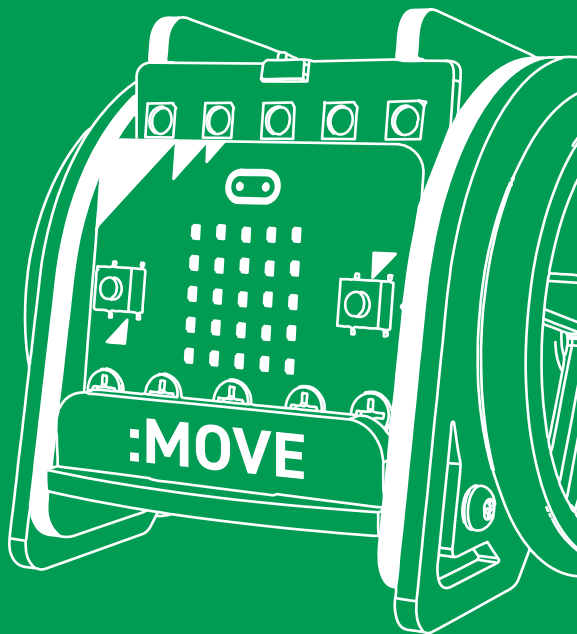# Kitronik

FOR BBC micro:bit

:MOVE
mini Mk2

V1.3

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
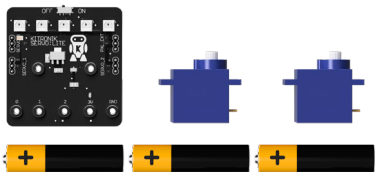
FIXINGS
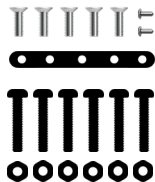
CHASSIS PANELS

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.**

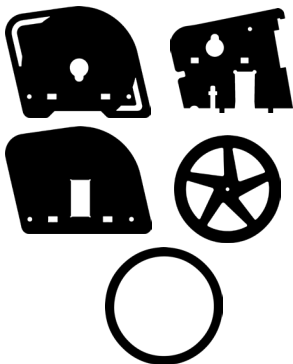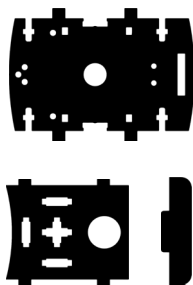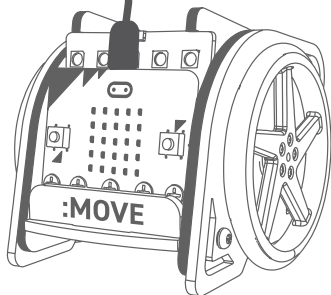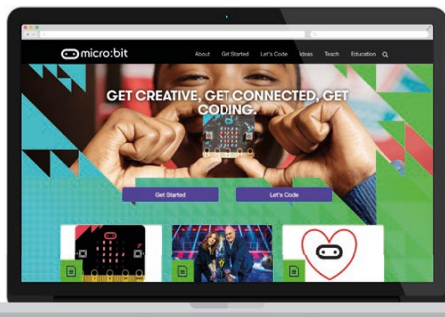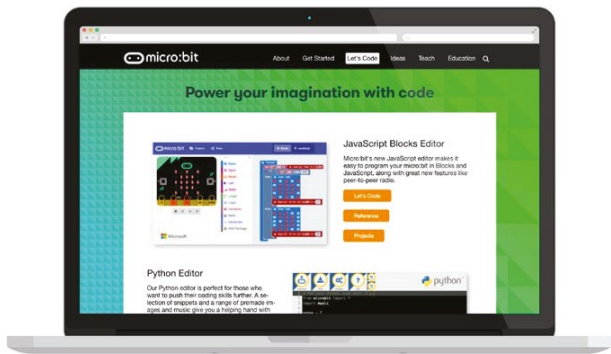To save programs to access at a later date you will need to save the .hex files you create (these files will be explained later in this booklet). To reload a program the relevant file needs to be dragged on to the editor screen. If this is the first time you have used your BBC micro:bit then please refer 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 reset the micro:bit, which can be done by using the switch at the top of the Servo:Lite board to turn it off and back on again.
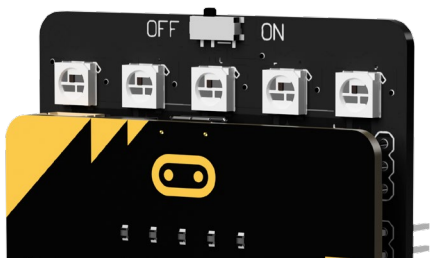
1 ASSEMBLING THE SERVO:LITE PCB
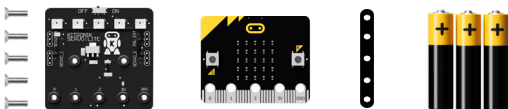
Micro:bit sold separately

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.

ZIP LEDs

Servo Connections

On/Off Switch

PCB Mounted Nuts

Spacer
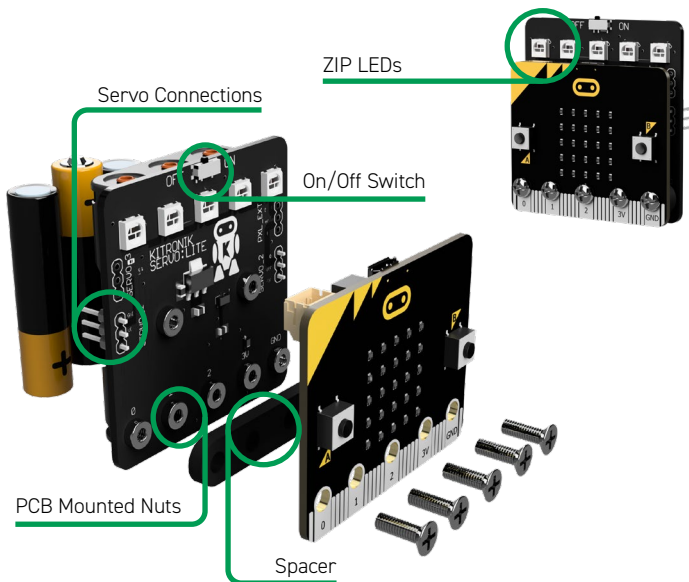
2 FLASH THE ZIP LEDs

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 MakeCode Blocks Editor (makecode.microbit.org).

NOTE: Kitronik's ZIP LEDs are compatible with Adafruit's Neopixels.
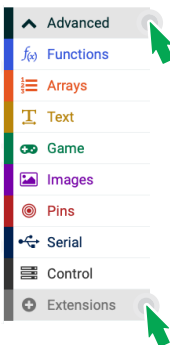
STEP 4: In the toolbox towards the left of the screen, select the 'Advanced' section. Additional packages should appear below.

STEP 5: Select 'Extensions'.

STEP 6: In the search bar type 'neopixel', then select the 'neopixel' box.
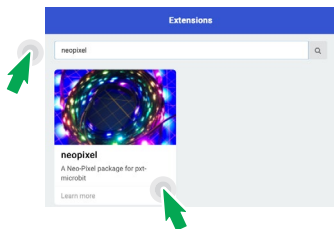
NOTE: This will load a set of blocks compatible with Kitronik's ZIP LEDs, which makes them really easy to code!

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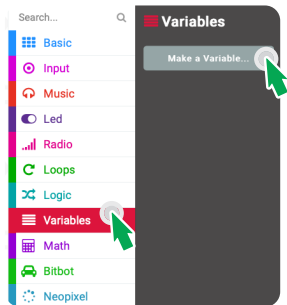
New variable name:

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.


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

WHAT THIS MEANS

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

```
on button A ▾ pressed
    Pixel Array ▾ show color red ▾
```

WHAT THIS MEANS

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

```
on button B ▾ pressed
    Pixel Array ▾ clear
    Pixel Array ▾ show
```
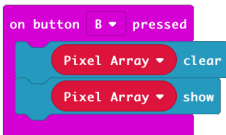
WHAT THIS MEANS

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

STEP 9: Name your program and 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).
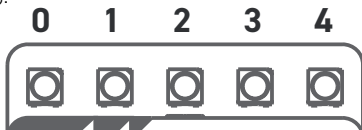
STEP 12: Change the code under 'On button A pressed' to the following.

NOTE: To get the 'range from' blocks, you may need to disassemble the 'set range' block.

You may also need to click on 'More' under the 'neopixel' toolset to access the 'set pixel colour at' block.
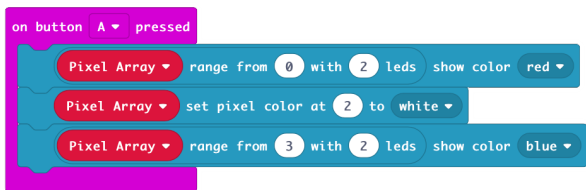
WHAT THIS MEANS

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.


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

```
on button A ▾ pressed
    Pixel Array ▾ set pixel color at 0 to red ▾
    Pixel Array ▾ set pixel color at 1 to yellow ▾
    Pixel Array ▾ set pixel color at 2 to green ▾
    Pixel Array ▾ set pixel color at 3 to blue ▾
    Pixel Array ▾ set pixel color at 4 to purple ▾
    Pixel Array ▾ show
```

```
on button B ▾ pressed
    Pixel Array ▾ clear
    Pixel Array ▾ show
```

```
forever
    pause (ms) 100 ▾
    Pixel Array ▾ rotate pixels by 1
    Pixel Array ▾ show
```

> **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 to the first LED.

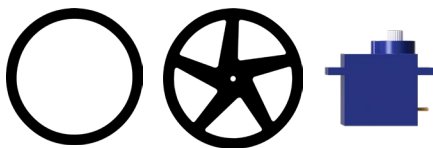**STEP 16:**   [Download]   ⟶   🔌 Upload
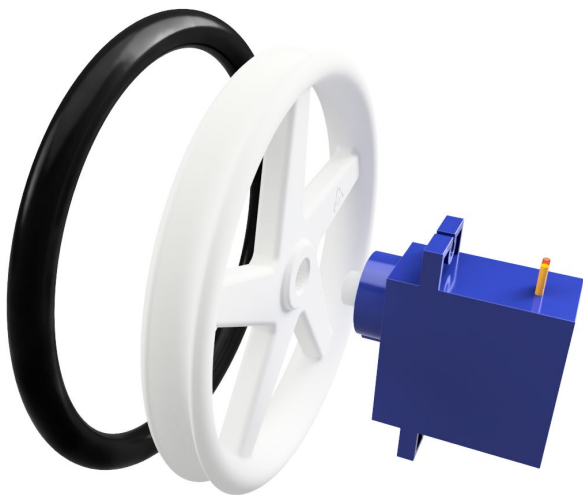
STEP 17: Press A and watch the light show!

3 BUILDING THE WHEELS

2x

STEP 1: Stretch the O-Ring round the wheel, this will act as a tyre for :MOVE mini. Press fit the wheel onto the servo shaft and repeat for the opposite side.

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

— **0 DEGREES**
— **90 DEGREES**
— **180 DEGREES**

CONTINUOS ROTATION SERVO

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

SPEED:

DEGREES: 0   90   180
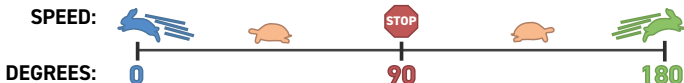
Because of component tolerances, we may need to set the centre point on the :MOVE mini servos to ensure it will stop when commanded. This is done with a trimmer, which is explained later on.

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

The board connections are:

BOARD LOCATION	WHAT IT DOES	WIRE COLOUR
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.
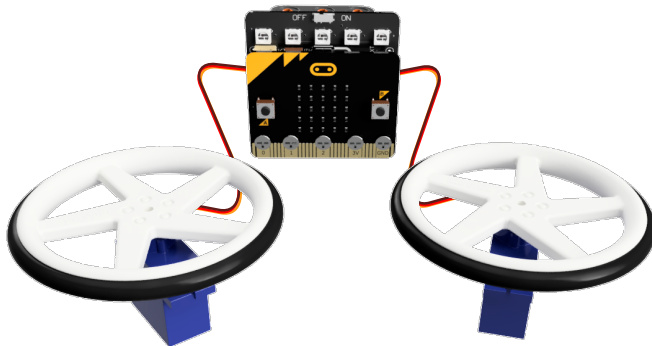
STEP 3: Bring up MakeCode Blocks Editor (makecode.microbit.org).
STEP 4: Create the following code.

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

```
on button A ▾ pressed
    servo write pin P1 ▾ to 180
    servo write pin P2 ▾ to 180
```

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

```
on button B ▾ pressed
    servo write pin P1 ▾ to 0
    servo write pin P2 ▾ to 0
```
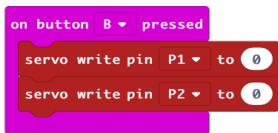
When buttons A + B are pressed the servos should stop turning.

```
on button A+B ▾ pressed
    servo write pin P1 ▾ to 90
    servo write pin P2 ▾ to 90
```

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:

Download ⟶ Upload

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.

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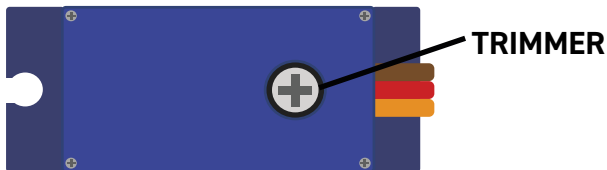
STEP 2: Once the servos are calibrated unplug them from the board and detach the wheels from the servos.

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

SPEED:			STOP		
DEGREES:	0		90		180

6 ASSEMBLING THE CHASSIS
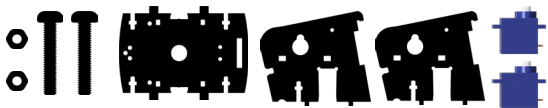
NOTE: During assembly take care not to overtighten the screws as this can damage :MOVE mini.
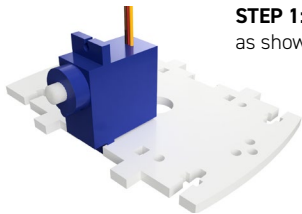
STEP 1: Slot one of the servos into the base plate as shown to the left.
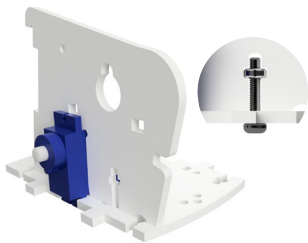
STEP 2: Slot one of the internal side panels over the top of the servos. Drop an M3 Nut into the T-Joint and secure in place with a 16mm bolt. Ensure that the nut is fully inserted into the joint before tightening.
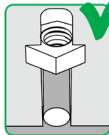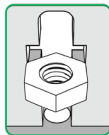
STEP 3: Repeat Step 1 & 2 for the opposite side.
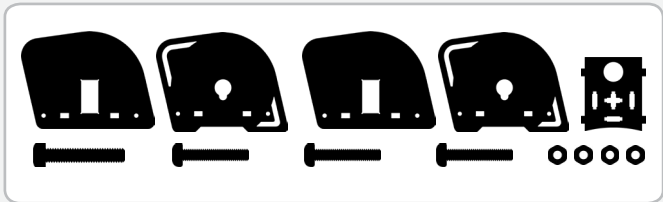
STEP 4: Snap the pen mounting plate in between the two vertical plates, just above the servo. The servo cables should pass out of the rear of the chassis.
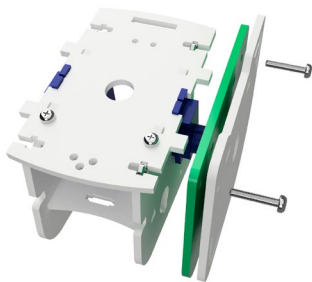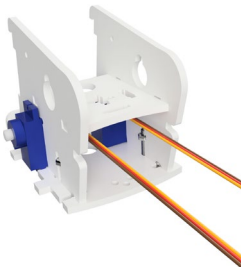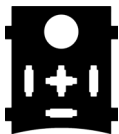
STEP 5: Turn the assembly upside down, align the smaller rectangular holes of the side panels up with the protruding bars on the base plate, and slot them together. Drop M3 nuts into the T-joints, and secure with the 16mm bolts.

STEP 6: Repeat for opposite side.

STEP 7: Add a wheel (built earlier) and screw the smallest servo screw through the middle to secure it to the servo.

STEP 8: Repeat for the opposite side. The chassis is now complete.
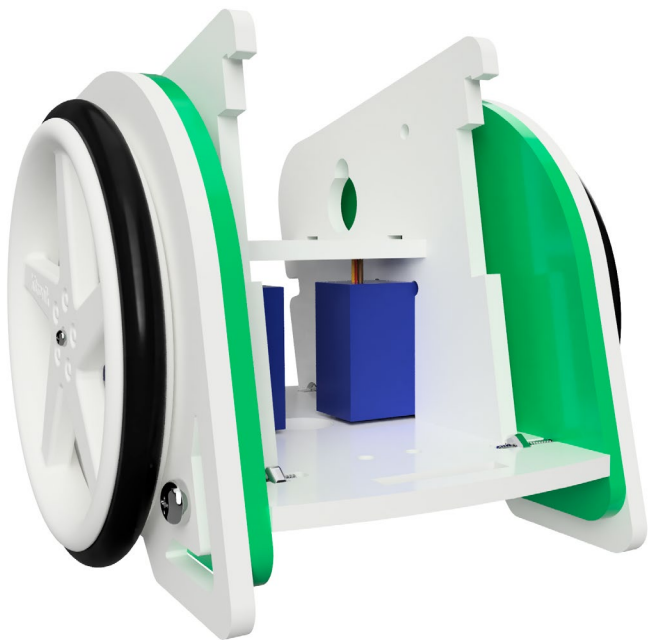
ATTACHING ELECTRONICS
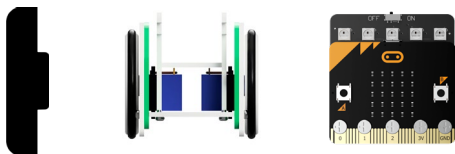
STEP 1: Plug the servo cables back into the Servo:Lite board. The left servo should plug into the left-hand side of the board and vice versa. The cables should be threaded down the gaps on either side as shown below.

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.
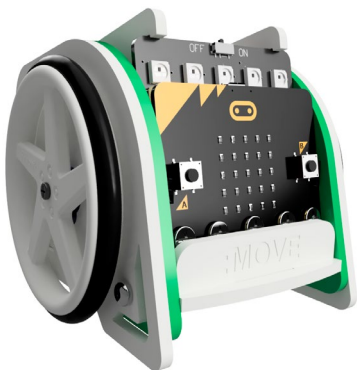
DRIVE IN A STRAIGHT LINE

STEP 1: Create this code.

```
on button A ▼ pressed
    servo write pin P1 ▼ to 0
    servo write pin P2 ▼ to 180
```

```
on button B ▼ pressed
    servo write pin P1 ▼ to 180
    servo write pin P2 ▼ to 0
```

```
on button A+B ▼ pressed
    servo write pin P1 ▼ to 90
    servo write pin P2 ▼ to 90
```

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 re-adjust 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:

Download ⟶ Upload

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
```
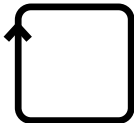
9 DRAW A SHAPE USING JAVASCRIPT

A square has four equal sides, and four 90 degree corners.
Instead of writing the same code four times for both the sides
and the corners, we can save some time by writing control
functions! This means we can write the code once, and then
tell the program to refer back to it as many times as we want.

STEP 1: Click on the 'JavaScript' button towards the top of your browser.
This will activate the JavaScript programming environment, and allow us to
write some code.

```
🖱 Blocks        { } JavaScript
```

STEP 2: Create the following constants. We will use these to store some
information about :MOVE mini.

```
1 let MICROSEC_IN_A_SEC = 1000000
2 let DISTANCE_PER_SEC = 100
3 let DEGREES_PER_SEC = 200
```

VARIABLES EXPLAINED

MICROSEC_IN_A_SEC - The micro:bit's processor counts
time in 'micro seconds'. We want to be counting in 'seconds'
so we will need to convert this. 1sec = 1million microsecs.
DISTANCE_PER_SEC - We will use this to calibrate how far
:MOVE mini *travels* in one second.
DEGREES_PER_SEC - We will use this to calibrate how far
:MOVE mini *turns* in one second.

These will allow us to tune the code to our individual :MOVE mini. For
example, if :MOVE mini turns past 90° when we code 'turn right', we can
adjust these values to correct it easily.

Next we will make a function where we can input a chosen number of degrees and :MOVE mini will output a signal to the servos to turn through the set number of degrees.

To do this, the function will:

1. Turn on the servos,
2. Wait a period of time,
3. Then turn them off.

The 'timeToWait' is calculated by taking the chosen number of degrees, dividing it by how many degrees :MOVE mini turns per second, and then converting the answer to microseconds so the micro:bit can understand it. This value is then output to the servos, resulting in :MOVE mini turning through the chosen angle.

STEP 3: Add the following code under the previous code.

```
5 //@param degrees desired degrees to turn through
6 function turnLeft(degrees: number): void {
7     let timeToWait = (degrees * MICROSEC_IN_A_SEC) / DEGREES_PER_SEC;
8     pins.servoWritePin(AnalogPin.P1, 45);
9     pins.servoWritePin(AnalogPin.P2, 45);
10    control.waitMicros(timeToWait);
11    pins.servoWritePin(AnalogPin.P1, 90);
12    pins.servoWritePin(AnalogPin.P2, 90);
13 }
```

WHY WE NEED TO DO THIS

Because :MOVE mini can't tell how far it has turned, we have to use time and a knowledge of how fast :MOVE mini turns to make the turn accurately.

Now we have made the function, lets use it!

STEP 4: Program button A to call the function we have just created:

```
15 input.onButtonPressed(Button.A, function () {
16     basic.pause(500)
17     turnLeft(90)
18 })
```

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

STEP 5:

Download ➝ Upload

Download this code to the micro:bit and press A, the :MOVE mini should wait half a second (or 500 milliseconds) before turning through 90 degrees.

STEP 6: If :MOVE mini turns too far, or not far enough, you will need to adjust the **'DEGREES_PER_SEC' variable** so that the :MOVE mini accurately turns through 90 degrees.

    - If :MOVE mini turns too far, try making the value bigger.
    - If :MOVE mini doesn't turn far enough, try making it smaller.

Now we've calibrated the turns, lets code driving forwards! This code is very similar to the block of code we used for turning, the only differences are:

- Function name - now called "driveForward', instead of 'turnLeft'.
- degrees - now called distance.
- timeToWait - now called 'timeToWait2'.
- The servo output is now Right0 and Left180 instead of Right45 and Left45.

STEP 7: Add the following code under the previous code.

```
34 //This function drives :MOVE mini forwards.
35 //@param distance how far to drive
36 function driveForward(distance: number): void {
37     let timeToWait2 = (distance * MICROSEC_IN_A_SEC) / DISTANCE_PER_SEC;
38     pins.servoWritePin(AnalogPin.P1, 0);
39     pins.servoWritePin(AnalogPin.P2, 180);
40     control.waitMicros(timeToWait2);
41     pins.servoWritePin(AnalogPin.P1, 90);
42     pins.servoWritePin(AnalogPin.P2, 90);
43 }
```

STEP 8: Program button B to call the 'DriveForward' function, like this:

```
31 input.onButtonPressed(Button.B, function () {
32     basic.pause(500)
33     driveForward(100)
34 })
```

NOTE: The value after 'driveForward' is the distance we are telling :MOVE mini to drive in mm.

NOTE: Similar to before, if :MOVE mini drives too far/not far enough, adjust the value of the 'distancePerSec' variable.
- If :MOVE mini drives too far, try making the value bigger.
- If :MOVE mini doesn't turn drive enough, try making it smaller.

STEP 9: Download ➝ Upload

Now we have the building blocks to make the :MOVE mini draw a shape! To draw a square, we will need to drive forward and turn left four times over.

STEP 10: Under 'onButtonPressed(Button.A, () ' change the code to what is shown below:

```
15  input.onButtonPressed(Button.A, function () {
16      basic.pause(500)
17      driveForward(100)
18      basic.pause(500)
19      turnLeft(90)
20      basic.pause(500)
21      driveForward(100)
22      basic.pause(500)
23      turnLeft(90)
24      basic.pause(500)
25      driveForward(100)
26      basic.pause(500)
27      turnLeft(90)
28      basic.pause(500)
29      driveForward(100)
30      basic.pause(500)
31      turnLeft(90)
32  })
```

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

STEP 11: Download ➝ Upload

10 GO ONLINE!

For additional tutorials, add-on packs & resources, scan the QR Code or visit: kitronik.co.uk/5652

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!
- Personalise the :MOVE mini with extra ZIP LED strips!

:MOVE mini PIN OUT
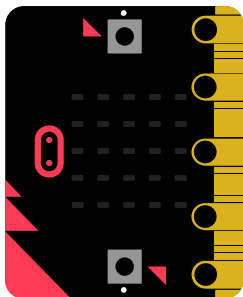
- **GROUND**

- **3V**

- **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/5652 or contact us at
support@kitronik.co.uk.

FOR BBC micro:bit
:MOVE
mini Mk2

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/5652 for more details.

Product Code: 5652

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

CE RoHS