Instructions:

Talking Head Kit

For micro:bit

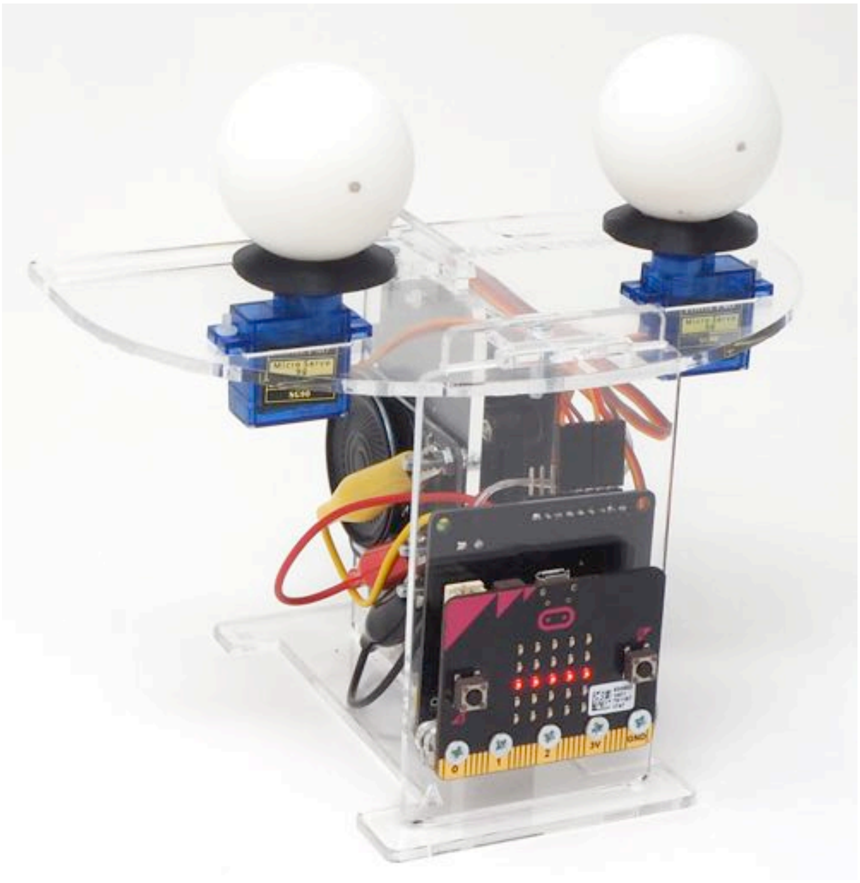


Table of Contents

Introduction

This kit contains everything you need (except a micro:bit and batteries) to make an animatronic head with two moving eyeballs and sound from a loudspeaker.

The kit includes software for three simple projects, but we also hope that you will use this kit to write you own programs and you can also add to the basic head mechanism and build it into something even more fun.

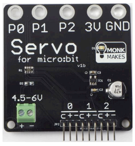
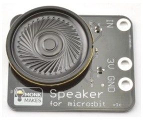
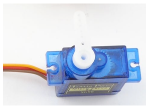
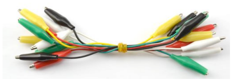
We'd love to see your creations, so please let us know by tweeting to @monkmakes. We are also monk_makes_ltd on Instagram.

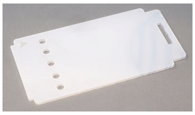
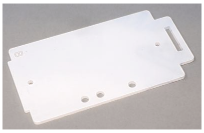
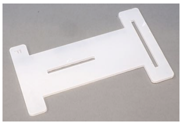
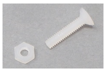
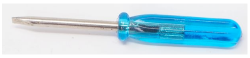
PARTS

Please note that a BBC micro:bit and batteries are NOT included in this kit.

As well as the parts below you will also need **3 x AA batteries,** a **BBC micro:bit** and **a USB lead**.

Before you do anything else, check that your kit includes the following items:

Quantity		
1	MonkMakes Servo for micro:bit board.	 P0 P1 P2 3V GND Servo for micro:bit 4.5-6V 0 1 2
1	MonkMakes Speaker for micro:bit board.	
2	9g micro-servomotors, each with a selection of servomotor arms and screw for attachment. You won't need these extra parts for the kit, but they may come in handy.	
1	3 x AA battery box with push switch for powering the servomotors and your micro:bit.	
1	Set of alligator clip leads (10 leads). You only need 3 for this kit, but we have thrown in some extras.	
2	Ping-pong balls with servo motor connector.	

1	Laser-cut acrylic piece marked A (front)	
1	Laser-cut acrylic piece marked B (back)	
1	Laser-cut acrylic piece marked C (top)	
1	Laser-cut acrylic piece marked F (base)	
2	laser-cut acrylic piece marked D (spacers)	
2	Laser-cut acrylic piece marked E (pegs)	
10	M3 steel nut and bolt pairs	
4	M2 plastic nut and bolt pairs	
1	Small screw-driver	
1	Spanner	

Introduction

Hardware

This kit takes a bit of putting together. There are quite a few steps to follow and parts to fix together. So, while it is designed for ages of 10 and up you may need a bit of adult help from time to time.

Be especially careful with the small nuts and bolts, as these are very easy to lose. When tightening the nuts, use the tiny spanner on one side and a screw-driver on the other, but don't over-tighten the nuts.

The kit is based around four laser-cut pieces of acrylic plastic, which have parts bolted to them. Theses parts include the servomotors that move the eye-balls, a battery box to power everything (including the micro:bit) an amplified loudspeaker and a servomotor controller board.

In the photographs, the acrylic pieces have their white protective films still attached to make it easier to see them. You can carefully peel these off all the pieces of acrylic leaving clear plastic or leave them on. Its entirely up to you.

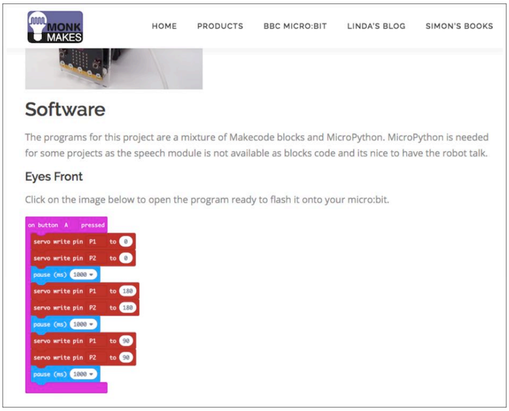
Software

This kit comes with programs for both the MakeCode Blocks Editor and for MicroPython. The kit mostly uses Blocks code. However, one of the programs uses speech that is only available in MicroPython.

The Blocks editor program allows you to program your micro:bit directly from the browser on your computer. All the programs for this kit are listed on a web page for this kit's software at: www.monkmakes.com/mb_head.

Don't worry if you are not sure how to install the programs, this will be described step-by-step in the following pages.

So lets get started!



Software

The programs for this project are a mixture of Makecode blocks and MicroPython. MicroPython is needed for some projects as the speech module is not available as blocks code and its nice to have the robot talk.

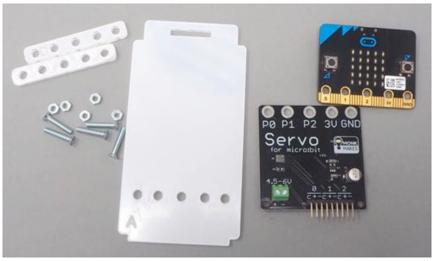
Eyes Front

Click on the image below to open the program ready to flash it onto your micro:bit.

```
on button A pressed
  servo write pin P1 to 0
  servo write pin P2 to 0
  pause (ms) 1000
  servo write pin P1 to 180
  servo write pin P2 to 180
  pause (ms) 1000
  servo write pin P1 to 90
  servo write pin P2 to 90
  pause (ms) 1000
```

Assembling the Head

Assemble the Front Panel



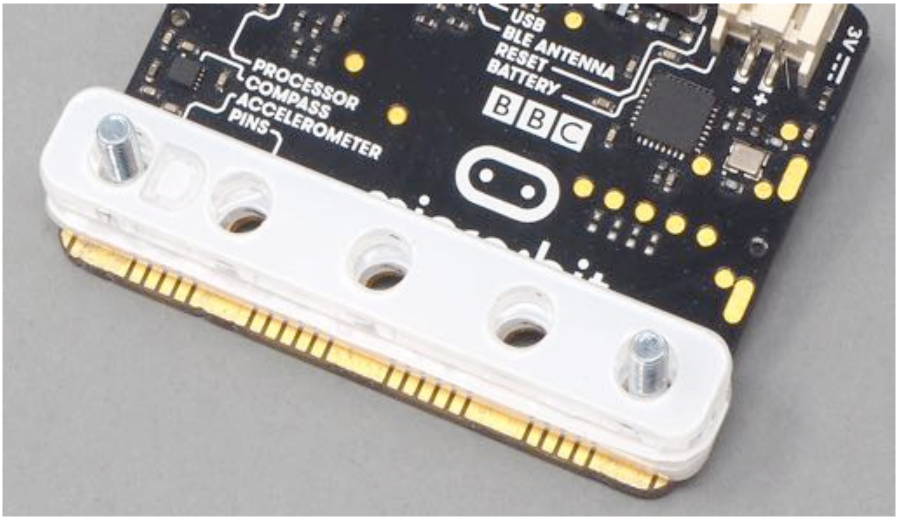
The front panel has the micro:bit on the front side, back-to-back with the Servo for micro:bit board on the reverse. The two are bolted together both to fix the two boards in place and make the necessary electrical connections between them.

Step 1. Start by finding the acrylic piece marked with an **A**, the two spacers marked with a **D**, the Servo for micro:bit board and your micro:bit. You will also need five of the steel nuts and bolts.

The acrylic piece **A** can be either side up.

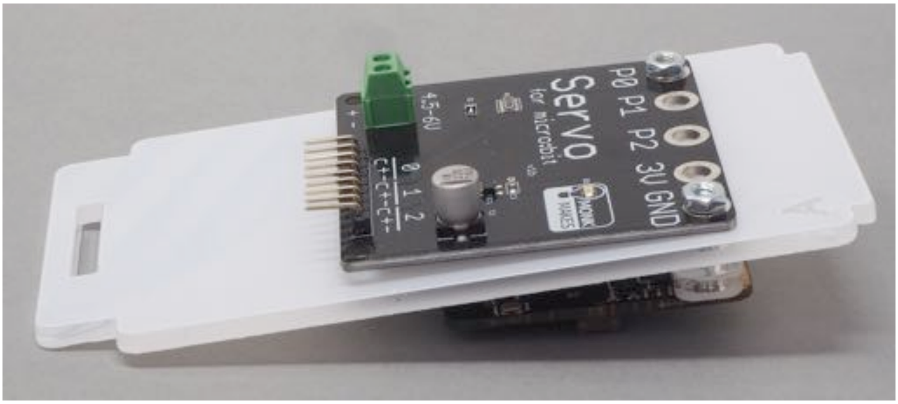
Step 2. Insert two of the bolts through the edge connector holes marked 0 and GND at either end of the micro:bit. Then flip the micro:bit over and place the two acrylic spacer pieces marker **D** into the bolts.

Place the micro:bit, spacers and bolts through the corresponding holes in piece **A**



Step 3. Turn the piece **A** over, put the Servo for micro:bit board over the bolt ends and then put the nuts on to hold everything in place.

Tighten the nuts by holding a nut still with a finger while turning the bolt using the screwdriver.

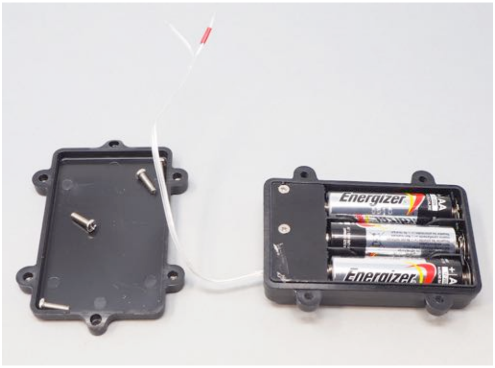


Step 4. Put three more steel bolts through the micro:bit's edge connector holes 1, 2 and 3V and secure with nuts.

Put the finished front panel to one side for now.

Insert the Batteries

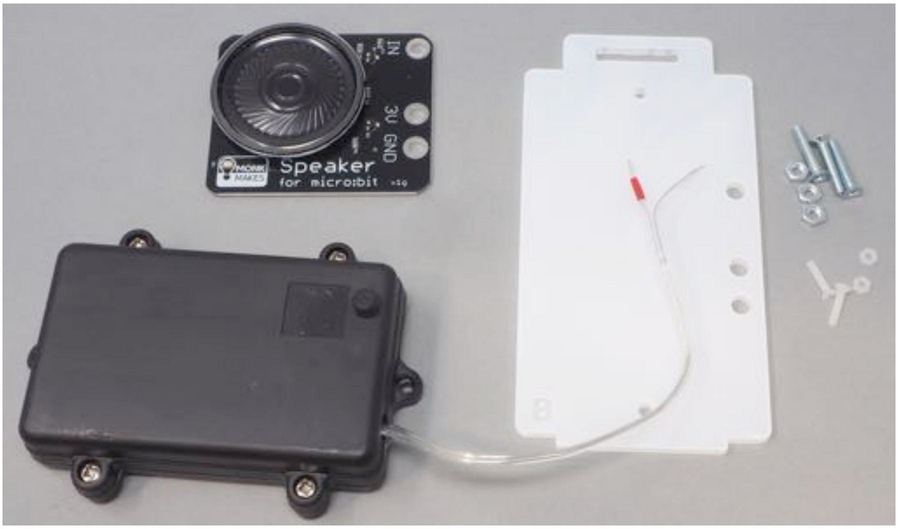
Note. The battery box has a push switch built-in so that you can turn the project on and off. When you push it in, you should be able to feel that the switch has two positions. When its pushed in its *on* and when its out its *off*. Make sure that the switch is in the *off* position because if the bare ends of the two wires accidentally touch, the wires will get hot and the batteries will become empty very quickly.



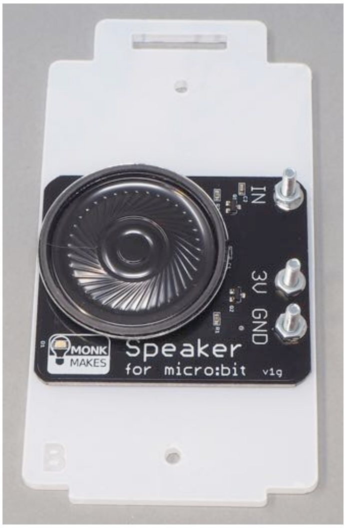
Step 5. Unscrew the four screws that hold the battery box together and remove the lid. Insert three AA batteries into the holder making sure that each battery is the correct way around. Then reassemble the battery box.

Assemble the Back Panel

The back panel holds the MonkMakes Speaker for micro:bit and the battery box.



Step 6. Start by finding the acrylic piece marked with an **B**, the battery box, the MonkMakes Speaker for micro:bit, 3 pairs of steel nuts and bolts and two sets of plastic nuts and bolts.



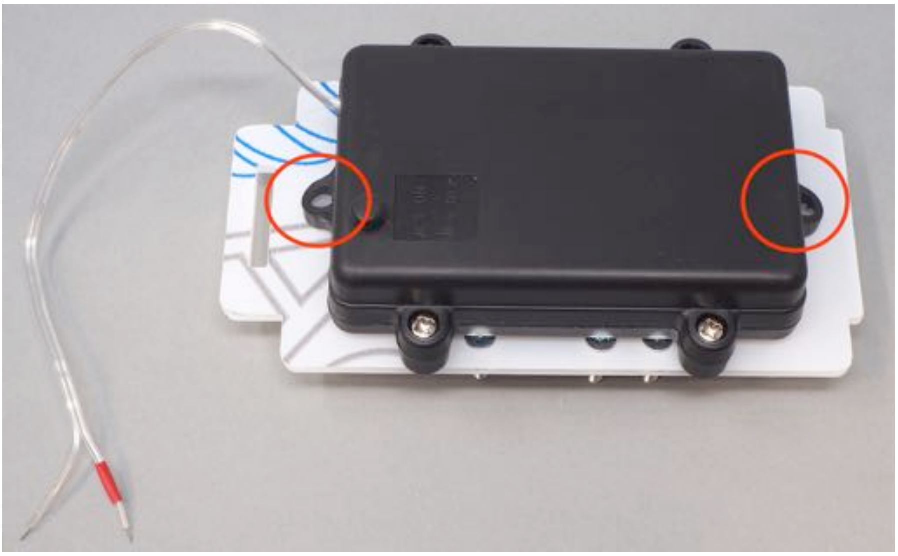
The acrylic piece **B** needs to be the way up shown here, with the three round holes for the Speaker for micro:bit to the right.

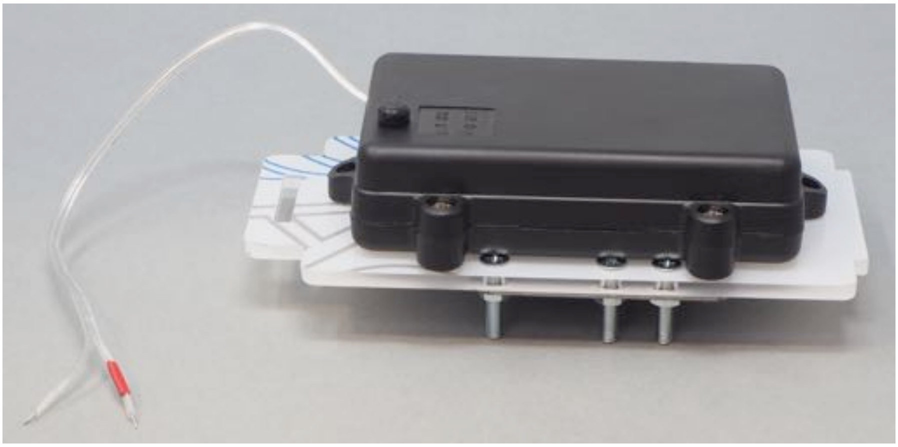
Step 7. Use the 3 steel nuts and bolts to attach the speaker to the acrylic piece **B**.

The nuts should be on the front as shown.

Step 8. Flip piece **B** over and use the plastic nuts and bolts to attach the battery box to it. The bolts should be on the battery box side. Be careful not to over-tighten the nylon nuts. The battery box switch should be at the top of piece **B**.

Set the back to one side, we now need to assemble the top.



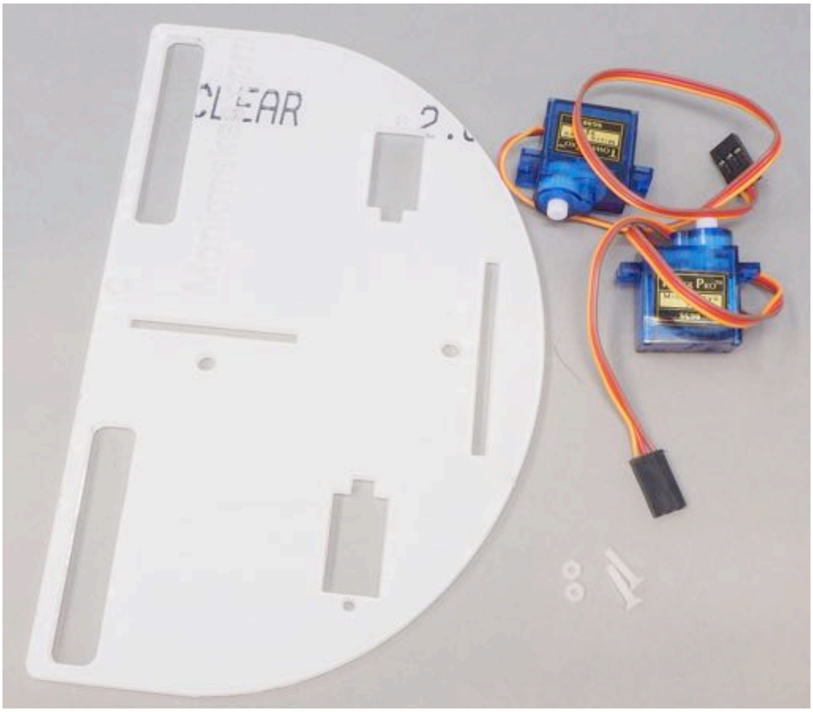


Assemble the Top Panel

The top panel holds the servo motors and links together the front and back panels.

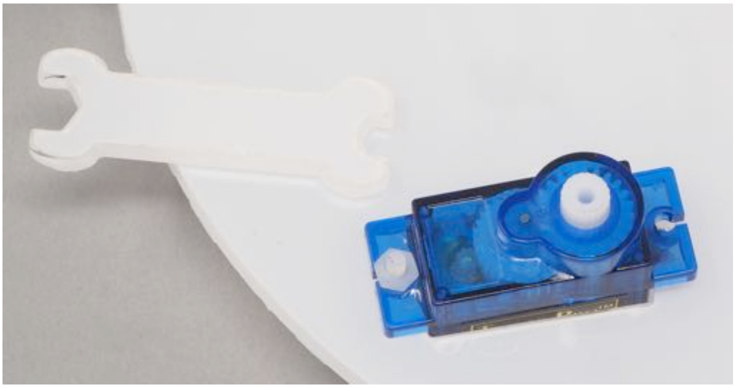
Step 9. To assemble the top panel you are going to need the acrylic piece marked **C**, the two servomotors and two sets of plastic nuts and bolts.

It doesn't matter which way up piece **C** goes.

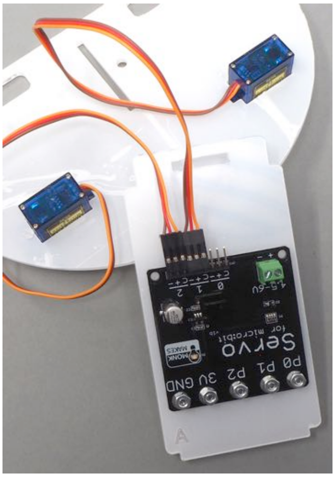


Step 10. Put the lead for one of the servomotors through the rectangular servomotor-shaped hole in **C** and then fit the servomotor into place, with the side of the servomotor where the wires come out towards the middle of **C**. Repeat for the other servomotor and then use the plastic nuts and bolts to fix the motors into place using the servomotor mounting lugs towards the edges of **C**. Its easier to do this with the nuts on the top side. Use the spanner to tighten the nuts or to hold them in place while you tighten the bolt.

Note that each servomotor is only secured by one nut and bolt.



Connecting the Servomotors and Battery Box



Before you assemble the parts of the project you need to make a few connections while everything is easy to get at.

Step 11. Flip piece **C** over as shown here, then connect the socket from the servomotor shown on the left to the MonkMakes Servo for micro:bit board edge pins marked with a **2**.

This socket will fit both ways around, so make sure that when you plug it in, the orange control wire is to the right as shown in the photograph.

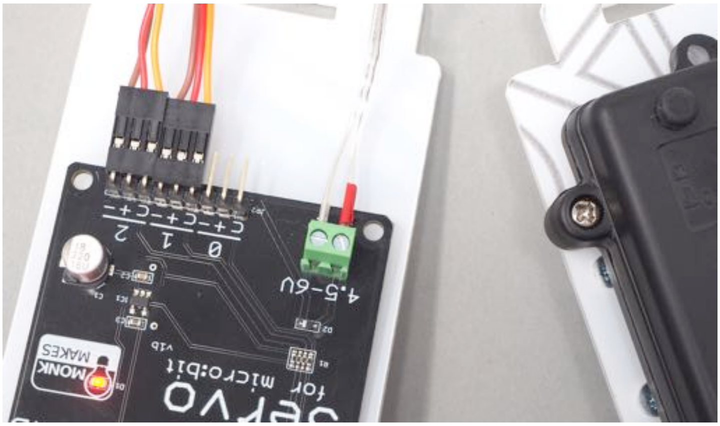
Repeat this for the servomotor shown on the right, but this time to the pins on the Servo for micro:bit board marked **1**. Again, make sure the orange control wire is to the right.

Step 12. Using the small screwdriver, undo (counter-clockwise) the two screws on the green terminal block on the Servo for micro:bit board. Undo them as far as they will go.

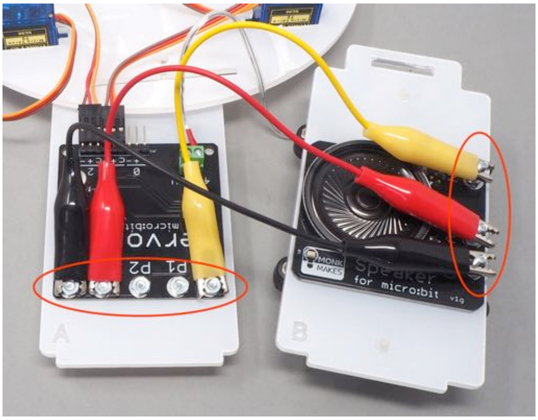
Take the lead marked with red (the positive) wire from the battery box and push the bare metal end of the wire into the hole on the outside of the screw terminal marked **+** and tighten up the screw on the terminal block and check that the lead is being gripped firmly.

Do the same thing for the back wire, but this time connect it to the connection marked with a **-**.

Note that if the orange LED on the Servo for micro:bit board lights up then it means your battery box is on, so press the push button to turn it off for now.



Connecting the Speaker



Step 13. Turn the back panel so that the speaker is face up and attach the black, red and yellow alligator clips. The black lead goes to the nut securing the connection on the Servo board marked **GND**, the red to **3V** and the yellow to **0**.

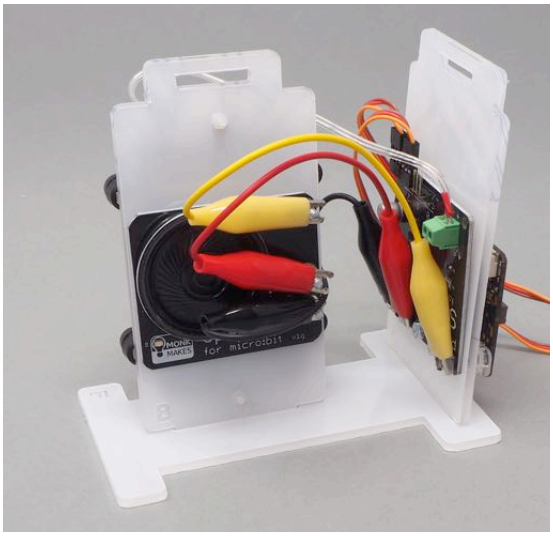
Once you have clipped the lead onto a nut, push the insulating plastic sheath over the metal part of the alligator clip as far as it will go to prevent the metal parts of the clips from touching each other.

Step 14. The other ends of the three alligator-clip-leads connect to the bolts that hold the Speaker in place. The black lead goes to the **GND** connection on the Speaker, the red to **3V** and the yellow to the connection marked **IN**.

Its hard to avoid the alligator leads covering part of the Speaker, but don't worry, this won't make much difference to the sound level.

Fixing the Panels Together

Step 15. Find piece **F**, this is the base. Push the bottom of the front panel into the slot at the front of piece **F**, so that the micro:bit is facing forward.



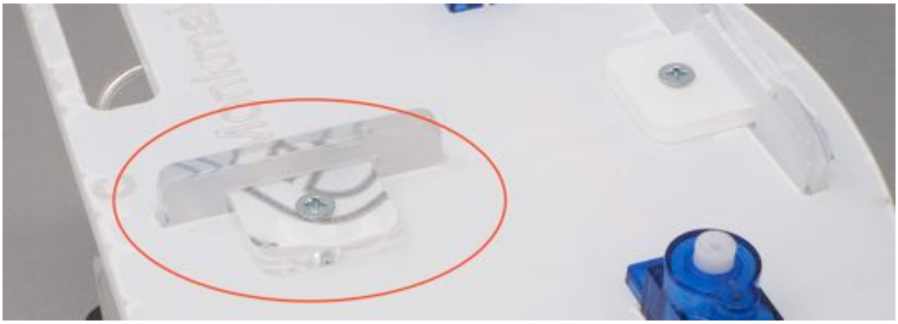
Do the same thing with the back panel, with the battery box towards the right. Note that the pieces only fit in loosely to the base.

Note that the base piece is quite loose, but if you want to make sure it stays put when you lift up the head, you can glue or tape it into place.

Step 16. Place the top piece C over pieces A and B.

Step 17. Next take the two acrylic pieces marked **E** and push them through the gaps in the front and side panels to hold them in place.

Now use the last two steel nuts and bolts to lock the peg pieces (**E**) into place.



In the next section you will be running a program on the micro:bit so that we can take the final assembly step of attaching the eyes, making sure that they face forward!

Attaching the Eyes

Now that you have assembled the chassis for the head, you just need to decorate and attach the eyes, however, you will need to make sure that the eyes are facing forward.

Eyes Forward

To make sure you put the eye's pupils facing straight ahead, you first need to run a test program that will set the servomotors to their center 90 degree position. This will also allow us to check that what we have done so far is working.

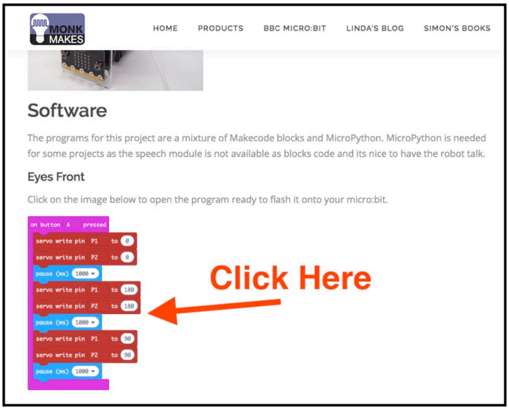
Step 18. Push the eyeballs onto the tops of the servo motors.

Step 19. Connect your micro:bit to a free USB port on your computer.

If your micro:bit is new, then its LED display will start displaying messages and showing off various features of the micro:bit. You are going to replace this program currently running on your micro:bit with a new one.

Step 20. Copy this URL www.monkmakes.com/mb_head into the address bar of your browser.

Scroll down a bit and click on the image of the code for the Eyes Front program. This will then open a new window that shows the program.



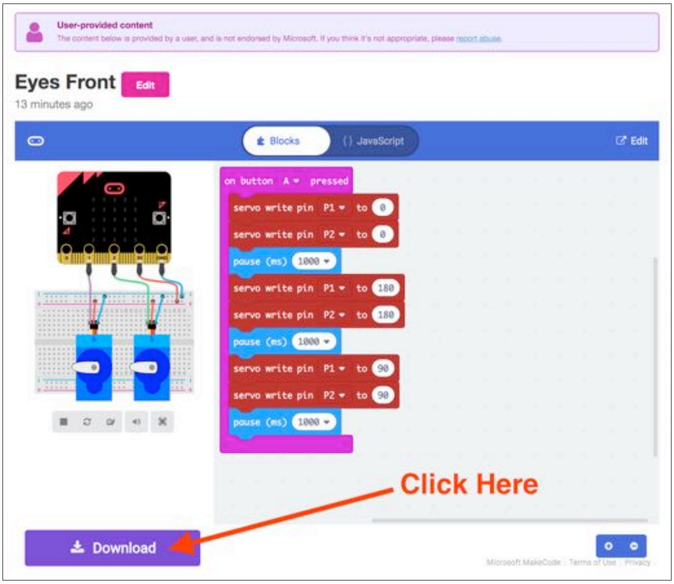
Software

The programs for this project are a mixture of Makecode blocks and MicroPython. MicroPython is needed for some projects as the speech module is not available as blocks code and its nice to have the robot talk.

Eyes Front

Click on the image below to open the program ready to flash it onto your micro:bit.

```
on button A pressed
  servo write pin P1 to 0
  servo write pin P2 to 0
  pause (ms) 1000
  servo write pin P1 to 180
  servo write pin P2 to 180
  pause (ms) 1000
  servo write pin P1 to 90
  servo write pin P2 to 90
  pause (ms) 1000
```



User-provided content

The content below is provided by a user, and is not endorsed by Microsoft. If you think it's not appropriate, please report abuse.

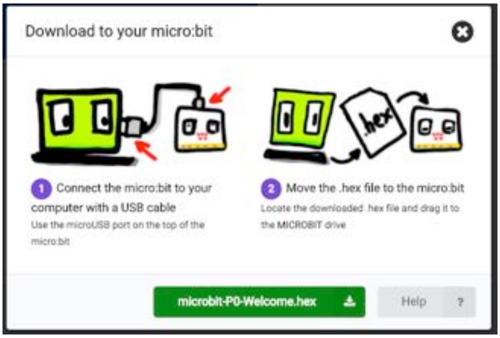
Eyes Front

13 minutes ago

```
on button A ▾ pressed
    servo write pin P1 ▾ to 0
    servo write pin P2 ▾ to 0
    pause (ms) 1000 ▾
    servo write pin P1 ▾ to 180
    servo write pin P2 ▾ to 180
    pause (ms) 1000 ▾
    servo write pin P1 ▾ to 90
    servo write pin P2 ▾ to 90
    pause (ms) 1000 ▾
```

Next, you are going to send the *eyes front* program to the micro:bit.

Click on the “Download” link at the bottom of the page. This will download a file for the program ending in *.hex*. You will also see the prompt below.



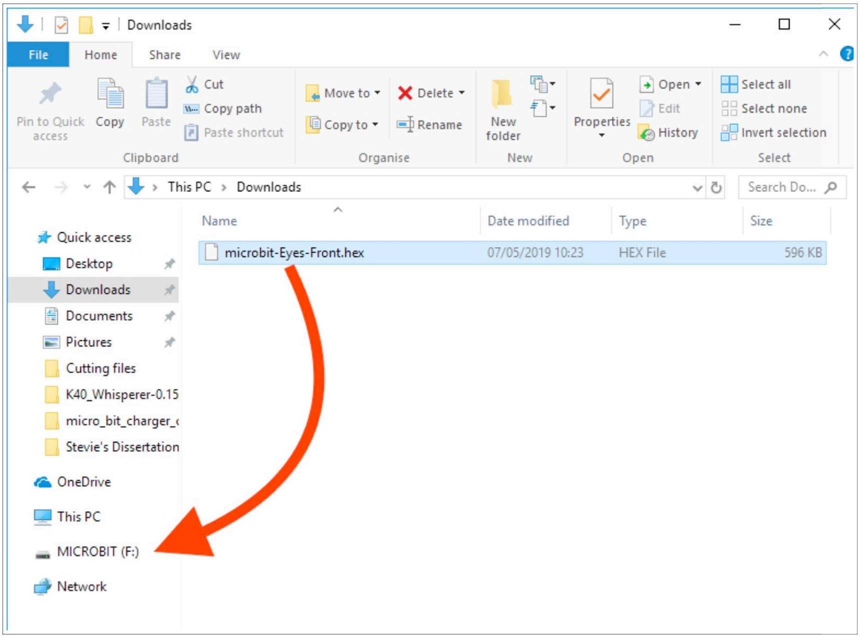
Download to your micro:bit

1 Connect the micro:bit to your computer with a USB cable

Use the microUSB port on the top of the micro:bit

2 Move the .hex file to the micro:bit

Locate the downloaded .hex file and drag it to the MICROBIT drive



Name	Date modified	Type	Size
microbit-Eyes-Front.hex	07/05/2019 10:23	HEX File	596 KB

As the prompt says, this *hex* file needs to be moved onto the micro:bit. The micro:bit will show up as a USB drive on your computer. In other words, you need to drag the hex file from your *Downloads* folder on your computer onto the micro:bit icon in the file explorer just as if the micro:bit were a USB flash drive.

If you are using a Mac, Linux or Raspberry Pi computer the process is the same, but the file explorer tool will be different.

As soon as the *hex* file is dragged onto the micro:bit icon, a yellow LED on the back of your micro:bit will start to flicker furiously as the program is *flashed* onto the micro:bit.

Step 21. Centering the Eyes

Even though you have flashed the program onto your micro:bit, nothing will happen until you press button A on your micro:bit. This will first turn the servo motor position fully one way, then fully the other way. Back to the center position and then stop.

Once movement has stopped and the servomotors are in the center position.

Decorating the Eyes

Step 22. Using a pencil, put a little dot where the pupil should be for the eyes to look straight ahead.

If you don't get it in just the right place, don't worry, pencil marks will rub off the ping-pong ball easily with an eraser.

Step 23.Take the eyes off the servomotors and enlarge the pupils using a permanent marker pen. You can decorate the eyes further if you want, perhaps adding a colored iris around the pupil.

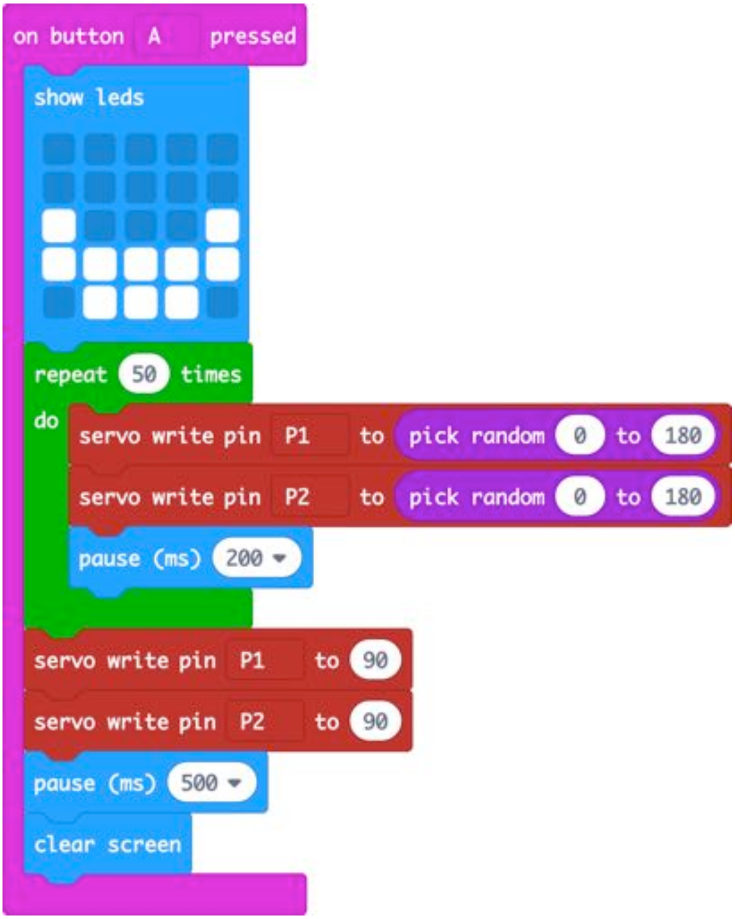
Its a good idea to mark out what you want to do in pencil before you start using permanent marker pen or colored paint.

When you have finished put the eyes back on and press button A again to see the eyes move.

Program 1. Crazy Eyes

This program will wait until you press button A on your micro:bit and then your Talking Head will move its eyes around wildly for about 10 seconds before going back to its resting position, with its eyes forward.

To flash the program onto your micro:bit, connect the micro:bit to your computer using the USB lead and then visit the kit's web page at www.monkmakes.com/mb_head. Scroll down a bit and click on the image of the code for the **Crazy Eyes** program. Then click on the *Download* button and copy the hex file onto your micro:bit as described in Step 20 on page 14.



```
on button A pressed
    show leds
        . . . . .
        . . . . .
        # . . . #
        # # # # #
        . # # # .
    repeat 50 times
    do
        servo write pin P1 to pick random 0 to 180
        servo write pin P2 to pick random 0 to 180
        pause (ms) 200
    servo write pin P1 to 90
    servo write pin P2 to 90
    pause (ms) 500
    clear screen
```

If you find that the servomotors do not move when you press button A, or the servomotors move rather feebly, then make sure that the batteries are connected and the right way around. There is also a Trouble-shooting guide on page 28.

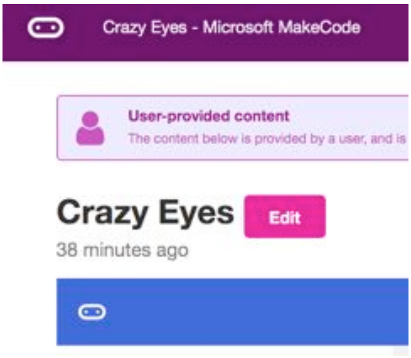
Have a look at the program and see if you can work out how it works.

When button A is pressed, all the blocks inside the `on button A` block will be run in sequence, starting with the `show leds` block that will display an evil grin.

All the blocks within the `repeat 50 times block` will also be run one after the other. In this case the first block to be run (`servo write pin P1`) rotates the left eye (as you look at the head) to a random angle between 0 and 180 degrees.

The `pause (ms)` block then delays (stops the micro:bit doing anything) for 200 milliseconds (1/5 of a second). This gives the servo motors time to physically move from one position to the next, before a new random eye movement is made. This all repeats 50 times.

Once all 50 repetitions have completed, both eyes are set back to their 90 degree (looking forward) position and the screen cleared using the `clear screen` block.


Crazy Eyes - Microsoft MakeCode
User-provided content
The content below is provided by a user, and is
Crazy Eyes
Edit
38 minutes ago

Try editing by clicking on the **Edit** button next to the code. This will make you your own copy of the project that you can change and experiment with.

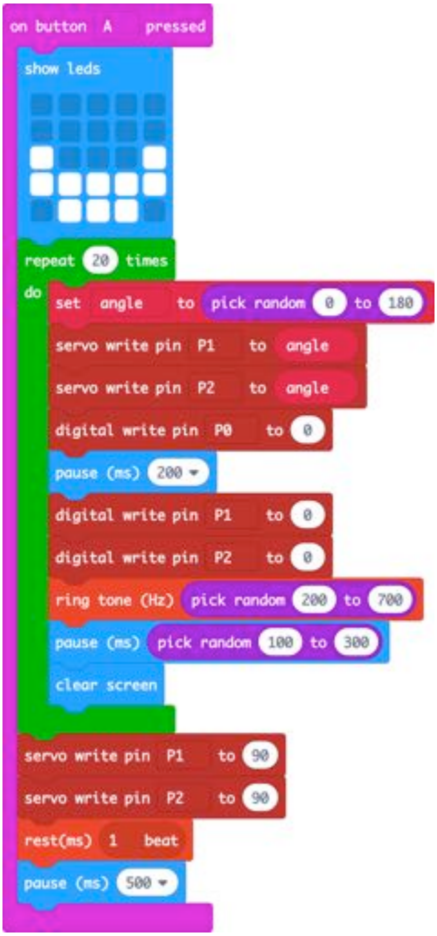
You might like to try changing the pattern of LEDs in the `show leds` block by clicking on them and then clicking the Download button to flash the changed program onto your micro:bit.

See Page 24 for more information on editing Blocks programs.

Program 2. Robot Beeping

This program builds on Program 1 adding sound in the form of random beeping to accompany the eye movements.

To flash the program onto your micro:bit, connect the micro:bit to your computer using the USB lead and then visit the kits web page at www.monkmakes.com/mb_head. Scroll down a bit and click on the image of the code for the **Robot Beeping** program. Then click on the **Download** button and copy the hex file onto your micro:bit as described in Step 20.



```
on button A pressed
    show leds
    repeat 20 times
    do
        set angle to pick random 0 to 180
        servo write pin P1 to angle
        servo write pin P2 to angle
        digital write pin P0 to 0
        pause (ms) 200
        digital write pin P1 to 0
        digital write pin P2 to 0
        ring tone (Hz) pick random 200 to 700
        pause (ms) pick random 100 to 300
        clear screen
    servo write pin P1 to 90
    servo write pin P2 to 90
    rest(ms) 1 beat
    pause (ms) 500
```

This time, inside the repeat 20 times block, we use a variable called `angle`. Variables are a way of giving a name to a value. In this case that value will be a random number between 0 and 180. We do this so that we can set the angle of both eyes to the same angle to make the eyes a bit less crazy.

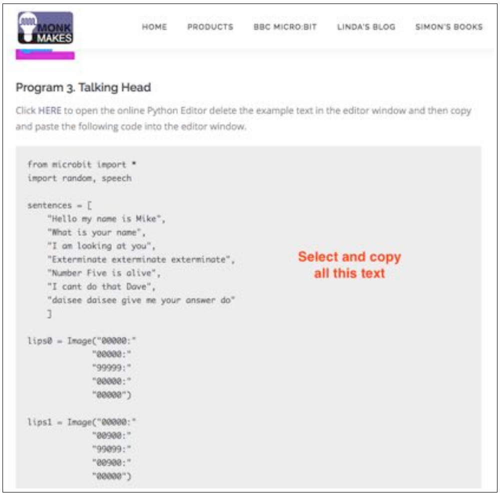
You may have noticed that before the first `pause (ms) 200` block, there is a `digital write` block. This turns off P0 (connected to the speaker). This prevents an annoying buzz from the speaker that happens as a side effect of pins P1 and P2 controlling the servomotors.

Similarly, after the `pause (ms) 200` block there are two more `digital write` blocks that this time turn off the two pins used by the servomotors before the `ring tone (Hz)` block plays a random tone of between 200 and 700 Hz.

This turning off of pins is needed because the micro:bit uses the same hardware for generating sound and controlling the servo motors.

Program 3. Talking Head

This program will make your robot speak, even if it's a bit indistinct and quiet. But with a little concentration and imagination you can tell what it's saying.



Program 3. Talking Head

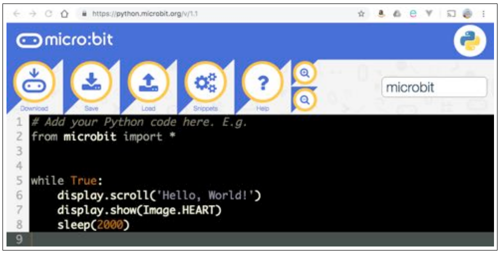
Click HERE to open the online Python Editor delete the example text in the editor window and then copy and paste the following code into the editor window.

```
from microbit import *
import random, speech

sentences = [
    "Hello my name is Mike",
    "What is your name",
    "I am looking at you",
    "Exterminate exterminate exterminate",
    "Number Five is alive",
    "I cant do that Dave",
    "daisee daisee give me your answer do"
    ]

lips0 = Image("00000:"
              "00000:"
              "99999:"
              "00000:"
              "00000")

lips1 = Image("00000:"
              "00900:"
              "99099:"
              "00900:"
              "00000")
```

Select and copy all this text

The Blocks code does not have a mechanism for speech, fortunately this is part of the MicroPython language for the micro:bit.

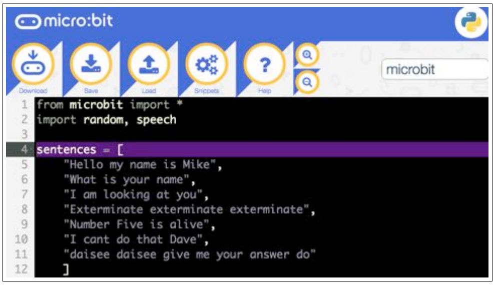
You can find the full program to copy and paste on the kits web page www.monkmakes.com/mb_head. Scroll down to the **Talking Head** section and select and copy all of the text in the grey code listing area.



```
# Add your Python code here. E.g.
from microbit import *


while True:
    display.scroll('Hello, World!')
    display.show(Image.HEART)
    sleep(2000)
```

To get the program onto your micro:bit, you can use the online MicroPython editor that you will find at python.microbit.org. Open this in a second browser tab (right).

Select all the text in the black editor area and replace it by pasting the code that you just copied from the other browser tab.



```
from microbit import *
import random, speech

sentences = [
    "Hello my name is Mike",
    "What is your name",
    "I am looking at you",
    "Exterminate exterminate exterminate",
    "Number Five is alive",
    "I cant do that Dave",
    "daisee daisee give me your answer do"
    ]
```

Now click on the **Download** icon at the top and copy the hex file onto your micro:bit in the same way as if you were using the Blocks editor (see the end of step 20).

This program is triggered by movement, so poke the head's micro:bit and you should trigger eye movement and talking. The head will also pick random moments to say something.

Here's the code.

```
from microbit import *
import random, speech

sentences = [
    "Hello my name is Mike",
    "What is your name",
    "I am looking at you",
    "Exterminate exterminate exterminate",
    "Number Five is alive",
    "I cant do that Dave",
    "daisee daisee give me your answer do"
    ]

lips0 = Image("00000:"
             "00000:"
             "99999:"
             "00000:"
             "00000")

lips1 = Image("00000:"
             "00900:"
             "99099:"
             "00900:"
             "00000")

lips2 = Image("00000:"
             "09990:"
             "99099:"
             "09990:"
             "00000")

lips = [lips0, lips1, lips2]

def set_servo_angle(pin, angle):
    duty = 26 + (angle * 51) / 90
    pin.write_analog(duty)

def speak(sentence):
    words = sentence.split()
    for i in range(0, len(words)):
        display.show(random.choice(lips))
        speech.say(words[i])
    display.show(lips0)

def act():
    angle = random.randint(0, 180)
    set_servo_angle(pin1, angle)
    set_servo_angle(pin2, angle)
    sleep(300)
    speak(random.choice(sentences))
    set_servo_angle(pin1, 90)
    set_servo_angle(pin2, 90)
    sleep(1000)

base_z = 0

while True:
```

You may have noticed that the talking head tends to repeat itself a lot. The phrases that it speaks are contained in the list **sentences**. The speech isn't very clear, but spelling the words phonetically rather than correctly can improve things.

The lip images (**lips0**..**2**) give the robot a moving mouth. The values represent the brightness of each of the LEDs with 0 being off and 9 being full brightness. So, for example **lips0** is a flat line in the middle of the display.

The lip images are kept in a list (**lips**) to make it easy to pick one at random while speaking.

The set_servo_angle function generates pulses to set the servomotor to the angle specified (see *Programming Servos* on page 27).

The **speak** function splits the phrase to be spoken into individual words. Then speaks each word and sets the lip image at random.

The **act** function sets the eyes to a random position, speaks a random sentence and then sets the eyes

```
    new_z = abs(accelerometer.get_z())
    if abs(new_z - base_z) > 20:
        base_z = new_z
        act()
    if random.randint(0, 1000) == 0:
        act()
    sleep(100)
```

facing forward again.
To detect slight movements, the base acceleration in the z dimension is measured and if it changes by 20 units **act** is called.
The second **if** statement calls **act** at random, 1 time in 1000.

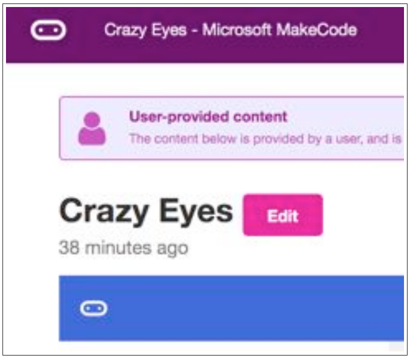
Programming in Python is a step up from Blocks programming. If you want to learn how to write your own programs in Python then visit microbit.org/guide/python/.

If you prefer learning from a book, then you should find my book useful, especially if you are completely new to programming.

Programming the BBC micro:bit
Getting Started with MicroPython
Simon Monk

The Blocks Editor

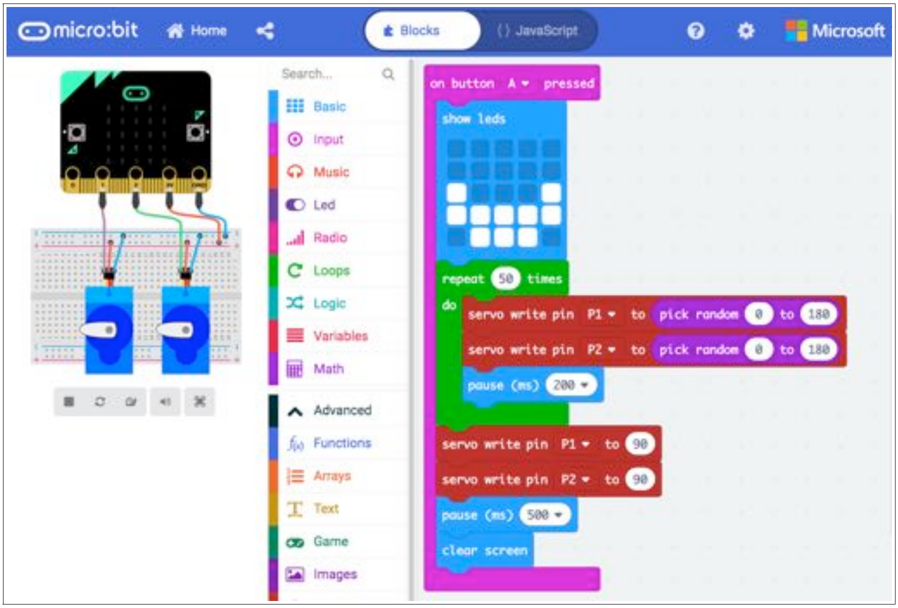
If you want to modify one of the Block programs for the kit, you can do so by clicking on the **Edit** button for any of the Block programs used in this kit.


Crazy Eyes - Microsoft MakeCode
User-provided content
The content below is provided by a user, and is
Crazy Eyes
Edit
38 minutes ago

Clicking on the **Edit** button for one of the projects will open the project in the Blocks Editor.

You will not be editing the original Blocks program, but rather your own copy that you can do what you like with.

For example, clicking on the **Edit** button for the Crazy Eyes program will open up an editor window looking like this:



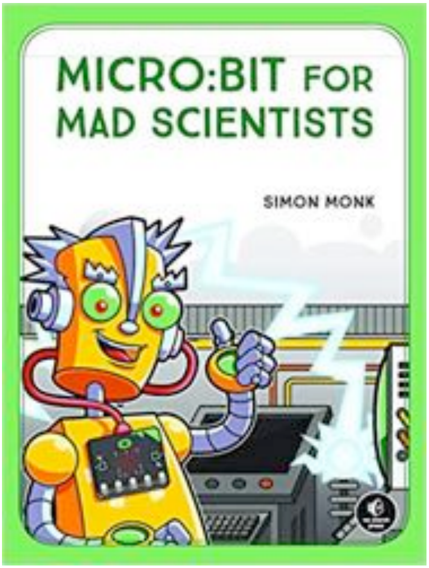
```
on button A ▾ pressed
    show leds
    repeat 50 times
    do
        servo write pin P1 ▾ to pick random 0 to 180
        servo write pin P2 ▾ to pick random 0 to 180
        pause (ms) 200 ▾
    servo write pin P1 ▾ to 90
    servo write pin P2 ▾ to 90
    pause (ms) 500 ▾
    clear screen
```

One really nice feature of the Blocks editor is that the image of a micro:bit on the left of the screen is a virtual micro:bit, that you can run your programs on before flashing them onto a real micro:bit. In this case, the virtual micro:bit even has two servo motors attached. You can press its buttons with your mouse and it will be as if you pressed button A on an actual micro:bit. The middle section of the screen has different categories of blocks: *Basic*, *Input*, *Music* etc. Here you can find blocks to put onto the right-hand ‘canvas’ area. You can also use the *Search* box just above the list of block categories if you are not sure where to find the block that you want.

The Blocks editor actually generates code in the JavaScript programming language and if you click on the button *{} JavaScript* you can see the code that has been generated. Clicking on *Blocks* takes you back to the blocks view of things, which is much easier to follow.

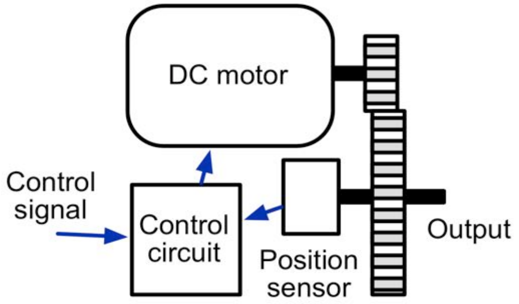
For tutorials and lots of other useful links to learn Blocks coding, visit microbit.org/code and then click on the link for **Lessons**.

This kit is inspired by one of the projects in the book below. So, if you enjoyed this kit, you might well enjoys the projects in this book.



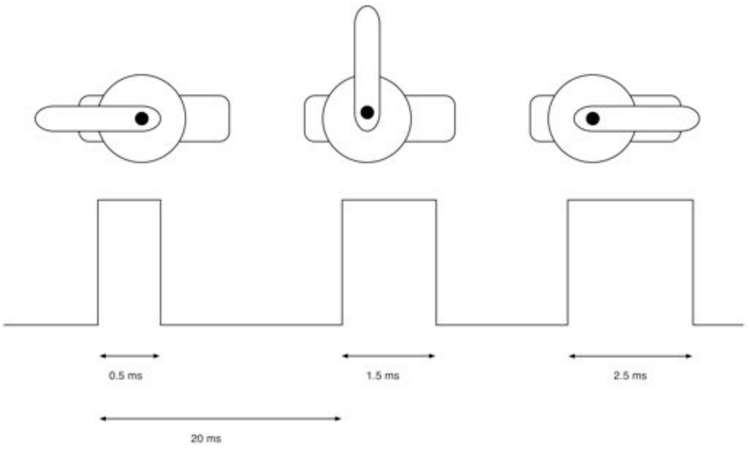
SERVOMOTORS

Despite their small size, servomotors are surprisingly complex little machines, controlled using signal pulses. Here's a diagram of how a servo motor works.



A DC motor (normal motor that can rotate continuously) is attached to a gear box. This drives the servo arm, but is also coupled to a position sensor that provides feedback to an electronic control circuit that keeps the servo arm at the correct angle.

Control of the position uses a series of pulses arriving every 20 milliseconds.

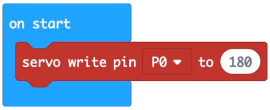


The length of each of these pulses will determine the position of the servo arm. So a short pulse of just 0.5 milliseconds will put the arm at one end of its travel. A pulse of 1.5 milliseconds will put the arm at its center position, and a maximum pulse length of 2.5 milliseconds will put it at the other end of its travel.

PROGRAMMING SERVOS

When it comes to programming your micro:bit to make the servos move, you can use either Block code, as we just have, or you can use MicroPython.

Blocks code



The only block you need to control a servomotor is the `servo write pin` block that you will find in the **Pins** category of the Blocks editor.

The block contains a drop-down list, where you specify which servomotor you want to change the arm position of. The value of 180 is the angle of the servo arm in degrees. The angle can be set to any value between 0 and 180 degrees. Where 0 and 180 are opposite ends of the arm's position and 90 is the center position.

Once you have used the servo write block on a particular pin, the servomotor will stay in that position until you set it to a different position.

MicroPython

Controlling servomotors using MicroPython is more complicated. There is no built-in function for setting a servo to a particular angle. The following example includes a function called **set_servo_angle** that does the same job as the `servo write` *block*.

The example below allows you to control the angle of a servo attached to pin 2, using the A and B buttons on the micro:bit. Pressing button A will decrease the servo angle by 10 degrees and pressing B will increase it by 10.

```
from microbit import *

def set_servo_angle(pin, angle):
    duty = 26 + (angle * 51) / 90
    pin.write_analog(duty)

angle = 90
set_servo_angle(pin2, angle)

while True:
    if button_a.was_pressed() and angle >= 10:
        angle -= 10
        set_servo_angle(pin2, angle)
    if button_b.was_pressed() and angle <= 170:
        angle += 10
        set_servo_angle(pin2, angle)
```

If you want to create your own programs in MicroPython for controlling a servomotor, I suggest you start by copying the **set_servo_angle** function.

Troubleshooting

Problem: The servomotors don't move, or only move a little, when the micro:bit is connected by USB.

Solution: This probably means that the Servo for micro:bit board is not getting power. A micro:bit that is only powered by USB does not provide enough current to power the two servomotors. Check the batteries, both that they are the right way around and that they are not depleted. If this doesn't work, check that the servomotor connecting sockets are connected to the Servo for micro:bit board the correct way around (see Step 11).

Problem: No sound is coming from the speaker when I run Programs 2 and 3.

Solution: Check that the three alligator clips from the micro:bit to the Speaker for micro:bit board are ok. See Steps 13 and 14.

Problem: When I set the servo angle to 0 or 180 the servo judders.

Solution: This is quite normal. Many servomotors will judder when they get to the ends of their range of angles. If you get this problem, then try reducing the range of angles that you try to set your servomotor to. Perhaps 10 to 170 rather than 0 to 180.

Problem: When I move the servo angle between 0 and 180 in the software, the actual angle moved my the servo is not exactly 180 degrees.

Solution: This is also quite normal. Most servomotors do not exactly match the angle the are commanded to by their control pulses. See also the previous problem.

For other support questions, please check out the product's web page (www.monkmakes.com/mb_head) and you can send support questions to support@monkmakes.com

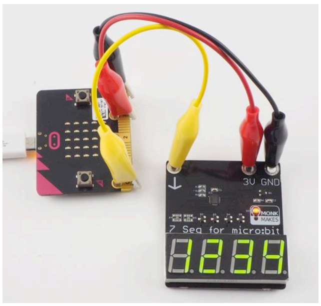
NOTES

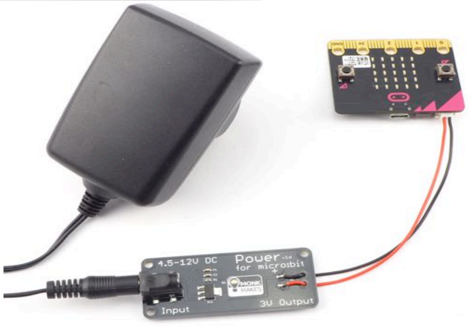
MonkMakes

For more information on this kit, the product's home page is here: monkmakes.com/mb_head.

As well as this kit, MonkMakes makes all sorts of kits and gadgets to help with your maker projects. Find out more, as well as where to buy here: www.monkmakes.com you can also follow MonkMakes on Twitter @monkmakes and Instagram where we are monk_makes_ltd.

Electronics Starter Kit for micro:bit
MONK MAKES
Relay Board
Sensor Board for micro:bit
Speaker
AA





From left to right: Electronics Starter Kit for micro:bit, Power for micro:bit (power adapter not included) and a 7 Segment for micro:bit.