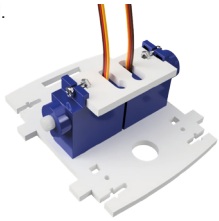
NOTE: The diagram below shows how the number of degrees set in the code relates to the speed of the servo.

6 ASSEMBLING THE CHASSIS

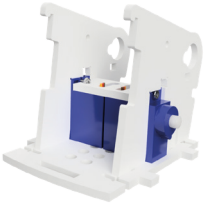




STEP 1: Take the servo cables and feed them through the gaps in the panel shown to the left. Drop an M2 Nut into the T-Joint and secure in place using the shorter M2 bolts provided.

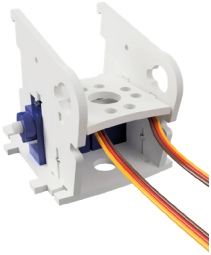


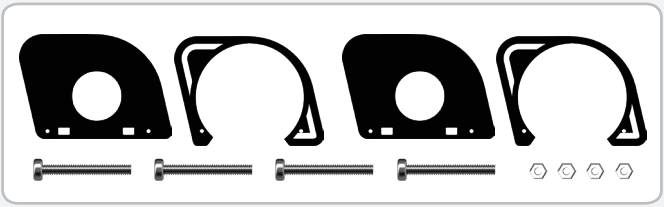
STEP 2: Slot the assembly into the base plate with the wires facing up and the widest section of the plate facing the large circular hole.



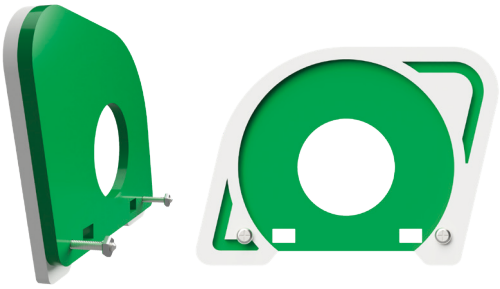
STEP 3: One at a time slot the side panels over the top of the servos, secure in place with an M2 nut and a longer bolt. Ensuring they sit outside the plate attached to the servos.

STEP 4: Snap the pen mounting plate in between the two vertical plates, just above the servo. The servo cables should pass out of the rear of the chassis.



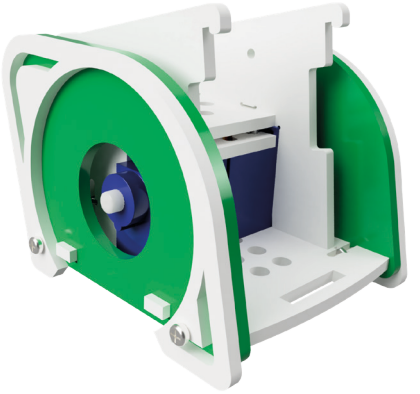


STEP 5: Slot the screws through the outer panel and into the green panel, then add a nut onto the end of each. Repeat for the opposite side.

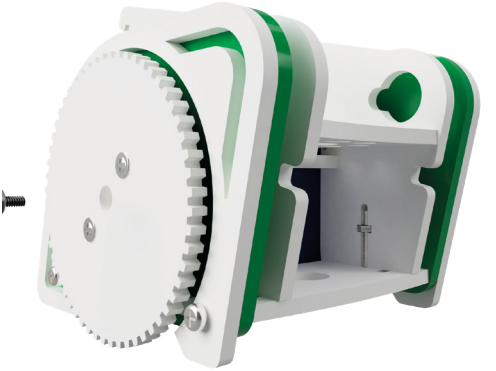


STEP 6: Attach to the chassis, slot the nut into the T-joint and tighten the screw.

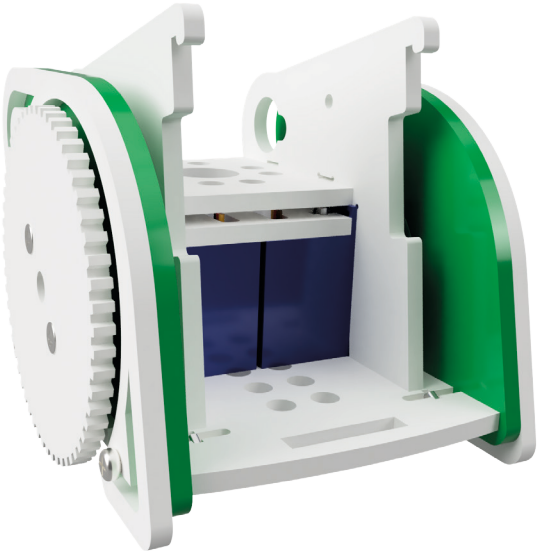
STEP 7: Repeat for the opposite side.



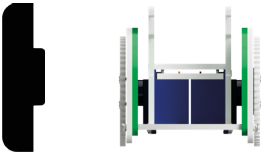
STEP 8: Add a wheel (built earlier) and screw the smallest servo screw through the middle to secure it.

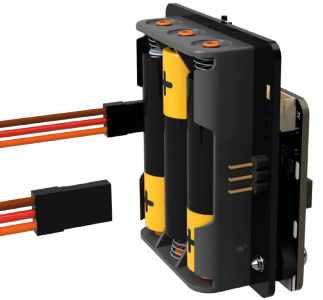


STEP 9: Repeat for the opposite side. The chassis is now complete.



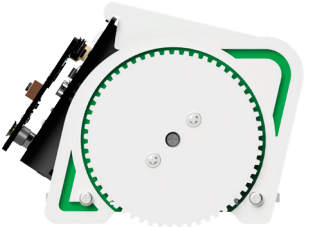
7 ATTACHING ELECTRONICS



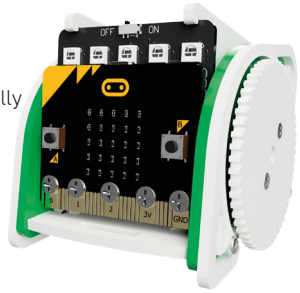


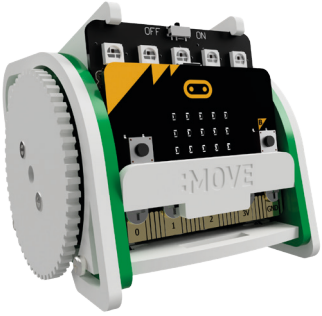
STEP 1: Plug the servo cables back into the Servo:Lite board. The left servo should plug into the left-hand side of the board and vice versa. The cables should be threaded down the gaps on either side as shown below.

STEP 2: Clip the Servo:Lite board under the hooks on the inner side plates and slide it between the outer plates.



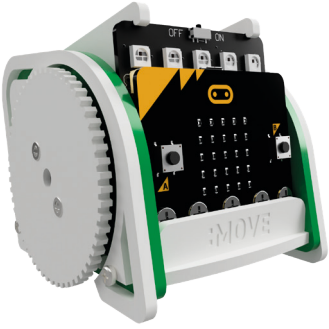
STEP 3: Push the Servo:Lite board fully back inside the :MOVE mini.





STEP 4: Secure the Servo:Lite board and micro:bit with the :MOVE T-piece.

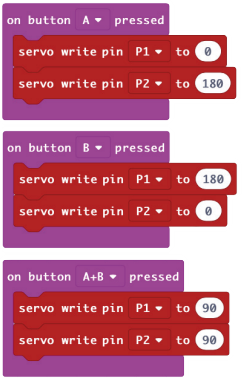
STEP 5: Excess wires can then be fed into the servo compartment at the rear.



8 CODE :MOVE MINI TO MOVE

DRIVE IN A STRAIGHT LINE

STEP 1: Create this code.



NOTE: If the move mini does not travel in a straight line then you can adjust the values to make the servos run at the same speed. You may also need to readjust the trimmer (see pg.17).

WHAT THIS MEANS

When button A is pressed, drive full speed forwards. When button B is pressed, drive full speed backwards.

NOTE: Because the servos are mounted on opposite sides of the :MOVE mini, one has to turn clockwise (Value 0), and one anti-clockwise (Value 180) to drive the :MOVE mini in a straight line.

STEP 2:

DRAW A CIRCLE!

STEP 3: Drop a marker through the designated hole on the pen mounting plate and create the code below.

```
on button A pressed
servo write pin P1 to 0
servo write pin P2 to 90
```

WHAT THIS MEANS

Drive one servo at full speed, and the stop the other. This will cause the :MOVE mini to drive in a circle.

STEP 4:  Download →  Upload

ADD SOME LIGHTS!

STEP 5: Create the following code to give :MOVE mini a head-light or tail-light.

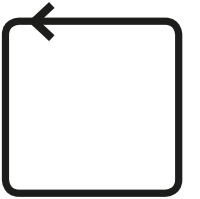
```
on start
set Pixel Array to NeoPixel at pin P0 with 5 leds as RGB (GRB format)

on button A pressed
Pixel Array range from 0 with 2 leds show color white
Pixel Array range from 3 with 2 leds show color white
servo write pin P1 to 0
servo write pin P2 to 180

on button B pressed
Pixel Array range from 0 with 2 leds show color red
Pixel Array range from 3 with 2 leds show color red
servo write pin P1 to 180
servo write pin P2 to 0
```

9 DRAW A SHAPE USING JAVASCRIPT

A square has four equal sides, and four 90 degree corners. Because there are four corners and four edges, it is worth writing some control functions for turning and driving. Otherwise we would have to write the same code four times.



STEP 1: To activate the JavaScript programming environment within MakeCode, click on the 'JavaScript' button towards the top of your browser.


```
let MicroSecInASecond = 1000000
let DistancePerSec = 100
let NumberOfDegreesPerSec = 200
```

These are used to store some constants about the :MOVE mini.

WHAT THIS MEANS

These tell the micro:bit about how fast the servos are moving, they allow easy tuning of the Javascript code.

Next, make a function that takes the desired number of degrees to turn and drives the servos to achieve that. To do this the function will turn on the servos, wait a period, and then turn them off. The following function turns Left through approximately the requested number of degrees.

STEP 2: Add the following code under the previous code.

```
//@param degrees desired degrees to turn through
function TurnLeft(degrees: number): void {
    let timeToWait = (degrees * MicroSecInASecond) / NumberOfDegreesPerSec;
    pins.servoWritePin(AnalogPin.P1, 45);
    pins.servoWritePin(AnalogPin.P2, 45);
    control.waitMicros(timeToWait);
    pins.servoWritePin(AnalogPin.P1, 90);
    pins.servoWritePin(AnalogPin.P2, 90);
}
```

WHAT THIS MEANS

Because we do not have position feedback from the 360 degree servos we have to use time and a knowledge of how fast the :MOVE mini is to make the turn accurately.

STEP 3: Program button A to call this function like this:

```
input.onButtonPressed(Button.A, () => {
    basic.pause(500)
    TurnLeft(90)
})
```

STEP 4:  Download   Upload

NOTE: There is a pause to allow you to move your hand before the :MOVE mini starts turning.

Download this code to the micro:bit and press A, the :MOVE mini should wait half a second before turning through 90 degrees. If :MOVE mini turns too far, or not far enough, you will need to adjust the **'NumberOfDegrees-PerSec' variable** so that the :MOVE mini accurately turns through 90 degrees. If the :MOVE mini turns too far try making the value bigger.

STEP 5: Next make a function to cause the MOVE mini to drive forwards:

```
//This function drives :MOVE mini forwards.
// @param distance how far to drive
function DriveForward(distance: number): void {
    let timeToWait2 = (distance * MicroSecInASecond) / DistancePerSec;
    pins.servoWritePin(AnalogPin.P1, 0);
    pins.servoWritePin(AnalogPin.P2, 180);
    control.waitMicros(timeToWait2);
    pins.servoWritePin(AnalogPin.P1, 90);
    pins.servoWritePin(AnalogPin.P2, 90);
}
```

WHAT THIS MEANS

This code is very similar to the block code used earlier, but includes a timer to stop the :MOVE mini after a certain distance.

STEP 6: Program button B to call this function like this:

```
input.onButtonPressed(Button.B, () => {
    basic.pause(500)
    DriveForward(100)
})
```

NOTE: If the :MOVE mini drives too far/not far enough, adjust the value of the 'DistancePerSec' variable made earlier. If you make it bigger it will drive for less distance.

STEP 7:

Download

Upload

Now we have the building blocks to make the :MOVE mini draw a shape.

STEP 8: Change the code in the 'onButtonPressed(Button.A, () ' function to:

```
input.onButtonPressed(Button.A, () => {
    basic.pause(500)
    DriveForward(100)
    basic.pause(500)
    TurnLeft(90)
    basic.pause(500)
    DriveForward(100)
    basic.pause(500)
    TurnLeft(90)
    basic.pause(500)
    DriveForward(100)
    basic.pause(500)
    TurnLeft(90)
})
```

WHAT THIS MEANS

This will draw a basic four-sided shape, which if your turns are accurate, will be a square.

YOUR TURN: Write code to make the MOVE mini turn right, and drive backwards. Then combine that code to draw other shapes/pictures.

STEP 9:  Download  Upload

10 GO ONLINE!

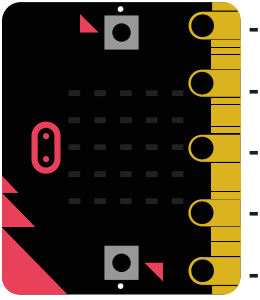
For additional tutorials, add-on packs & resources, scan the QR Code or visit: kitronik.co.uk/movemini

Kitronik have created custom blocks for the MakeCode coding environment, these make it ultra simple to code your :MOVE mini. Give it a try by adding the 'Servo:Lite' blocks from the Extensions tab in MakeCode!

CODE EXAMPLES

- Use your phone as a remote control!
- Use a second micro:bit as a remote control!
- Output to a third servo, such as a bulldozer!
- Pimp the :MOVE mini with extra ZIP LED strips!

:MOVE mini PIN OUT



- 3V
- GROUND
- P2 LEFT SERVO OUTPUT
- P1 RIGHT SERVO OUTPUT
- P0 ZIP LED OUTPUT

TROUBLESHOOTING

If you are having issues with the :MOVE mini, try the steps below!

THE SERVOS ARE MISBEHAVING
- Is the orange servo wire at the bottom of the connector?
- Check all connections are secure.
- Are the screws holding the electronics together tight?
- Review your code, are you outputting to the correct pin(s)?

:MOVE MINI WON'T DRIVE IN A STRAIGHT LINE
- You may need to recalibrate your servos (see page 17).
- Review your code, are you outputting the correct values?

ZIP LED ISSUES
- Check all screw connections are tight.
- Review your code, are you outputting to the correct pin?

POWER ISSUES
- Is it switched on?
- Are the batteries flat?
- Check all screw connections are tight.

DRAWING ISSUES
- Try a felt tip pen, these tend to work best.
- Try adding some weight to the top of the pen.
- Try using a rubber band to secure the pen.

If this hasn't solved your issue, visit kitronik.co.uk/movemini

FOR BBC micro:bit

:MOVE mini

The Kitronik :MOVE mini for the BBC micro:bit provides an introduction to robotics. The mini is a 2 wheeled robot, suitable for both remote control and autonomous operation. A range of add on boards can expand the capabilities to include more advanced functionality.

The :MOVE mini board included in this pack can also be used in conjunction with a BBC micro:bit to build other movement based projects.

Visit kitronik.co.uk/movemini for more details.

T: +44 (0) 845 8380781

W: www.kitronik.co.uk

E: support@kitronik.co.uk

Designed & manufactured in the UK by Kitronik

kitronik.co.uk/twitter

kitronik.co.uk/facebook

kitronik.co.uk/youtube

kitronik.co.uk/google

CE RoHS

For more information on CE please visit kitronik.co.uk/rohs-ce. Children assembling this product should be supervised by a competent adult. The product contains small parts so should be kept out of reach of children under 3 years old. **THIS IS NOT A TOY.**