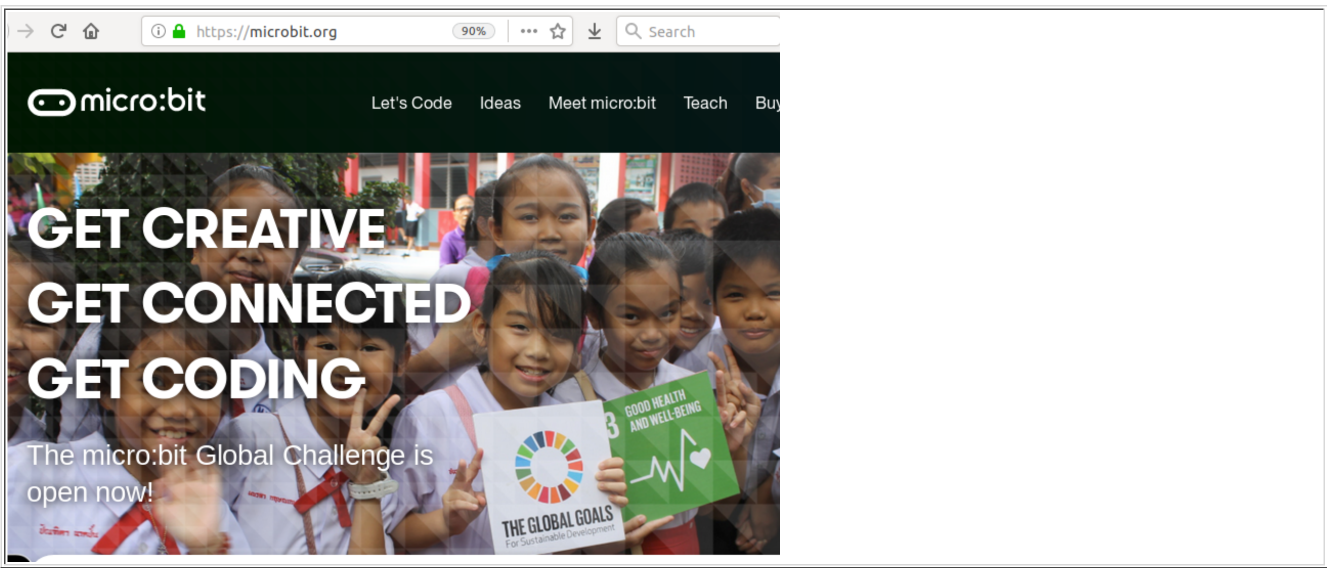
BBC micro:bit:
What is it good for?

Richard Mann, Associate Professor, Computer Science
http://www.cs.uwaterloo.ca/~mannr

Monday 5 November 2018
MC4063, 5:30pm
University of Waterloo



Introduced 2015, aimed at K-12 education, non specialist hobby/makers, ...

This is a great platform for CS teaching and experimentation

It makes realtime programming and hardware design look easy.

Realtime is not easy (!), but this is a great way to get started down this road. Experimentation, design and measurement.

"Internet of things" (IoT) == Programming in the small

Continuous running (very low power, source of power or charging required)
Small programs run in SRAM with Flash storage for programs and/or logging
Real time operation

Embedded in products or personal devices
May or may not have wireless, or blue tooth (low energy)
Devices should be battery operated, very low power, and only "wake" when needed transmit and/or process data.

My interest:
- body worn assistive devices (hearing aids, health monitors)
- devices in home or workplace

Caution warranted, both in security (tracking) and in safety of devices (wireless exposure)

More complicated devices would involve realtime control and more computation,
eg., drone control, navigation, automatic driving, etc.

QUERY: Has anyone done programming of this type?

Outline of Talk

1 BBC micro:bit,
  Programming Examples
  MakeCode/ Javascript/ Python

2 Gateway to electronics... Kitronik INVENTORS KIT
  Introduction to electronics
  Expt 2, using electronics inputs
  Extend1, allow bar meter
  Extend2, drive output LED

3 Analysis of device capabilities
  Analog I/O (rate, accuracy # of bits per sample)
  Real time operation (throughput, latency)

4. Micropython. "CircuitPython", running on Adafruit Feather M4

Part 1: BBC micro:bit

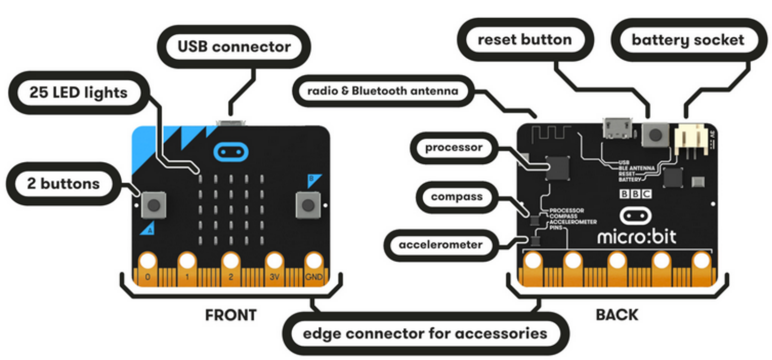
Hardware:
- Cortex M0 processor (ARM CPU)
- Simple embedded operating system/runtime
- USB interface (can be battery powered as well)

Programming:
- Connect to USB
- Web based editor
    (Microsoft "make code" blocks, javascript, python)
- Drag and drop HEX file to device
--> Runs program standalone (resident SRAM),
     can communicate with PC via Serial port over USB

Peripherals, built in to device
- Two buttons A, B
- 7x7 segment LED (static, flashing or scrolling characters)
- Compass
- Accelerometer
- Infra red (IR) Transmit and Receive (can emulate TV remote)
- Bluetooth Low Energy wireless with antenna

- Digital I/O
- Analog I/O
    Virtually any external instrument can be sensed/measured using inputs and controlled using outputs

https://microbit.org/guide/features/



Specs:
https://os.mbed.com/platforms/Microbit/ (hardware specific)

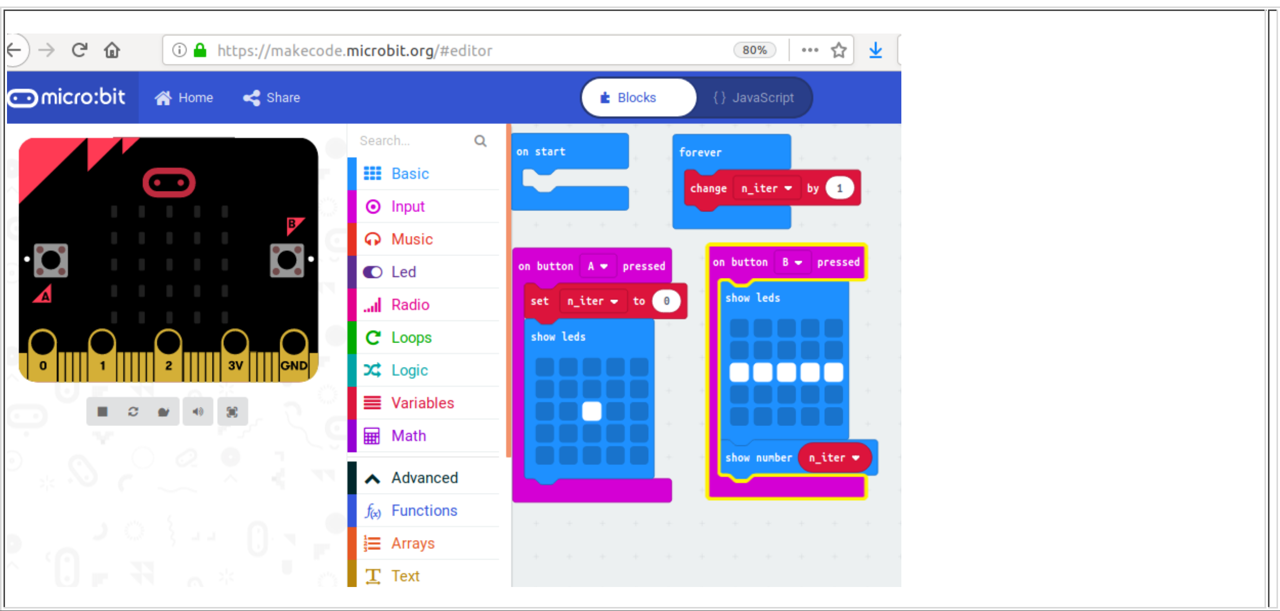
https://lancaster-university.github.io/microbit-docs/ (C api)

- Nordic nRF51822 Multi-protocol Bluetooth® 4.0 low energy/2.4GHz RF SoC
  - 32-bit ARM Cortex M0 processor (16MHz)
  - 16kB RAM
  - 256kB Flash
  - Bluetooth Low Energy Master/Slave capable
- Input/Output
  - 25 LED Matrix
  - Freescale MMA8652 3-axis Accelerometer
  - Freescale MAG3110 3-axis Magnetometer (e-compass)
  - Push Button x2
  - USB and Edge connector Serial I/O
  - 2/3 reconfigurable PWM outputs
  - 5 x Banana/Croc-clip connectors
  - Edge connector
  - 6 x Analog In
  - 6-17 GPIO (configuration dependent)
  - SPI
  - i2c
- USB Micro B connector
- JST power connector (3v)

Web programming
https://makecode.microbit.org/#editor
- java or blocks mode, show
- simulator, fast and slow mode
- data "explorer" even shows ASM code
  (but some runtime is not shown, in particular, Radio)

*** Caution: You are downloading unknown HEX code ***

Example #1: Buttons



Simulator at left

Blocks at right

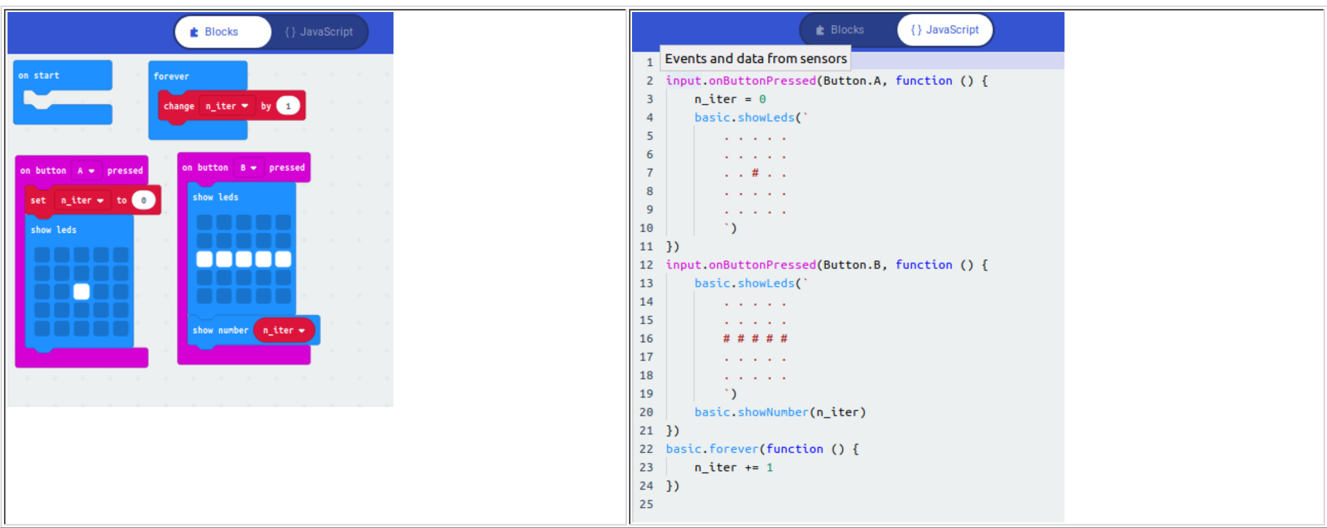
Hardware questions:
- what displays on startup?
- what A and B are hit at same time?
- A and B are electronic switches.
  What happens if I hit close-open very quickly.
  Could they even be missed?

Timer counts cycle time from A press to B press

Those are all the problems of "real time programming". (Trains Course ...)

Blocks vs Javascript

Note: some things, like variables, have to be entered in javascript
(blocks forms won't allow show of variable name...)

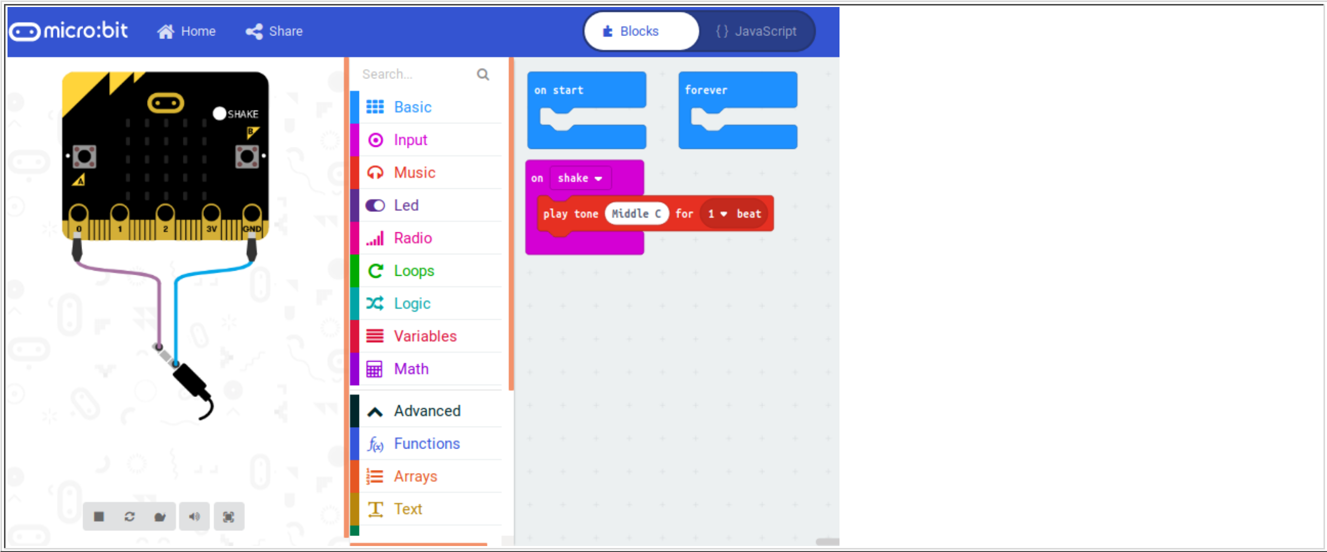



```
input.onButtonPressed(Button.A, function () {
    n_iter = 0
    basic.showLeds(`
        . . . . .
        . . . . .
        . . # . .
        . . . . .
        . . . . .
        `)
})
input.onButtonPressed(Button.B, function () {
    basic.showLeds(`
        . . . . .
        . . . . .
        # # # # #
        . . . . .
        . . . . .
        `)
    basic.showNumber(n_iter)
})
basic.forever(function () {
    n_iter += 1
})
```

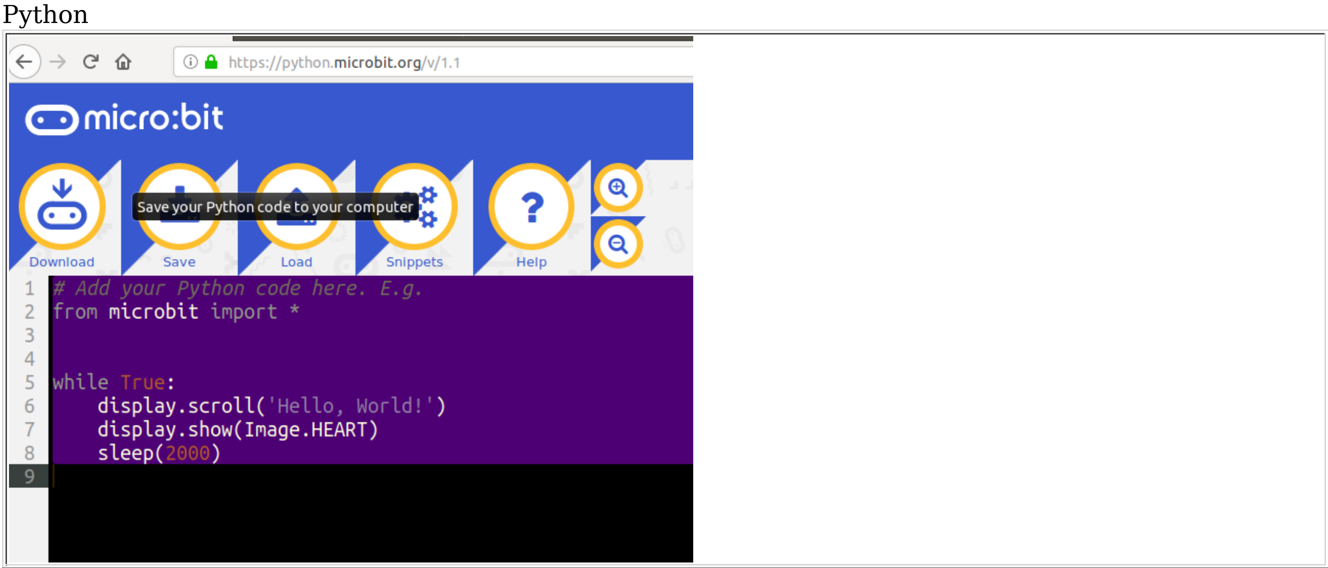
<u>Example #2 - Shake.</u>

Gesture input (accelerometer),
Sound output (Analog out #0)

<u>Notes</u>:
- seems to queue events.
- sound wave is not a nice tone. It is a rough "square wave".



Python



Part 2: Electronics

https://www.kitronik.co.uk/

Excl. VAT: £15.98
Incl. VAT: **£19.176**
As low as: £17.40

:KLEF Piano for the BBC micro:bit

Excl. VAT: £10.75
Incl. VAT: **£12.90**

Noise Pack for Kitronik Inventor's Kit for the BBC micro:bit

Excl. VAT: £16.66
Incl. VAT: **£19.992**
As low as: £16.80

All-in-one Robotics Board for BBC micro:bit

Kitronik Experimenter kit

Electronics 101

<handwritten notes>

Experiment
Input: LDR
Ouput: LED

Part 3. Measuring performance

Realtime programming

** measured behaviour, no guarantee worst case performance!

- latency: time to respond to input
- throughput: # of inputs processed/second
- typically program is by polling or interrupt driven
    polling: check for status change on regular interval (eg., 1/1000 second)
    interrupt: wait for events

- real systems, such as audio processors, networks, etc, work on chunks of data.
  further the data is

	Polling	Interrupt driven
Latency	fixed (eg., 1/1000 sec)	delay (seconds, or # samples)
Throughput	samples/interval	samples/second

Simple experiment to measure performance

Part 4. Micropython

MicroPython

From Wikipedia, the free encyclopedia

MicroPython[3] is a software implementation of the Python 3 programming language, written in C, that is optimized to run on a microcontroller.[4][5] MicroPython is a full Python compiler and runtime that runs on the micro-controller hardware. The user is presented with an interactive prompt (the REPL) to execute supported commands immediately. Included are a selection of core Python libraries; MicroPython includes modules which give the programmer access to low-level hardware.[3]

MicroPython was originally created by the Australian programmer and physicist Damien George, after a successful Kickstarter backed campaign in 2013.[6] While the original Kickstart campaign released MicroPython with a pyboard microcontroller, MicroPython supports a number of

MicroPython

Original author(s)	Damien George
Initial release	May 3, 2014; 4 years ago
Stable release	1.9.4 / May 11, 2018; 5 months ago

The interpreter runs directly on hardware.

Demo: Adafruit Feather M4. Cortex M4 @ 120 MHz, single precision FPU
https://circuitpython.readthedocs.io/en/3.x/docs/index.html

Environment: Mu-editor

Example: Reading BMP 280 pressure sensor

The Future: Audio Programming?

Audio programming requires:

high sampling rates (44.1kHz, 48.0 kHz or more)
high resolution ADC (16 bits per sample, up to 24 bits per sample)

real time streaming of input and output (samples stored in circular buffer)
Low latency
(eg., 1ms response --> 44100 * 1/1000 ~= 44 samples)

floating point performance (DSP = digital filtering)
real time Fourier transform (256 --> 1024 points processed at one time)