Programming the BBC micro:bit with MicroPython

By: Wang Siyin, Pearlyn Loh, Liu Zixin, Jin Zilong


Google
Ruby doc

Overview

- **Introduction to BBC micro:bit**

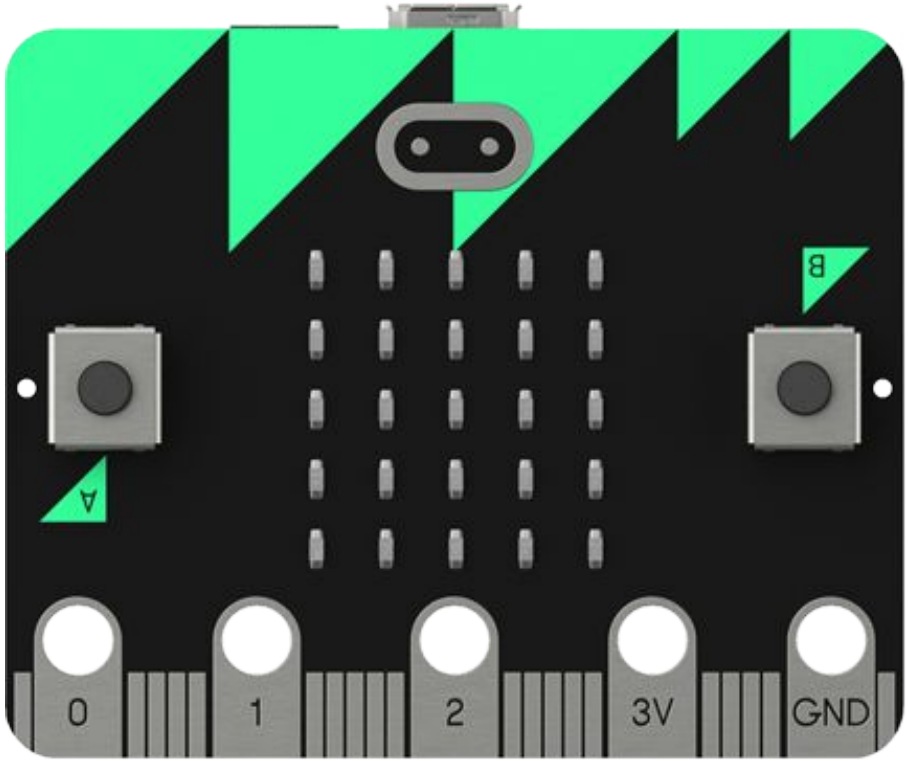
- **Basic Components of micro:bit**
  - Buttons
  - LED Display
  - Accelerometers
  - Radio Module

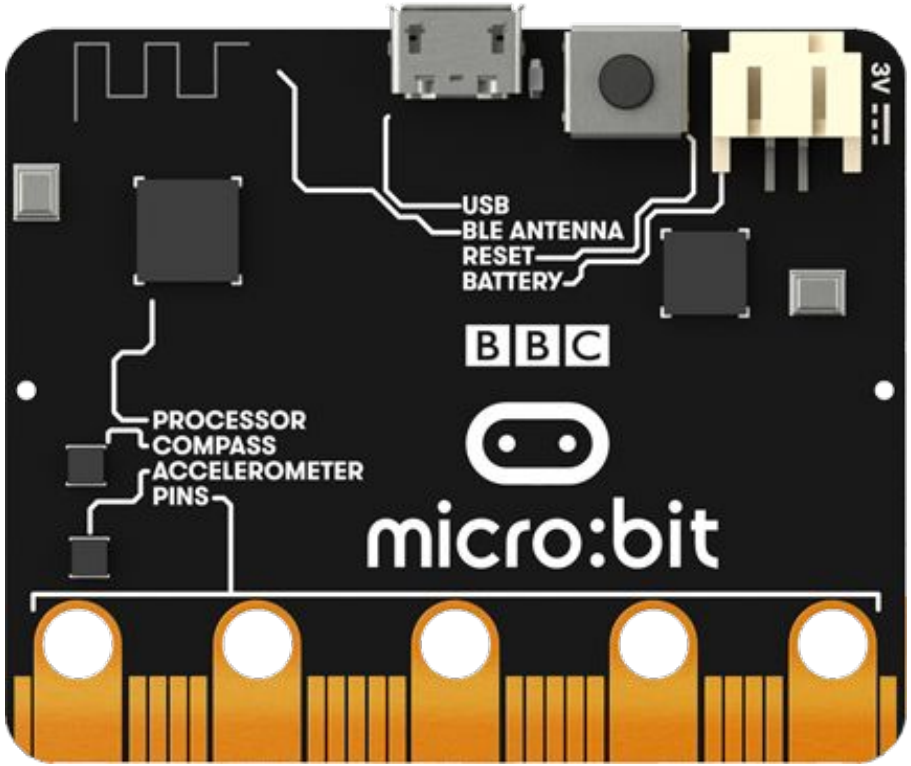
  → **Demo**

  → **Demo**

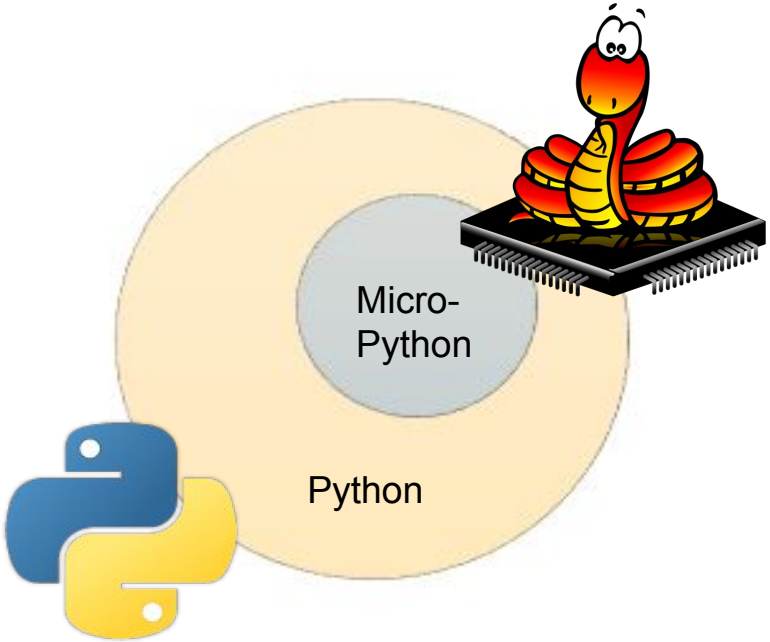
- **Accessories & Add-ons** → **Demo**


BBC
It floated up to heights
of more than 32km


A
B
0
1
2
3V
GND


USB
BLE ANTENNA
RESET
BATTERY
3V
BBC
micro:bit
PROCESSOR
COMPASS
ACCELEROMETER
PINS

Introduction to micro:bit


Micro-
Python
Python

Let’s Start Coding!!

Official Micro:bit

http://microbit.org/code/

Offline Alternative:

https://codewith.mu/#download

With Emulator:

https://create.withcode.uk/

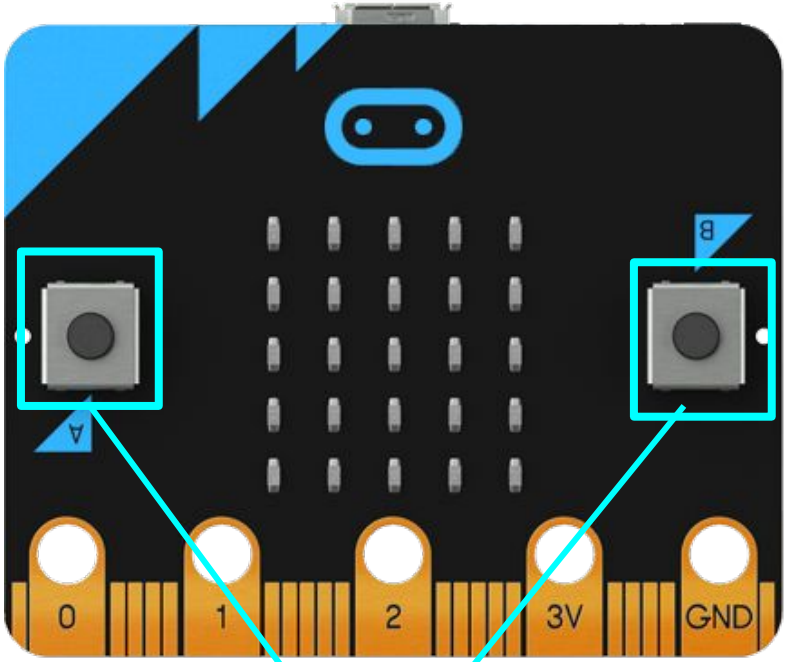
Today!

https://github.com/Pyconmicrobit2017

1

Basic Components of micro:bit

- Buttons
- LED Display
- Accelerometers
- Radio Function



Buttons

Methods

`is_pressed()`

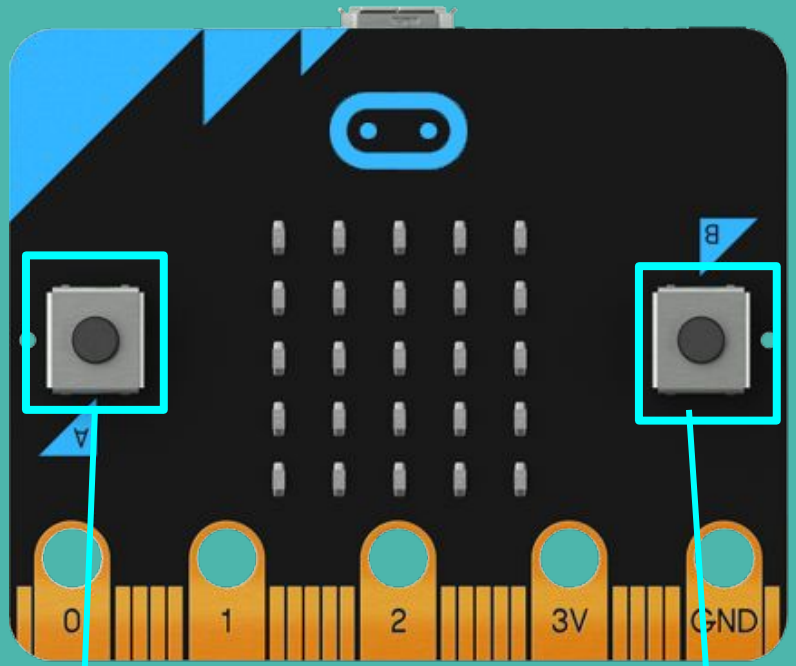
Returns `True` if the specified button `button` is pressed, and `False` otherwise.

`was_pressed()`

Returns `True` or `False` to indicate if the button was pressed since the device started or the last time this method was called.

`get_presses()`

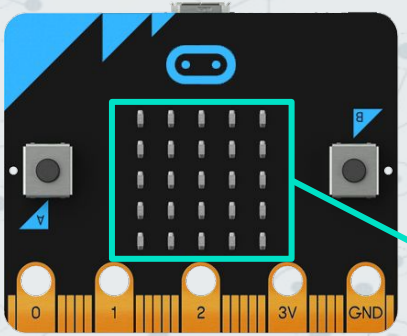
Returns the running total of button presses, and resets this total to zero before returning.

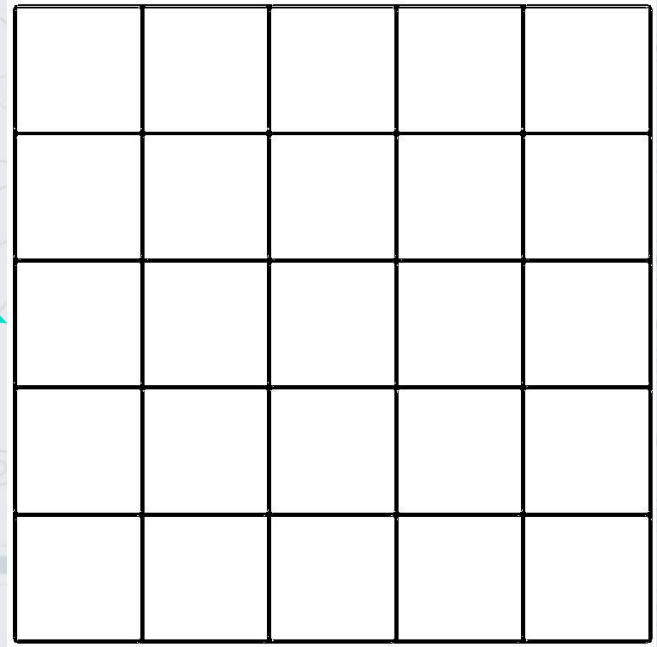


button_a button_b

https://create.withcode.uk/

Drawing (with LEDs)

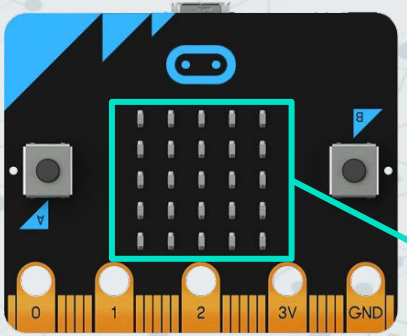


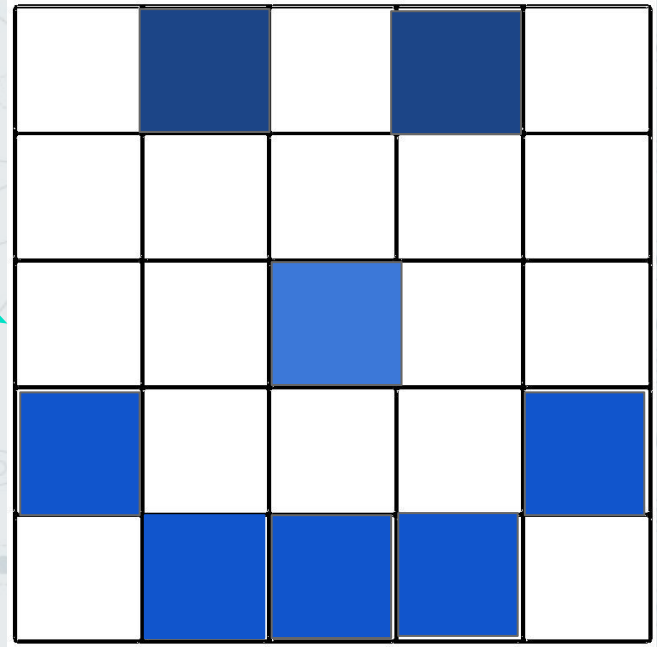


0 1 2 3 4 5 6 7 8 9

Increasing intensity

Drawing (with LEDs)





0 1 2 3 4 5 6 7 8 9

Increasing intensity

Blood Pressure Tester

Alarm Boxes

Lie Detectors

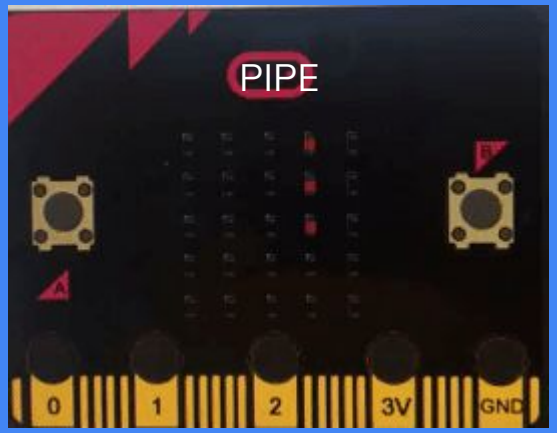
Music Keyboards


Candy Crush
SAGA

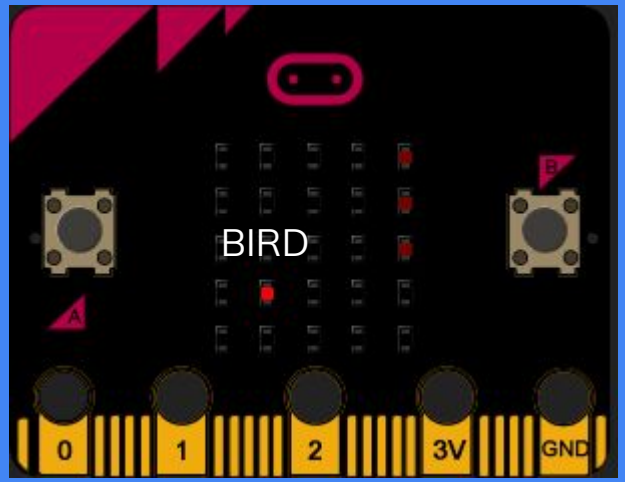
- Display text `display.show()`
- Assign variables
- Display images `Images = Image("00003:00003:00003:00003:00003")`
- Scroll images `display.scroll()`
- Loop
- Detect button input `button_a.was_pressed`

```python
from microbit import *
import random


display.scroll("Get ready...")


# Game constants
DELAY = 20                        # ms between each frame
FRAMES_PER_WALL_SHIFT = 20        # number of frames between each time a wall
moves a pixel to the left
FRAMES_PER_NEW_WALL = 100         # number of frames between each new wall
FRAMES_PER_SCORE = 50             # number of frames between score rising by 1


# Global variables
y = 50
speed = 0
score = 0
frame = 0


# Make an image that represents a pipe to dodge
def make_pipe():
    i = Image("00003:00003:00003:00003:00003")
    gap = random.randint(0,3)   # random wall position
    i.set_pixel(4, gap, 0)      # blast a hole in the pipe
    i.set_pixel(4, gap+1, 0)
    return i

# create first pipe
i = make_pipe()


# Game loop
while True:
    frame += 1

    # show pipe
    display.show(i)

    # flap if button a was pressed
    if button_a.was_pressed():
        speed = -8

    # show score if button b was pressed
    if button_b.was_pressed():
        display.scroll("Score:" + str(score))


    # accelerate down to terminal velocity
    speed += 1
    if speed > 2:
        speed = 2

    # move bird, but not off the edge
    y += speed
    if y > 99:
        y = 99
    if y < 0:
        y = 0

    # draw bird
    led_y = int(y / 20)
    display.set_pixel(1, led_y, 9)

    # check for collision
    if i.get_pixel(1, led_y) != 0:
        display.show(Image.SAD)
        sleep(500)
        display.scroll("Score: " + str(score))
        break

    # move wall left
    if(frame % FRAMES_PER_WALL_SHIFT == 0):
        i = i.shift_left(1)

    # create new wall
    if(frame % FRAMES_PER_NEW_WALL == 0):
        i = make_pipe()

    # increase score
    if(frame % FRAMES_PER_SCORE == 0):
        score += 1

    # wait 20ms
    sleep(20)
```

Creation



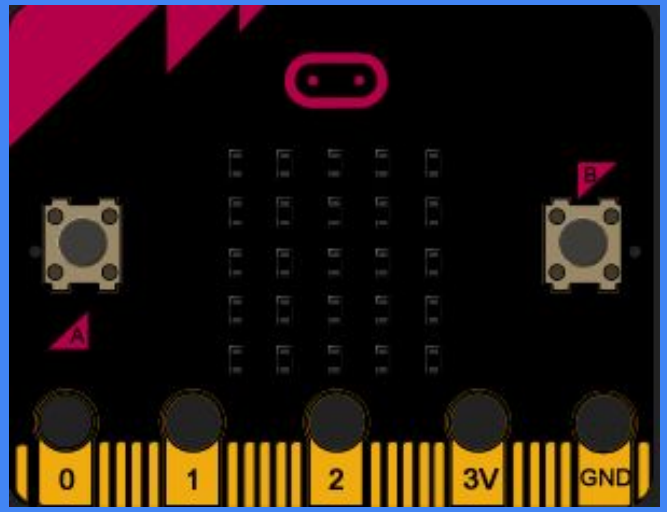

```
# Make an image that represents a pipe to dodge
def make_pipe():
        i = Image("00003:00003:00003:00003:00003")
        gap = random.randint(0,3)      # random wall position
        i.set_pixel(4, gap, 0)         # blast a hole in the pipe
        i.set_pixel(4, gap+1, 0)
        return i
```



Variable y: y-position of bird

```
    led_y = int(y / 20)
    display.set_pixel(1, led_y, 9)
```

Game mechanism



While True:

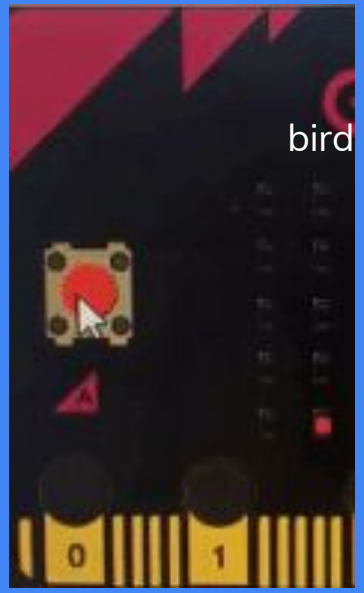
```
frame += 1

# move wall left
if(frame % FRAMES_PER_WALL_SHIFT == 0):
    i = i.shift_left(1)

# create new wall
if(frame % FRAMES_PER_NEW_WALL == 0):
    i = make_pipe()

# increase score
if(frame % FRAMES_PER_SCORE == 0):
    score += 1
```

20

100

50

Gravity

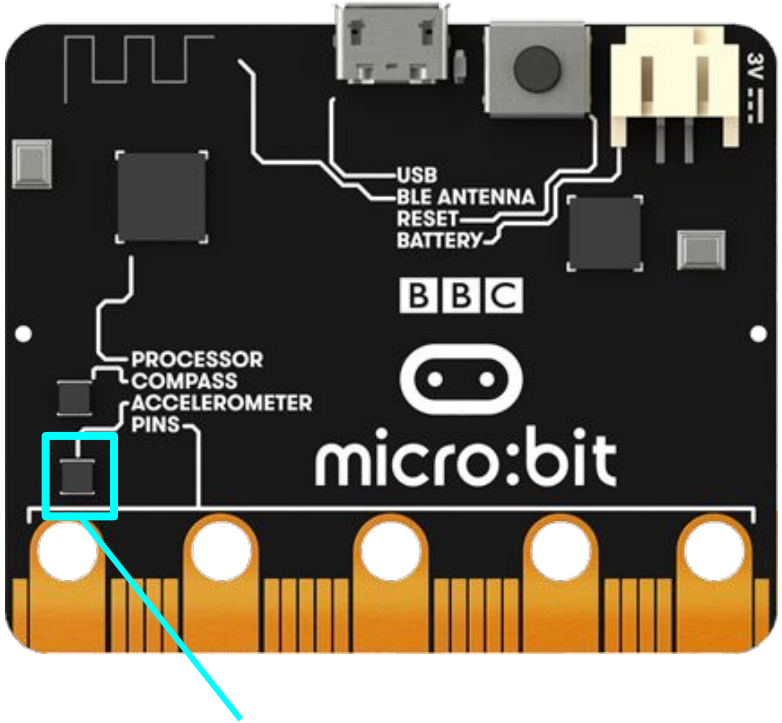



```
# accelerate down to terminal velocity
    speed += 1
    if speed > 2:
        speed = 2

#move bird
    y += speed
```

Small details to note

```
# flap if button a was pressed
if button_a.was_pressed():
    speed = -8
```

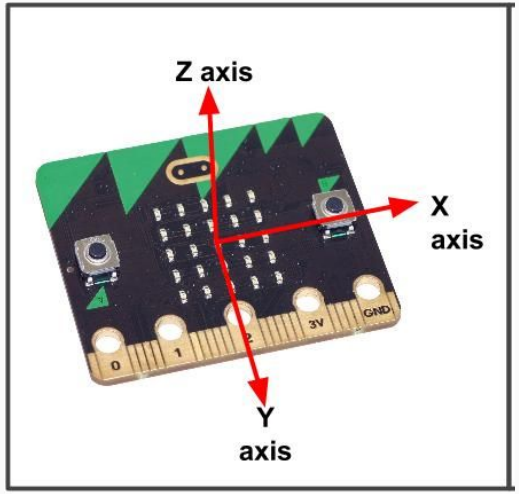
```
sleep(20)
```

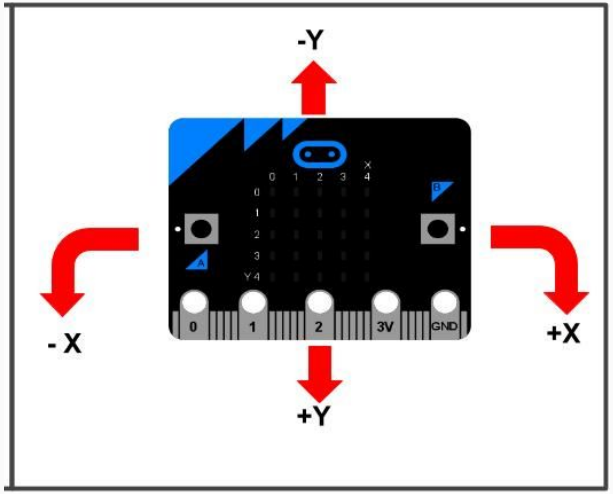


Accelerometer

Measuring Movement

```
x= accelerometer.get_x()
y= accelerometer.get_y()
z= accelerometer.get_z()
```



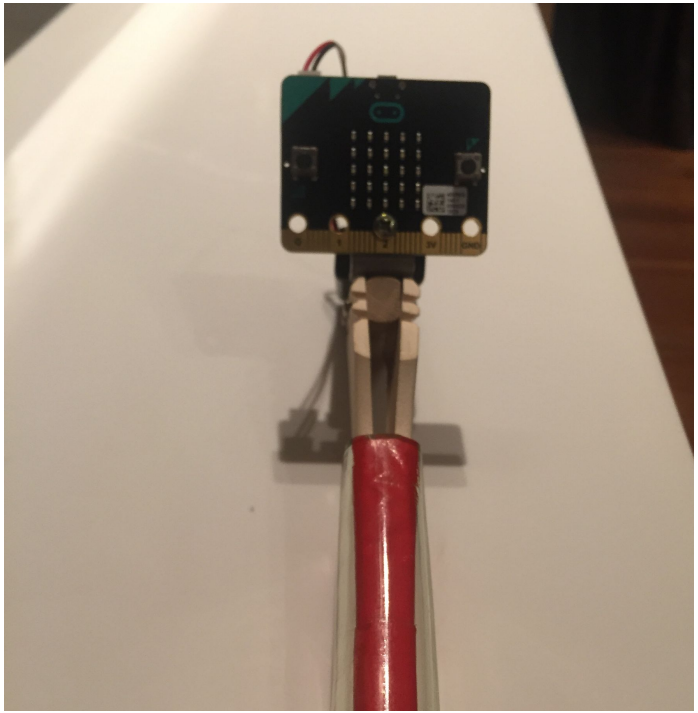


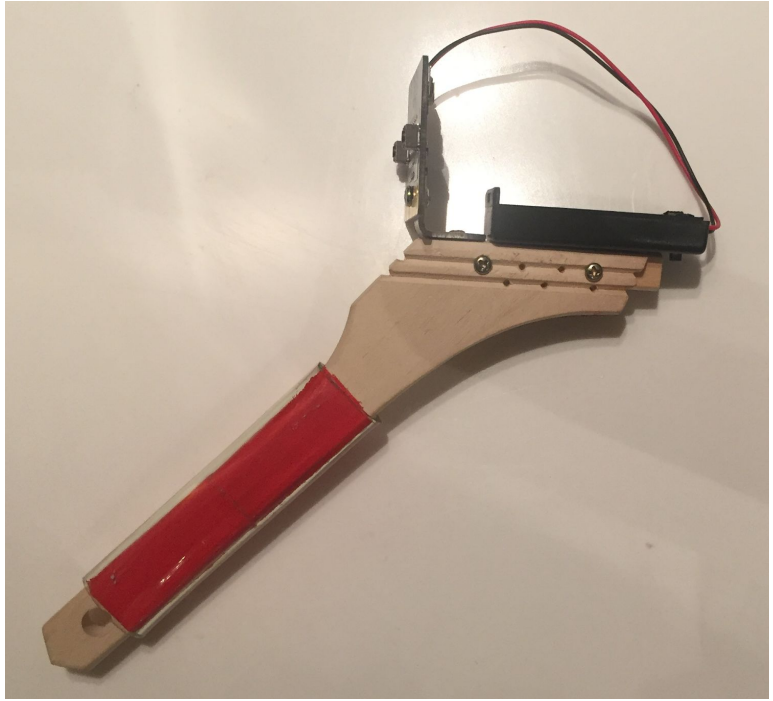
Detecting Gestures

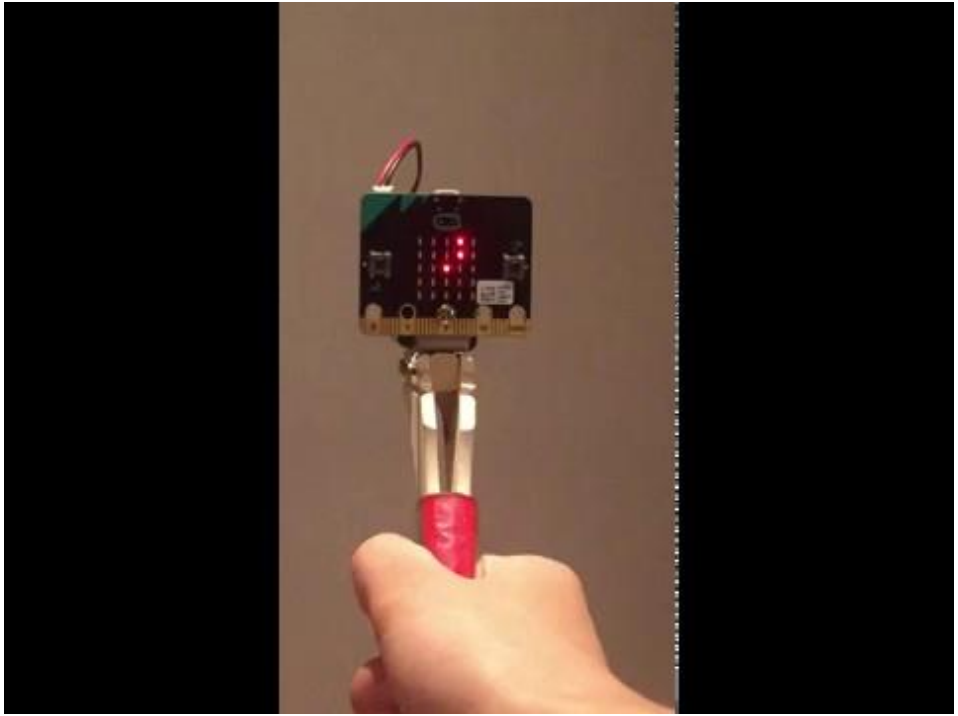
```
1.  gesture= accelerometer.current_gesture()
2.  if gesture == “up”
```

List of Gestures:

- up
- down
- left
- right
- face up
- face down
- Freefall
- shake
- 3g
- 6g
- 8g

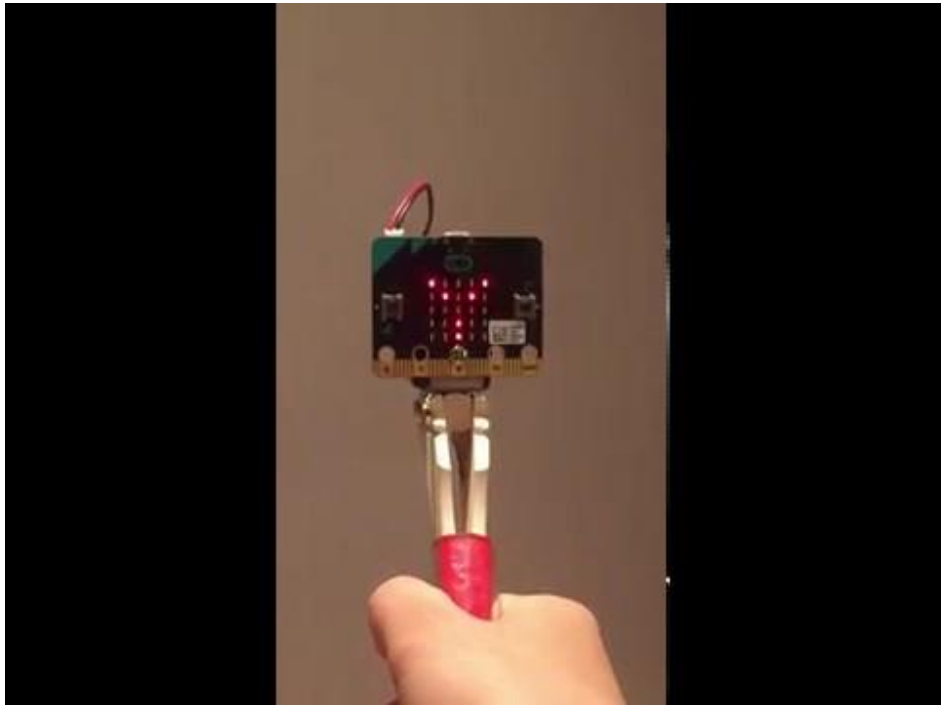






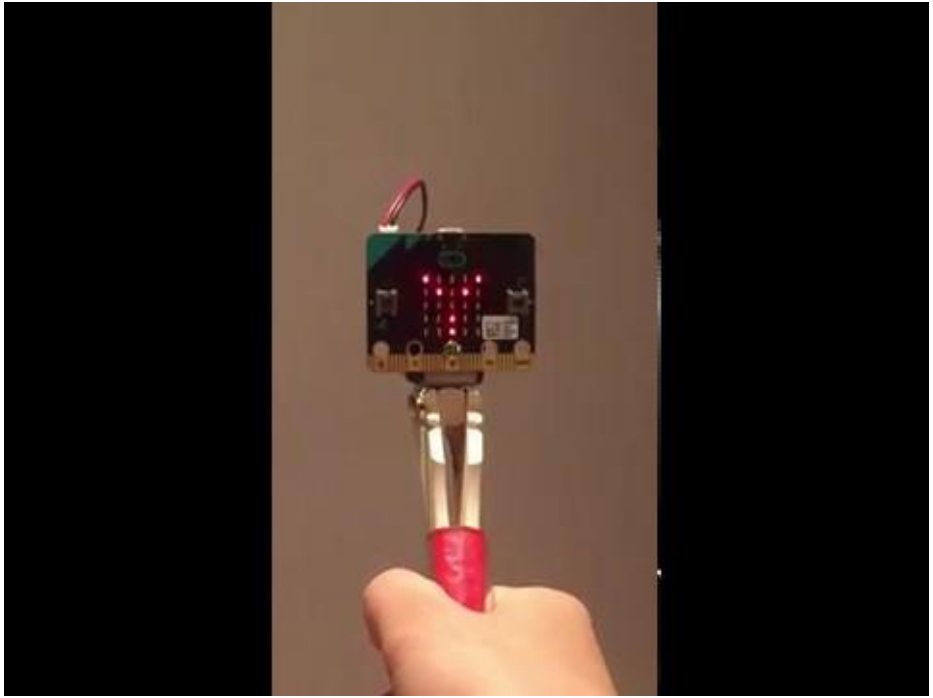
<u>5 seconds</u>

Get into stable position



<u>10 seconds</u>

Deviation from x,y, z reference readings will contribute to total points.



End

The final score is scrolled across the microbit display at the end of it.

Radio Module

```
1.  import radio
```

List of Radio Functions:

- `radio.on`
- `radio.off`
- `radio.reset`
- `radio.send_bytes(message)`
- `radio.receive_bytes()`
- `radio.config(kwarg*)`
- `radio.send(message)`
- `radio.receive(message)`

micro:bit 1

```
from microbit import *
import radio

radio.on()

while True:
    if button_a.was_pressed():
        radio.send('reset')
```

micro:bit 2

```
from microbit import *
import radio

total = 0
scorex = 0
scorey = 0
scorez = 0

radio.on()

while running_time() < 5000:
    display.show(Image.ALL_CLOCKS)
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    z = accelerometer.get_z()
    incoming = radio.receive()
    if incoming == 'reset':
        reset()
```

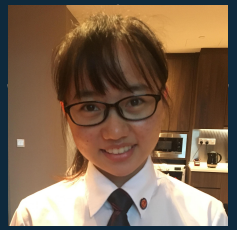
Radio Function

2

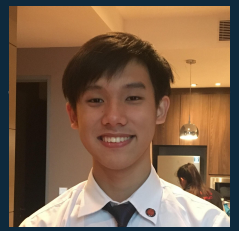
Additional Add-ons to complement the micro:bit

Contacts

Zilong
jin.zilong@dhs.sg



Siyin
wang.siyin@dhs.sg



Zixin
liu.zixin@dhs.sg

Pearlyn
loh.woonqing.pearlyn@dhs.sg

micro:bit resources

http://microbit.org/

https://github.com/bbcmicrobit/micropython

https://github.com/bbcmicrobit/PythonEditor

Getting your own micro:bits...

http://microbit.org/resellers/

(approximately USD$17 per micro:bit)

www.tinyurl.com/
pyconmicrobit