
Microsoft
Microsoft Research
Faculty
Summit
2016

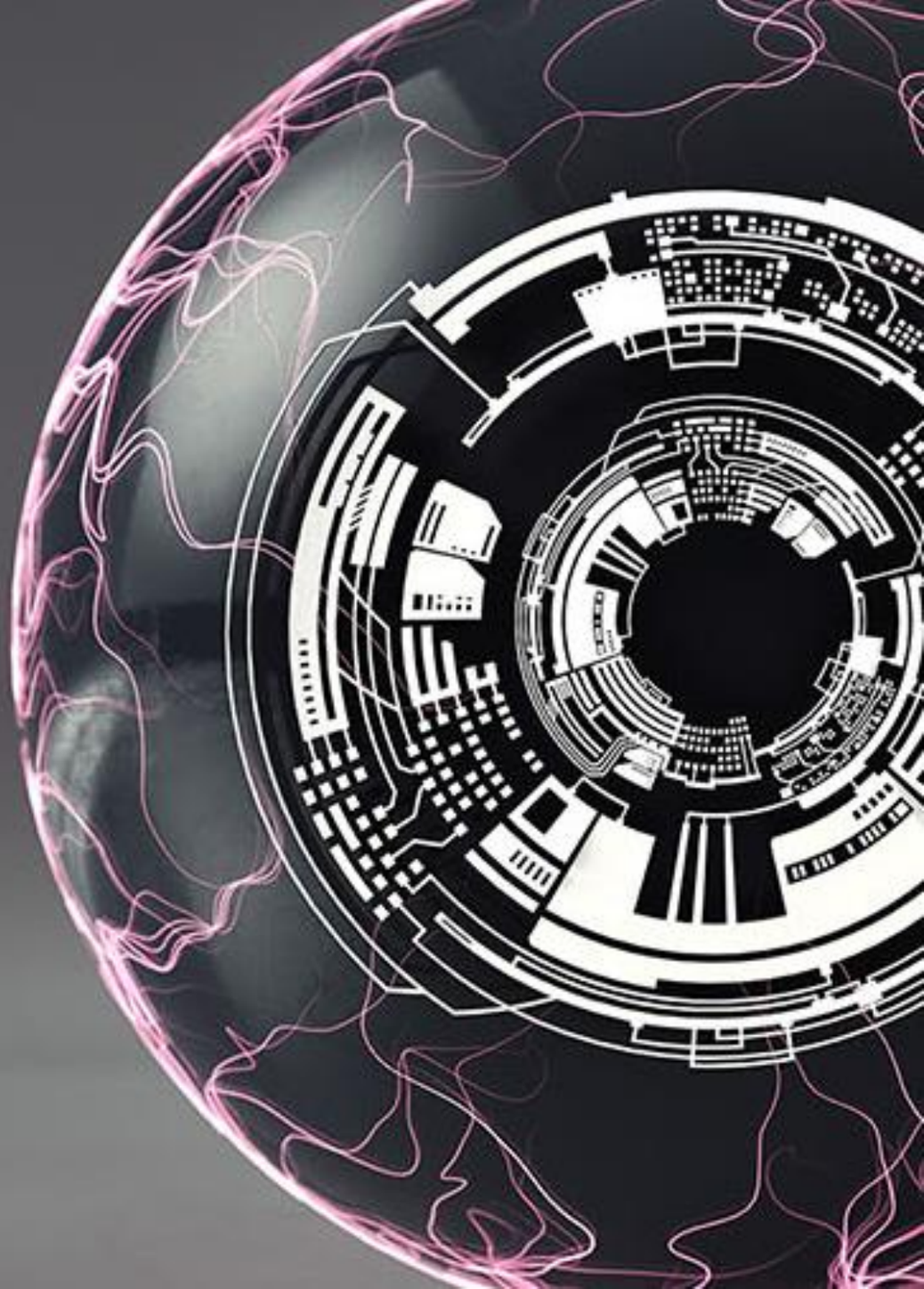
This session – The BBC Microbit

- Thomas Ball — *The micro:bit – overview*
  - MSR
- Joseph Finney — *The micro:bit runtime – inside and out*
  - Lancaster Univ.
- Benjamin Shapiro — *The micro:bit and CS education*
  - Univ. of Colorado

Microsoft

The micro:bit – Overview

Tom Ball
Microsoft Research




commodore 64


apple II


BRITISH BROADCASTING CORPORATION
MICROCOMPUTER SYSTEM


sinclair
ZX81

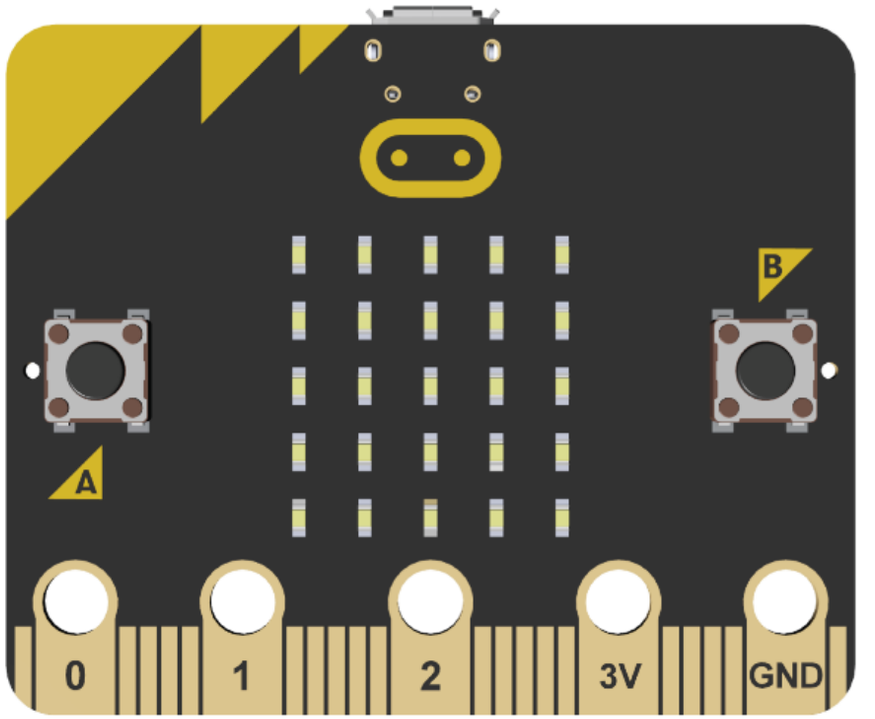

sinclair
ZX Spectrum

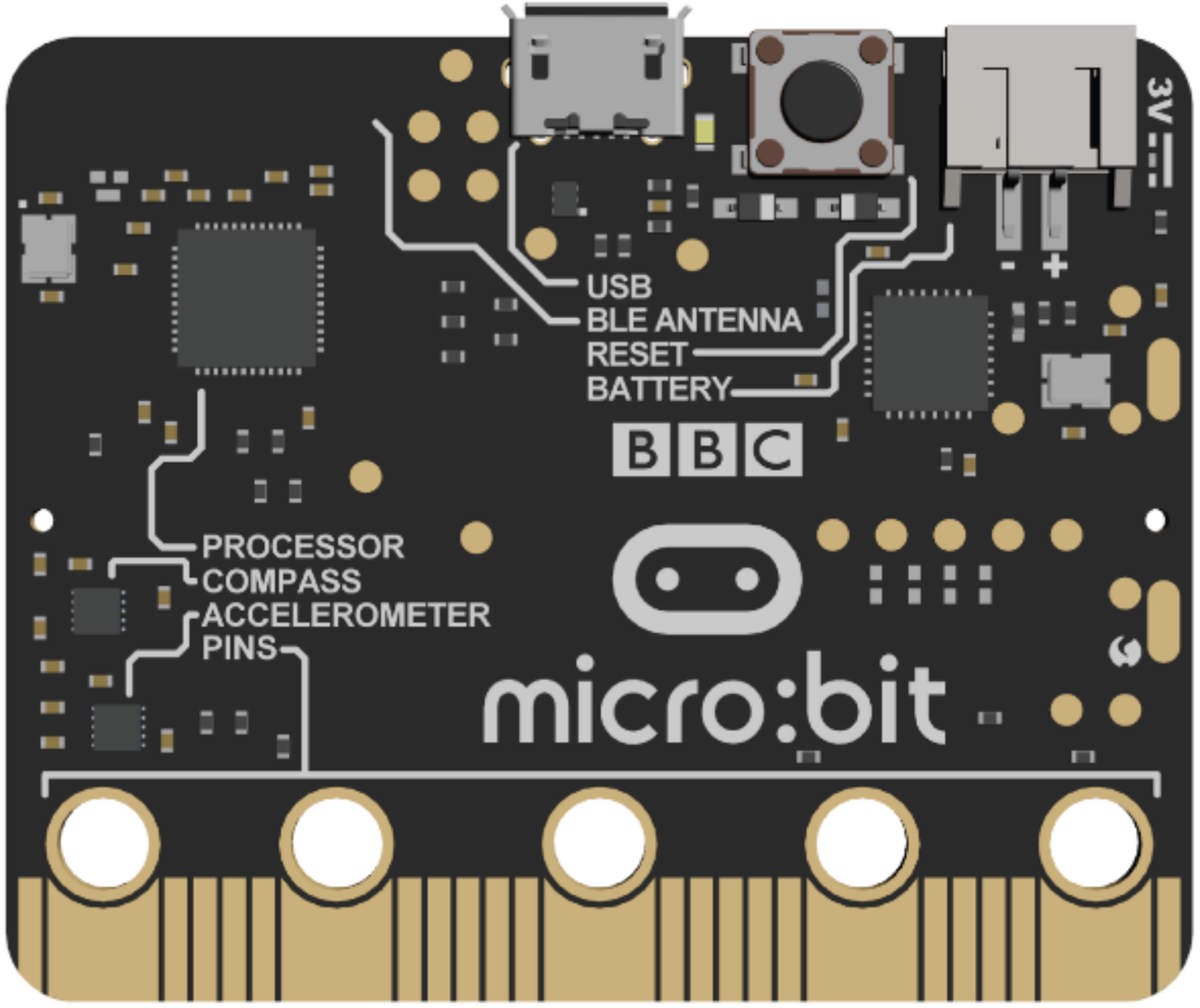

BBC
micro:bit

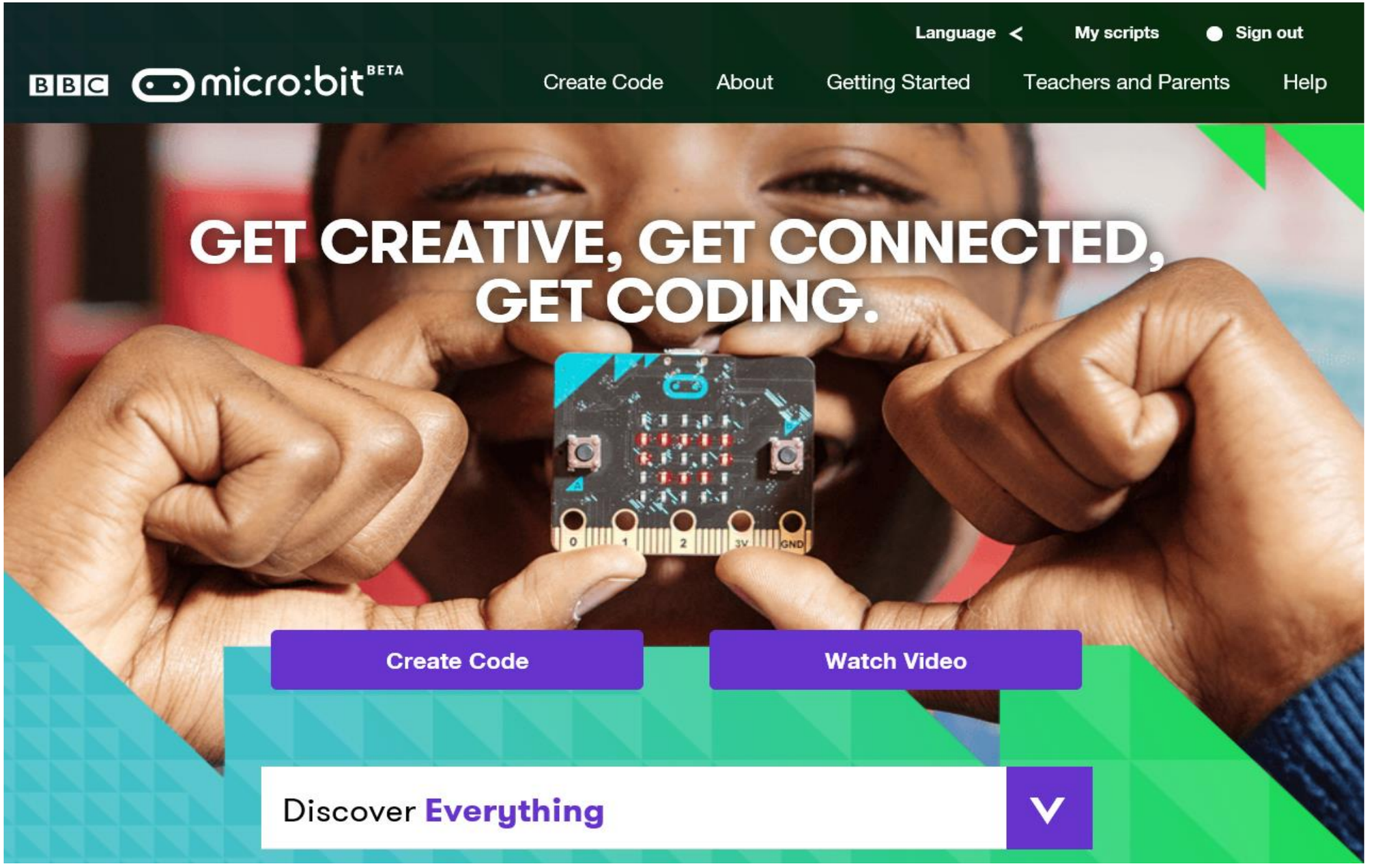
BBC micro:bit

- Friendly hardware
- Friendly software
- Learning/training materials

- 1,000,000 devices seeded into UK this school year



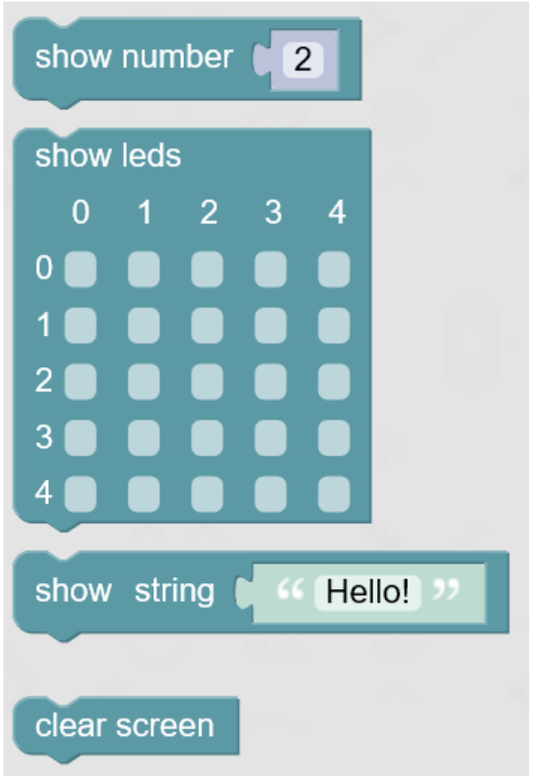

3V
USB
BLE ANTENNA
RESET
BATTERY
BBC
PROCESSOR
COMPASS
ACCELEROMETER
PINS
micro:bit



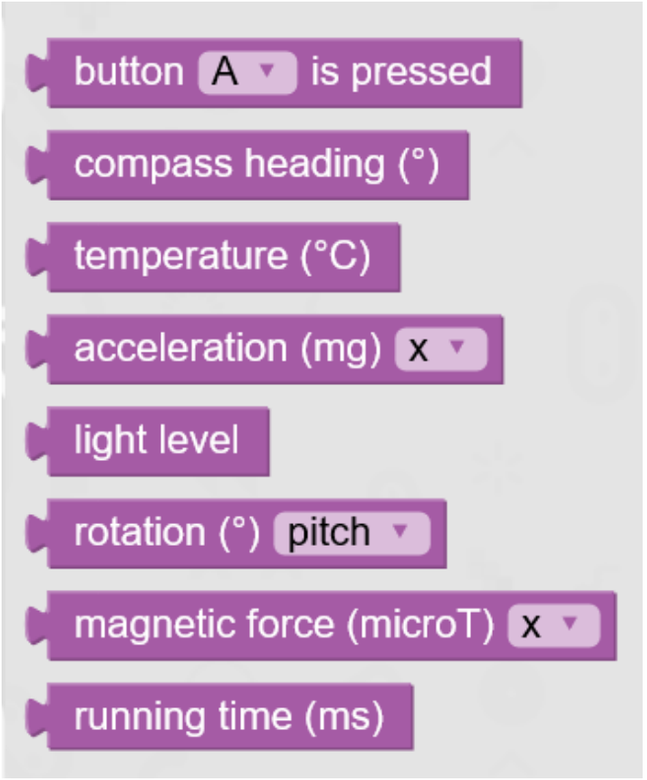
www.microbit.co.uk

Feature Rich

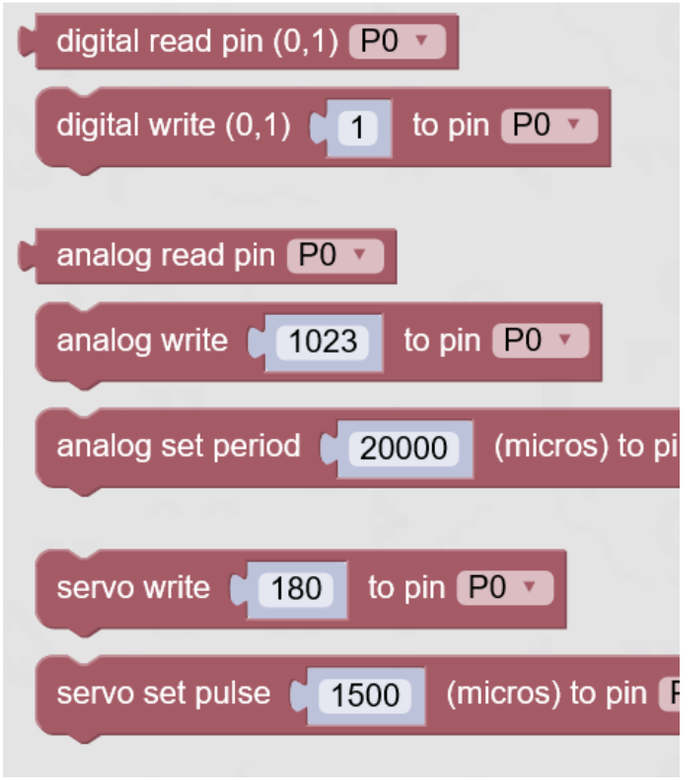
Display



Sensors

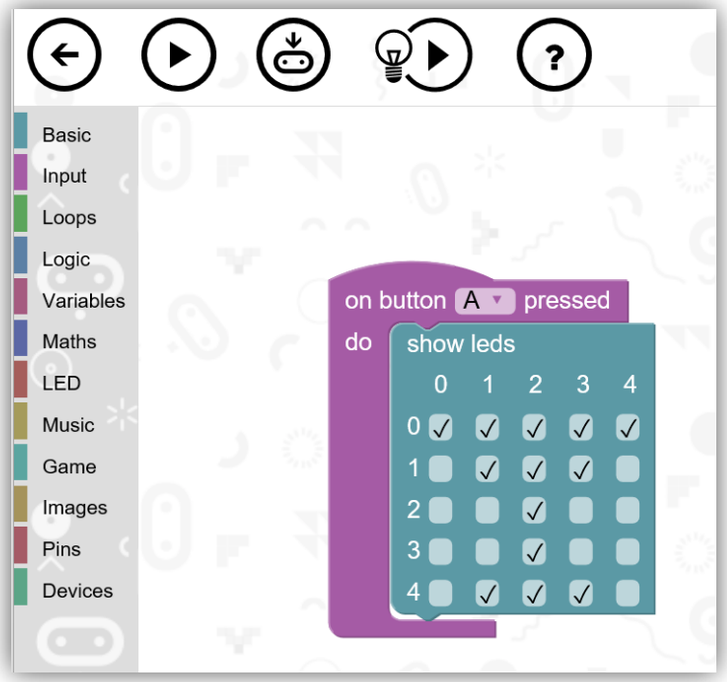


I/O Pins



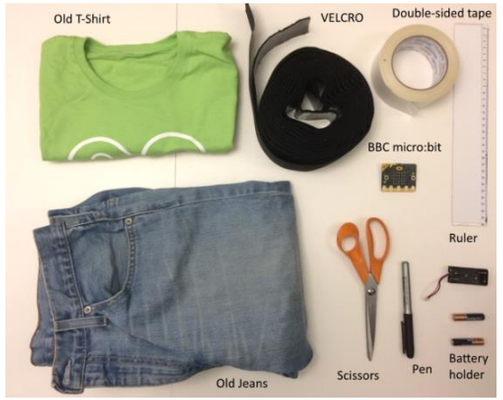
Radio

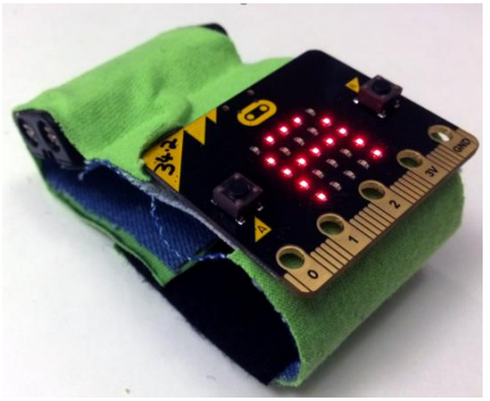
Simple to Code, Anywhere



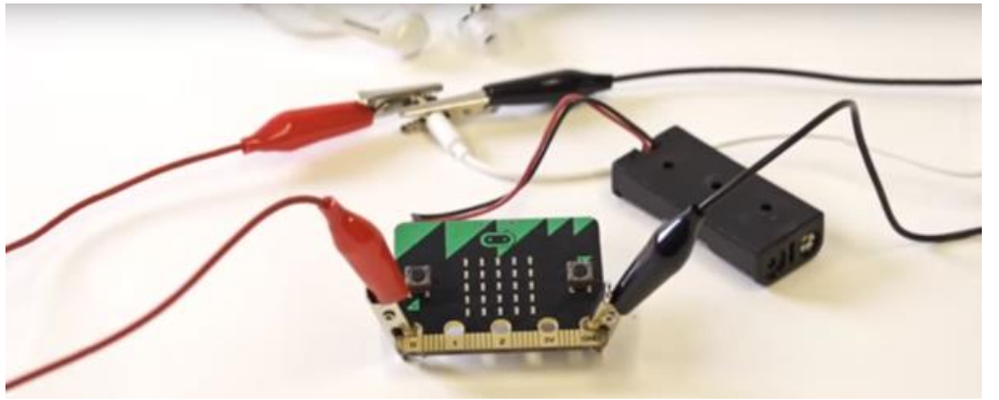
Lessons

Make your own watch

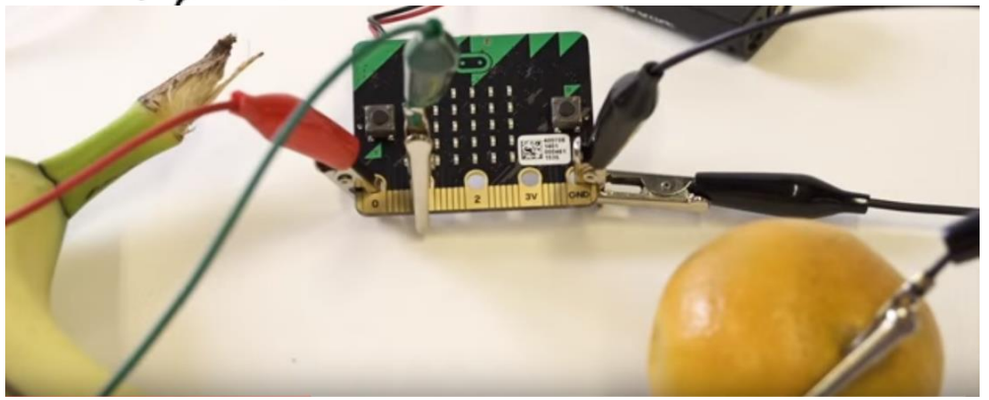




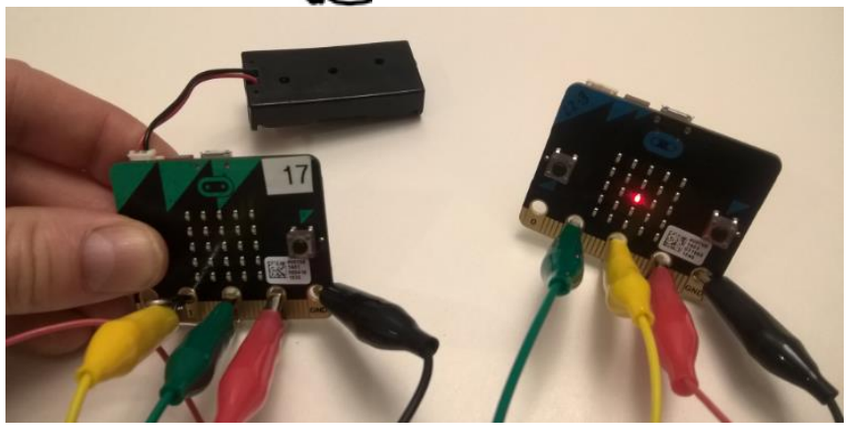
Hack your headphones



Banana keyboard



Telegraph



Core Technology Partnership

Microsoft

- edit/run/simulate/compile
- web site, services, lessons
- hdw design

PL

Lancaster University

- C++ micro:bit runtime

OS

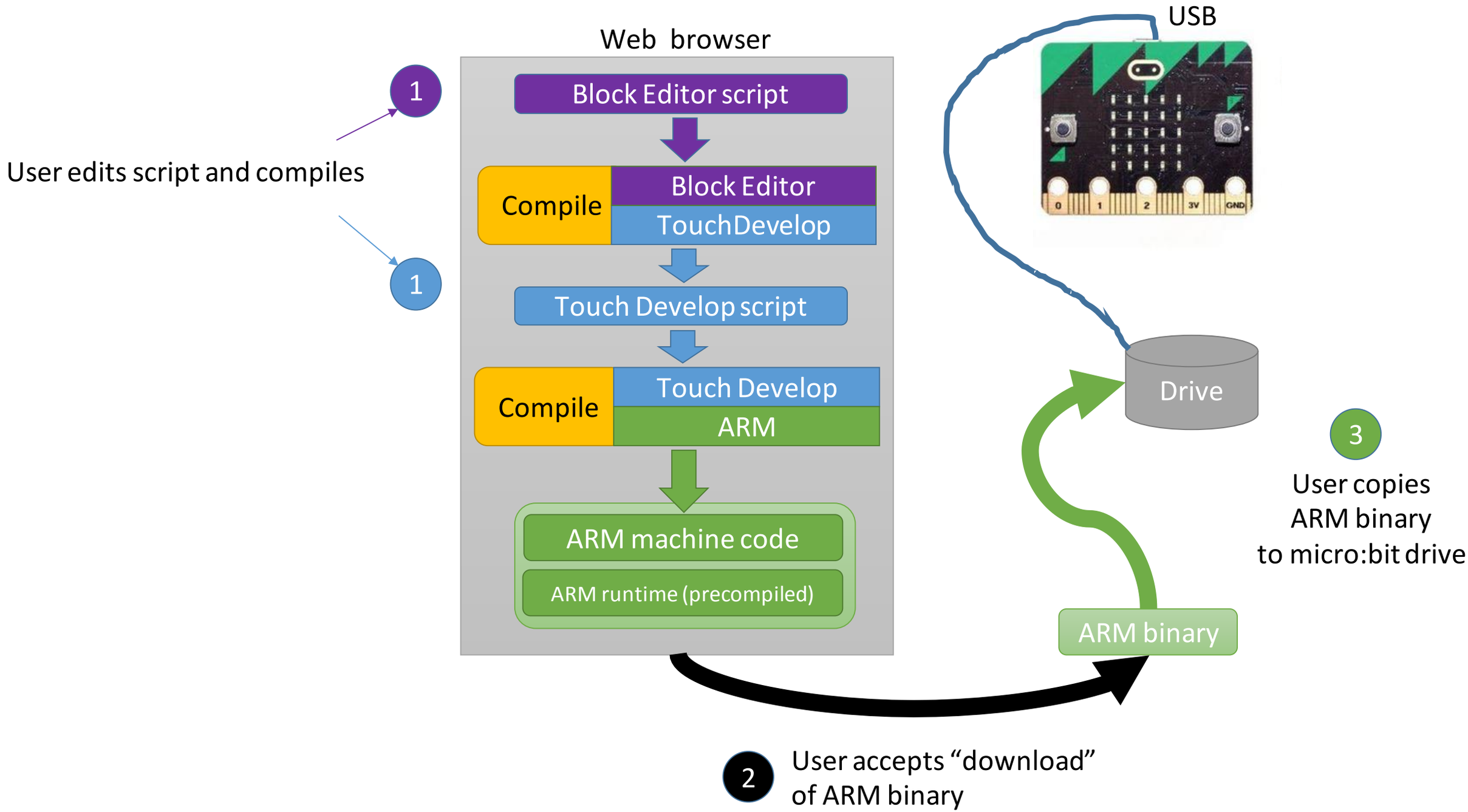
ARM mbed

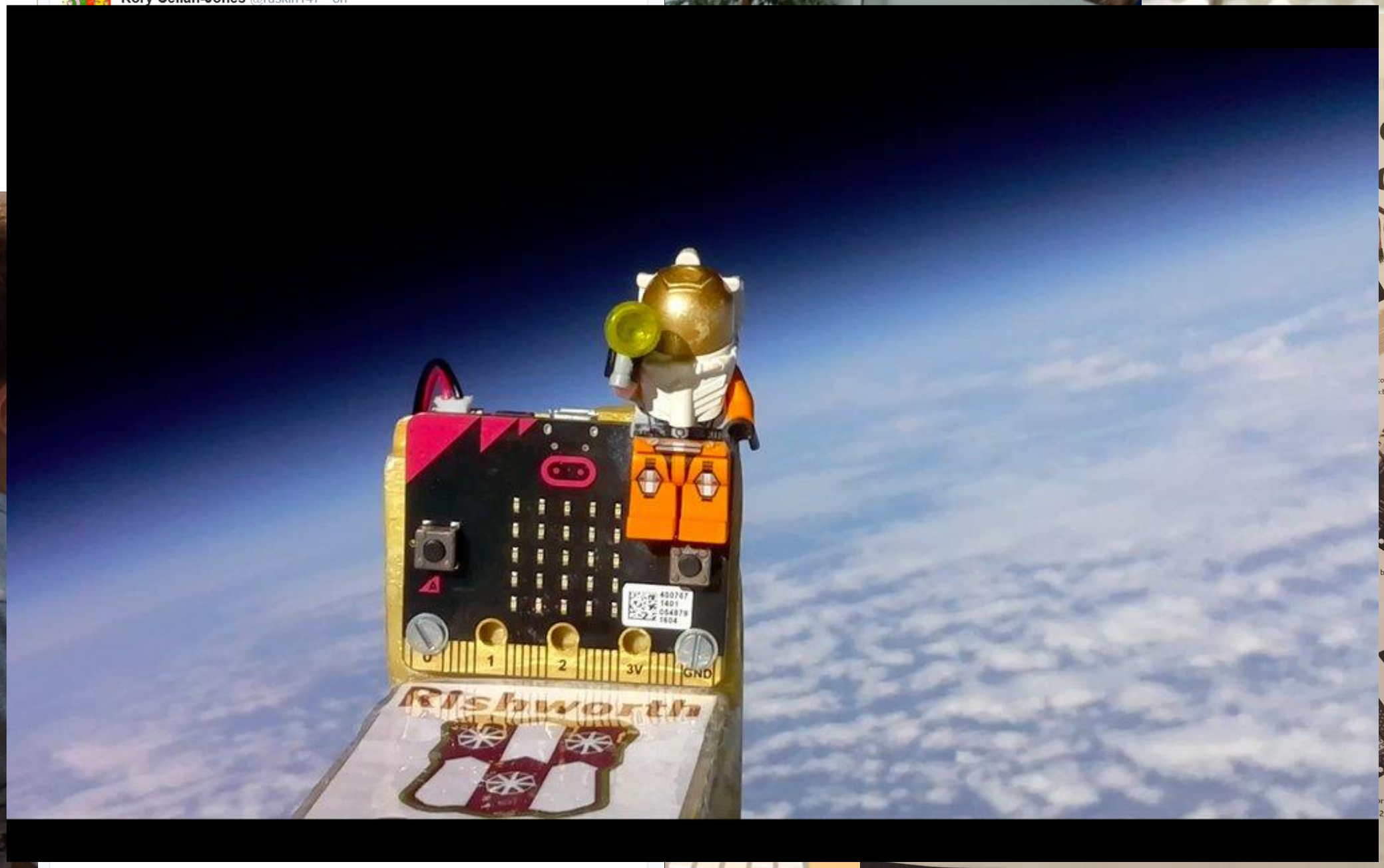
- design, mbed platform, SDK

HDW

Farnell element14

- design, manufacture, package


Web browser
1
User edits script and compiles
1
Block Editor script
Compile
Block Editor
TouchDevelop
Touch Develop script
Compile
Touch Develop
ARM
ARM machine code
ARM runtime (precompiled)
USB
Drive
3
User copies
ARM binary
to micro:bit drive
ARM binary
2
User accepts “download”
of ARM binary


0
1
2
3V
GND


Kitronik
CONTAINS OVER 50 PARTS PLUS A TUTORIAL BOOK!
INVENTOR'S KIT
FOR BBC micro:bit
THE SIMPLE WAY TO LEARN ABOUT CREATING CIRCUITS AND CODE
CONTAINS 50+ PARTS!
Kitronik
INVENTOR'S KIT
FOR BBC micro:bit


micro:bit
Panasonic
Pro Power
Panasonic
Pro Power


iPHONE 6S PLUS vs ANDROID'S BIG GUNS
Stuff
WHAT'S NEXT
TECH
YOU CAN
MAKE
SONY XPERIA X
2016's hottest phone yet
iPHONE 7
You're going to need new headphones
Free to 1 million people!
No soldering required!
35
Techtastic projects you can do NOW!
HADOUKEN!
Street Fighter V kicks off
GRAB YOUR BBC MICRO:BIT AND JOIN THE DIY TECH REVOLUTION

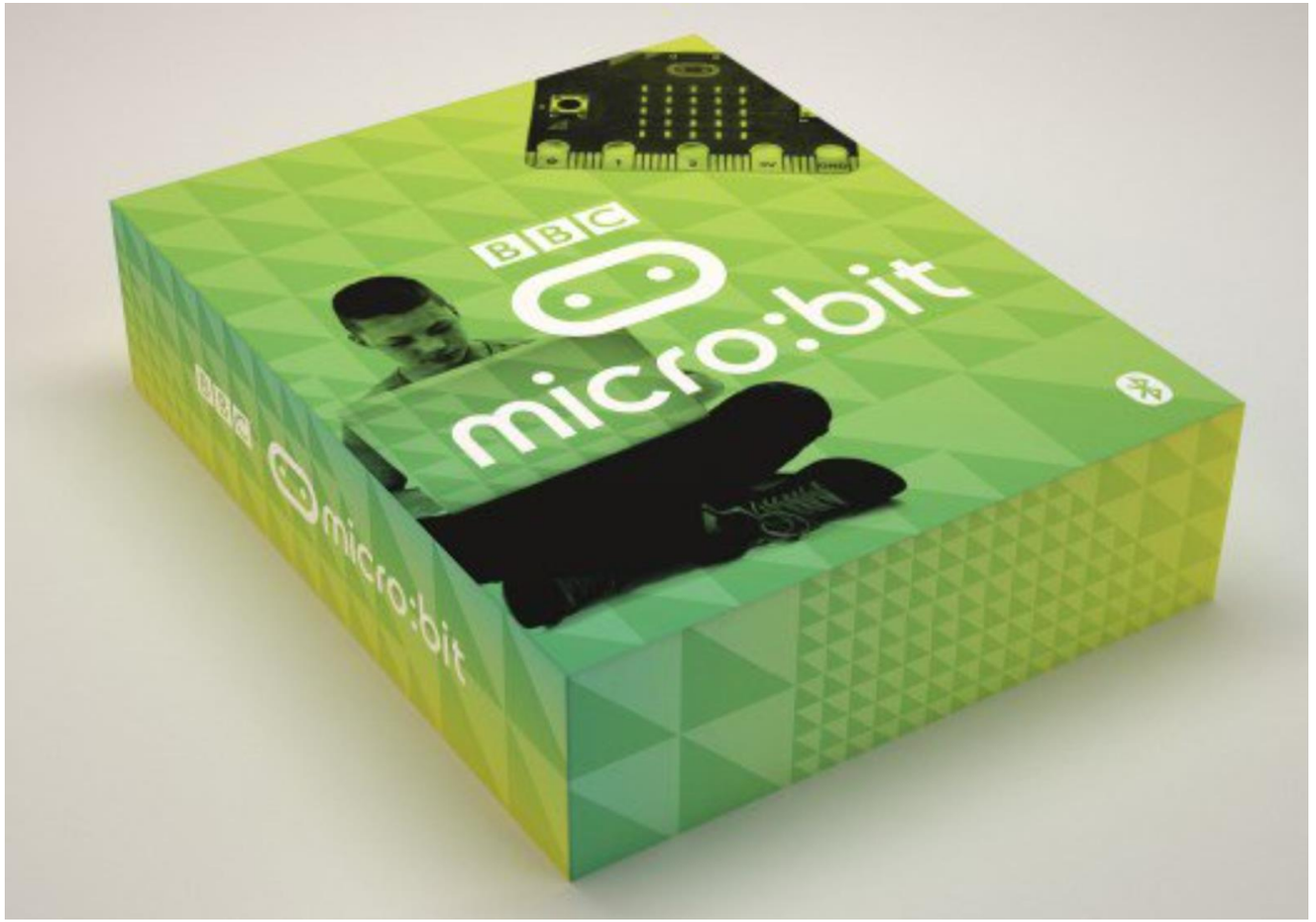
Home / The

170
Members

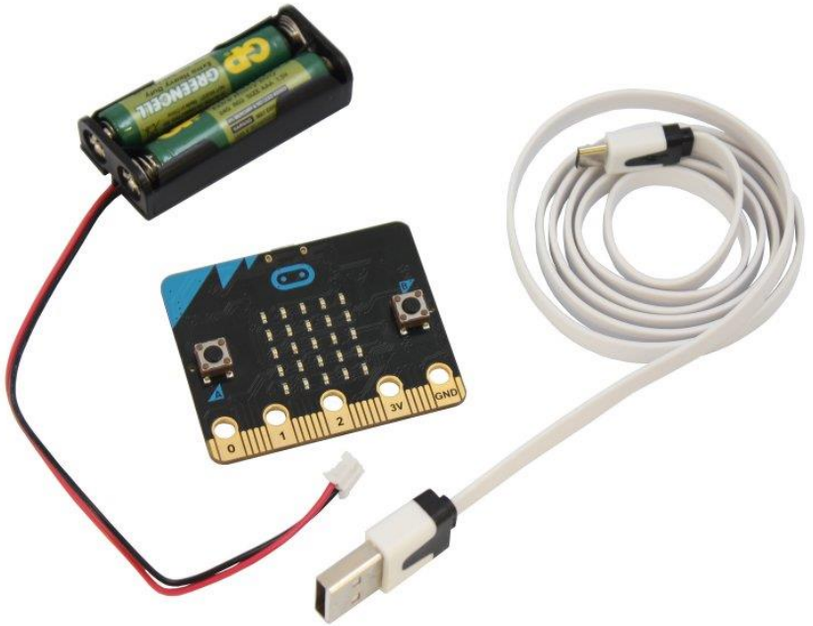
ngland

What's Next?

Commercial availability: UK/EU first…



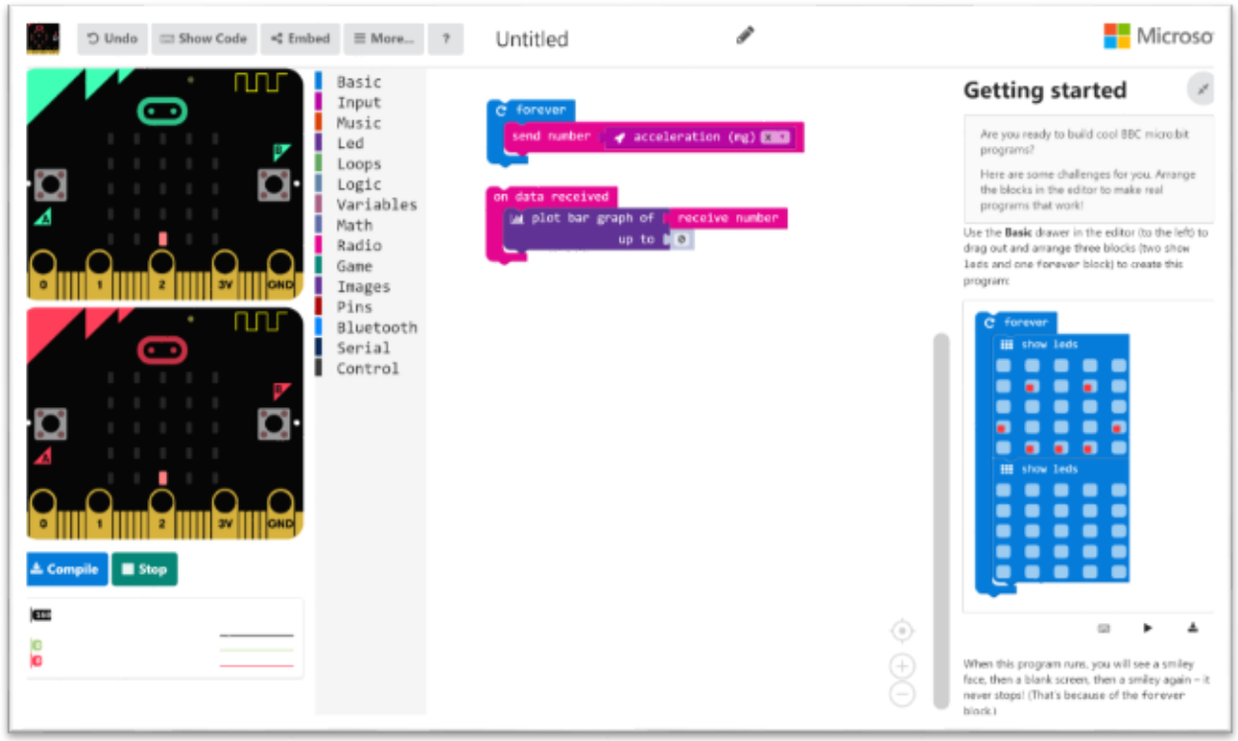
http://uk.farnell.com/bbc-microbit



www.codethemicrobit.com

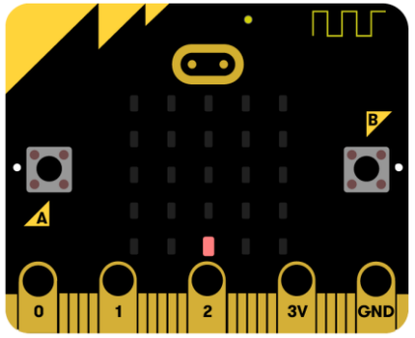
- New Microsoft web site
- Blocks <-> JavaScript
- Streaming data to Azure

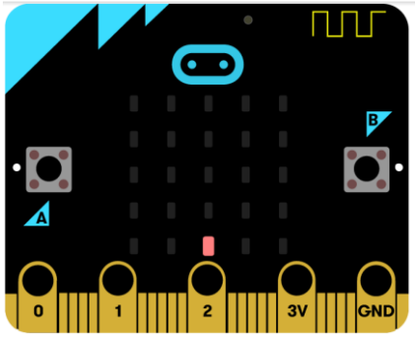
- Architected for extensibility

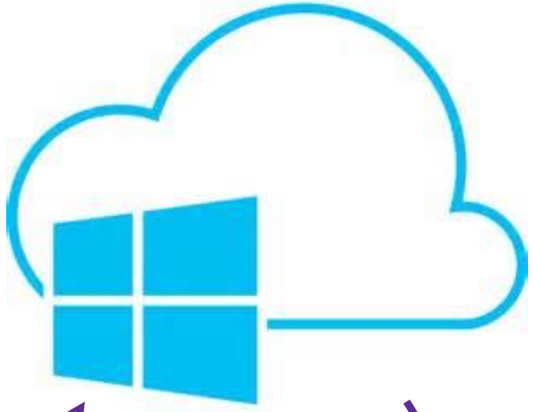


https://codethemicrobit.com/tfllepvcrh

Bluetooth -> Gateway -> Azure -> Excel

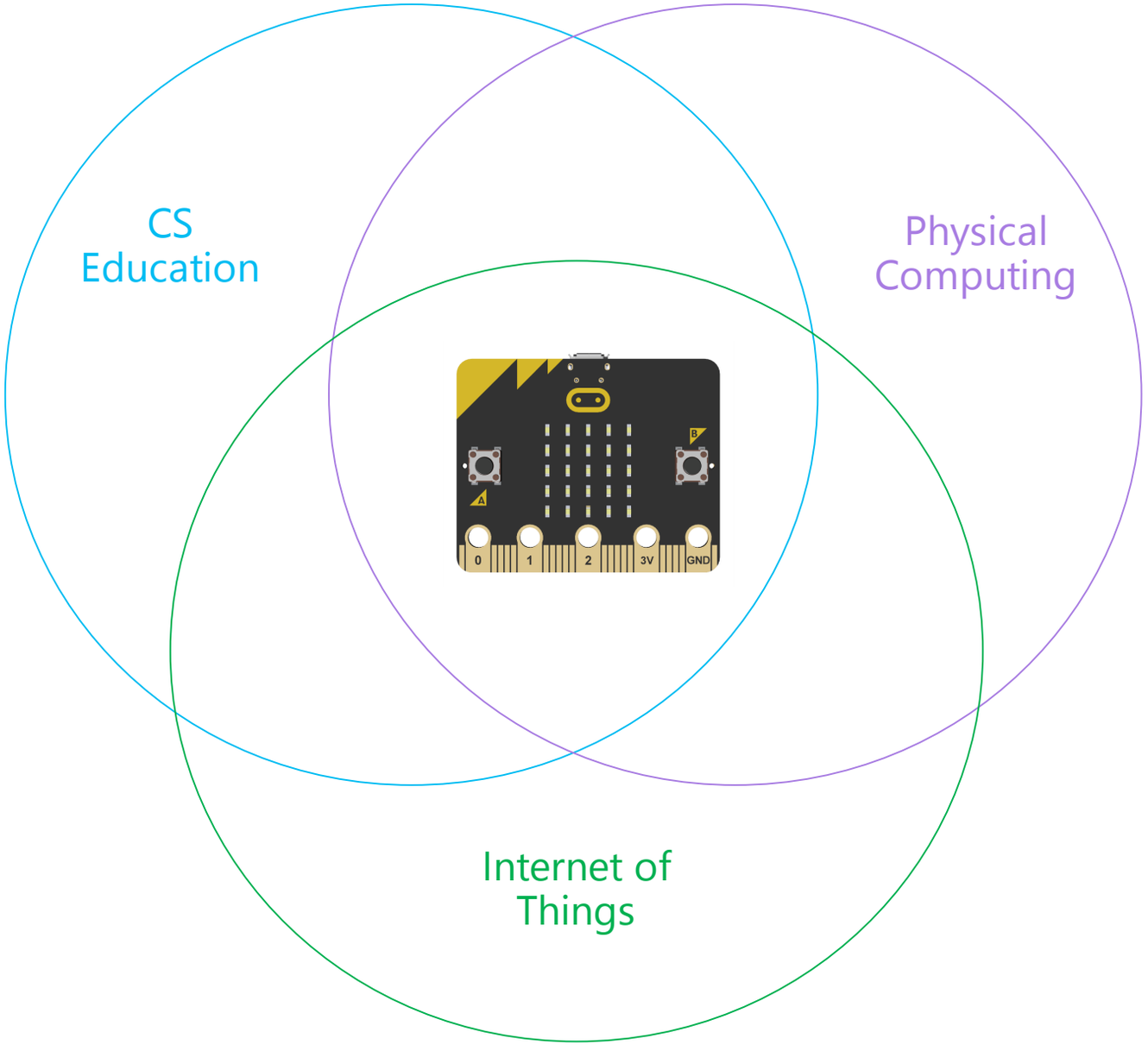






Microsoft Programming Experience Toolkit

- Support new generation of microcontroller-based devices
- A simple experience combining
  - Programming progression
  - Networking of devices
  - Data analysis
- Open source
  - http://github.com/microsoft/pxt
  - http://github.com/microsoft/pxt-microbit
  - http://github.com/microsoft/pxt-arduino


CS
Education
Physical
Computing
Internet of
Things
A
B
0
1
2
3V
GND

THE SATELLITE INVENTORS KIT FOR THE BBC MICRO:BIT

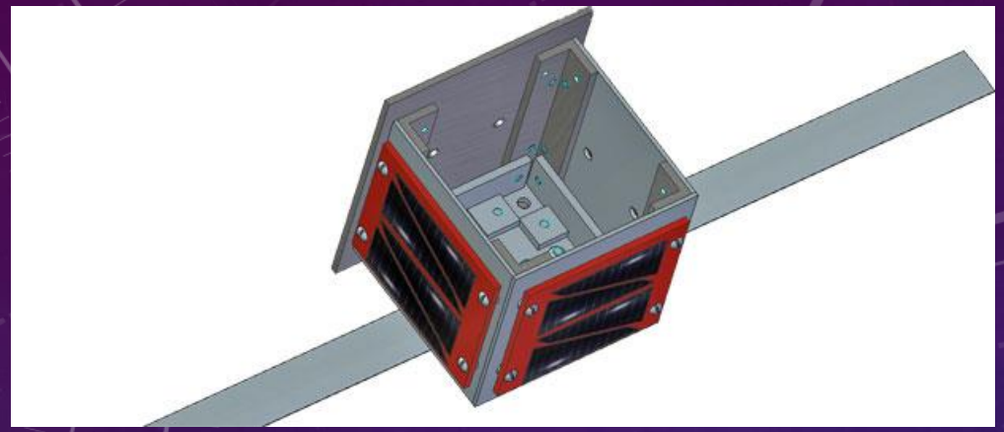
INSPIRING THE NEXT GENERATION OF SCIENTISTS AND ENGINEERS

ELENA BRANET, PAUL FOSTER

MICROSOFT UK, WITH THE SPACE APPLICATION CATAPULT

Microsoft

Nano Satellites



The Catapult PocketQube Satellite Kit

- A complete satellite in a 5 cm cube
- A fully functional, self-contained, remotely operated platform

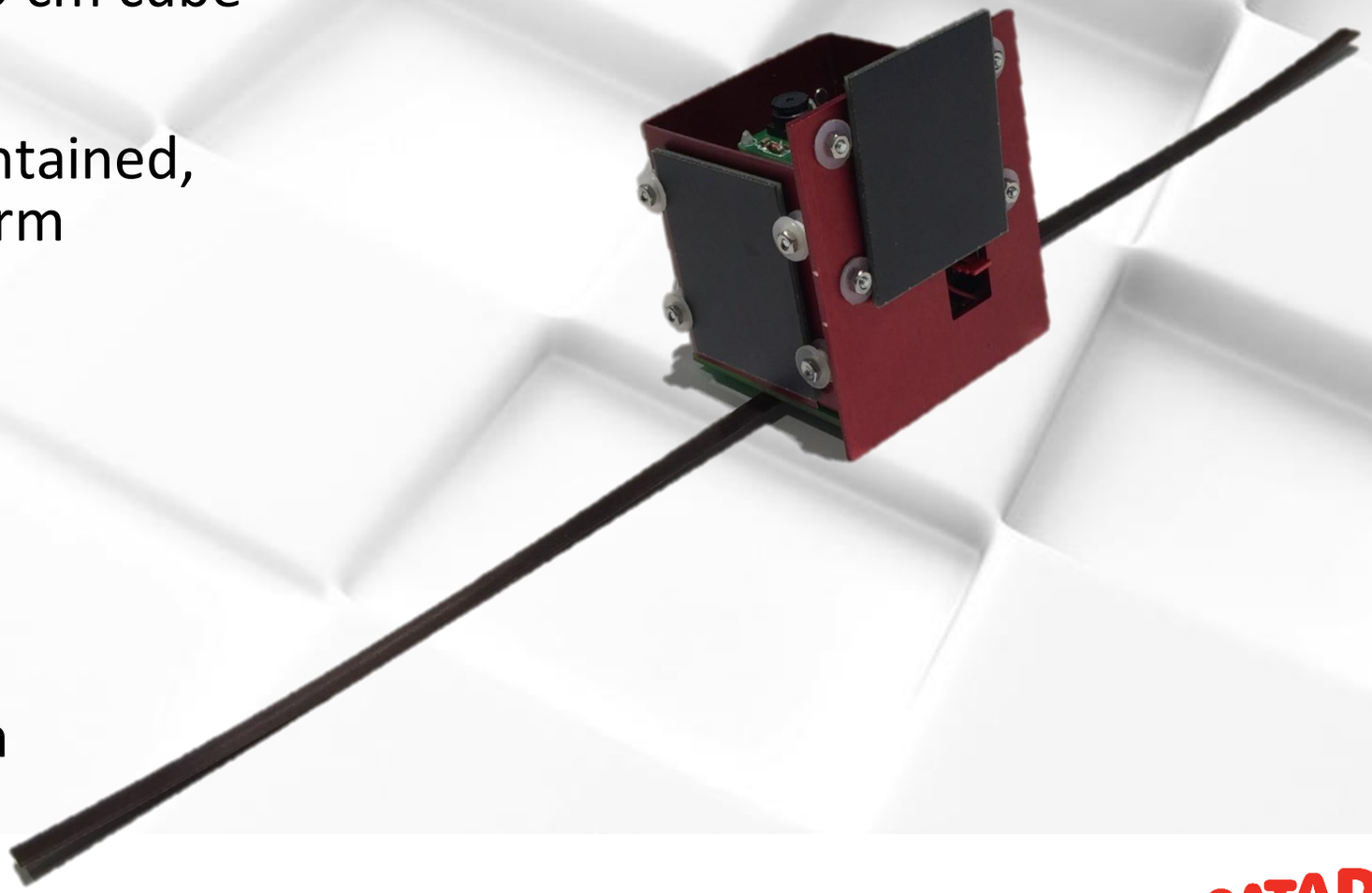
Education

Training

Experiment

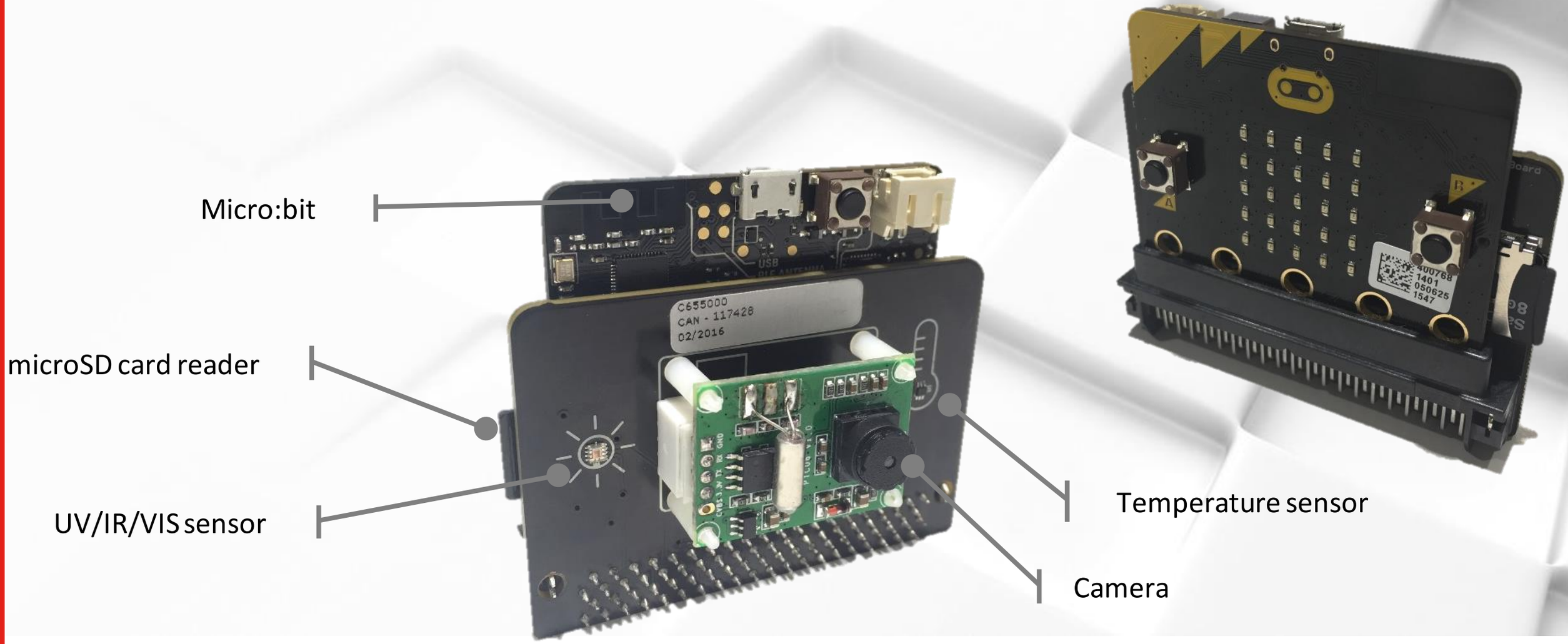
Instrument

Mission design



CATAPULT Satellite Applications

The Micro:bit Satellite Inventors Kit

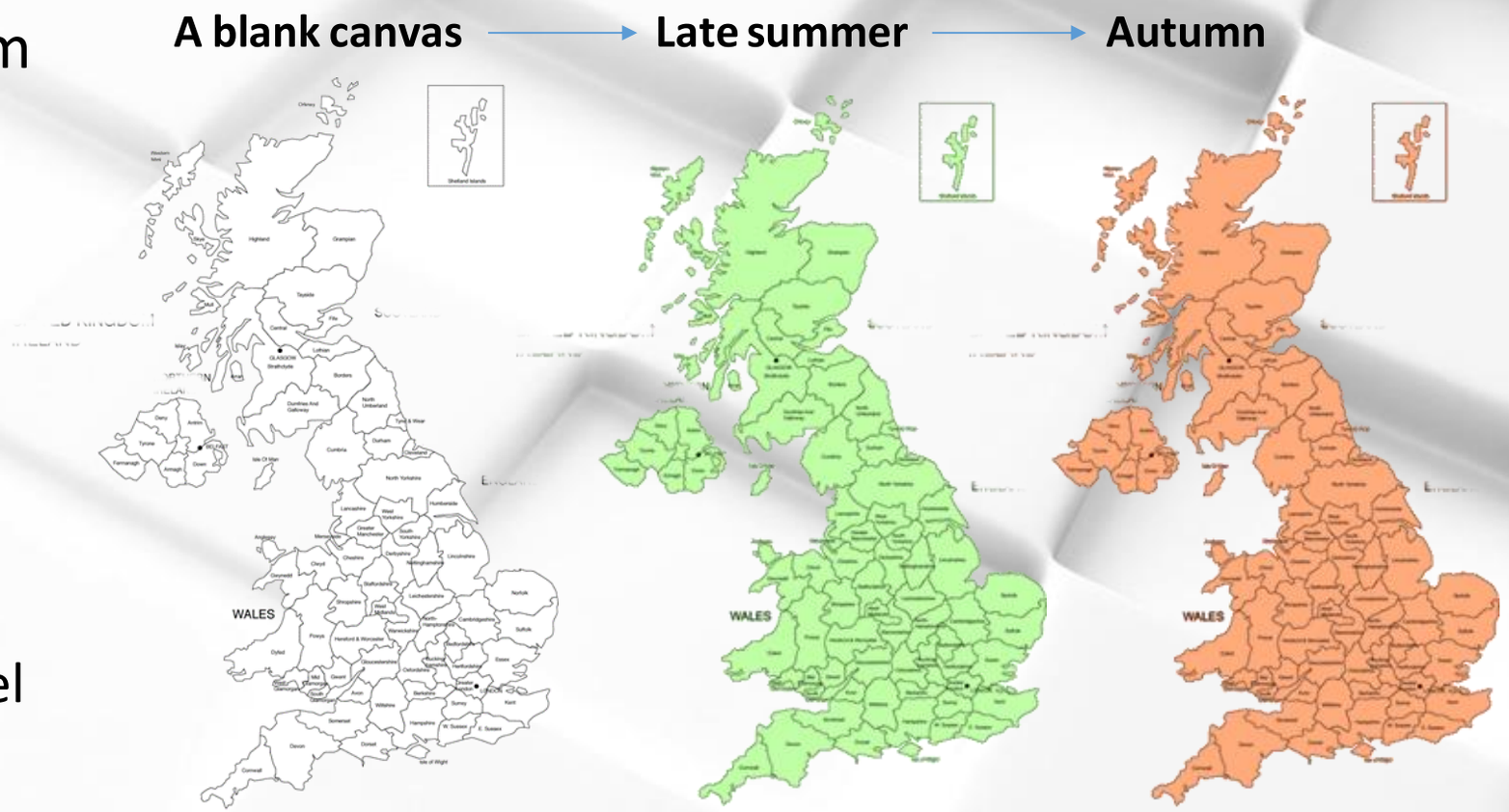


Concepts

- **Coding**
  - Real world problems and physical systems
  - Developing creativity
- **Design and technology**
  - Iterative design and building
- **Biology**
  - Biological ecosystems
  - Plant reproduction and the environment
  - Photosynthetic processes
  - Photosynthesis and the atmosphere
  - Adaptation of leaves for photosynthesis
- **Chemistry**
  - The carbon cycle
- **Physics**
  - Radiation, convection and conduction
  - Gravity as a force
  - Waves
  - Electricity
  - Magnetism
  - Forces
  - The Sun
  - The seasons
- **Geography**
  - Interpreting geographical info

A national crowd-sourced data experiment

- Using micro:bits and satellite inventor's kits in the classroom
- Minimum of one per county
- Crowd-source monitoring the change in seasons
  - Collect an image
  - Process the image data
  - Upload the data point to the cloud
  - Geotag data to locate the pixel
  - Play back to demonstrate the change in season



http://www.freeusandworldmaps.com/images/Countries_Maps/EuropeanCountries/UnitedKingdomMaps/UnitedKingdomPrint.jpg

Call to Action

- Educators to incorporate micro:bit into courses
- Research partners to extend the PXT platform
- Hardware partners to extend to new devices

- See me during Faculty Summit to get a micro:bit

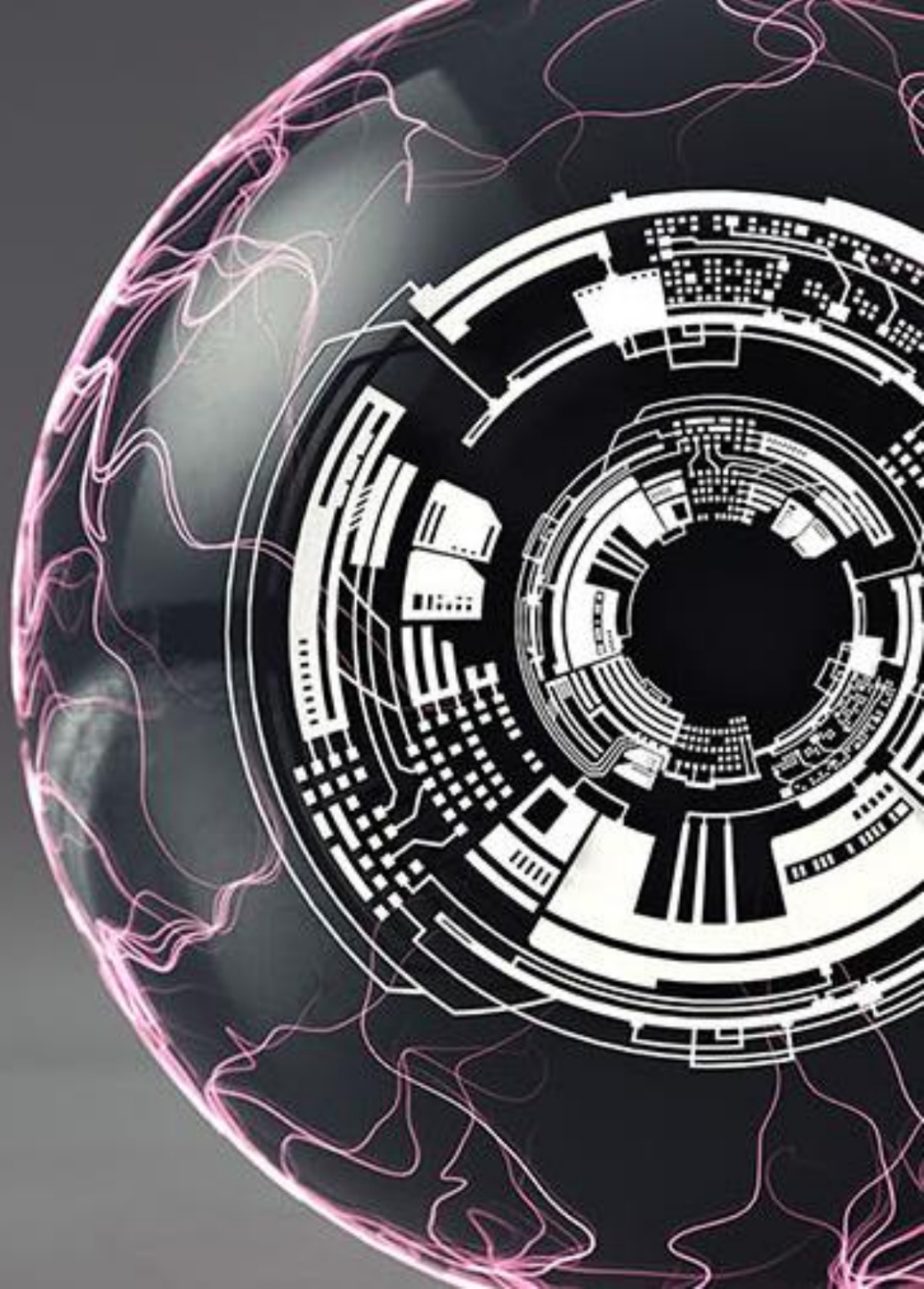
- Follow-up via e-mail to tball@microsoft.com

Microsoft

The micro:bit runtime Inside and Out

Joe Finney
Lancaster University, UK

joe@comp.lancs.ac.uk



- 25 LED matrix screen
- Light sensor
- User definable buttons

- 17 Digital input/output
- 6 Analog input
- 3 PWM output
- 3 Touch sensitive
- I2C, SPI, UART

BBC micro:bit

- 16MHz ARM Cortex M0
- 16KB RAM, 256K FLASH

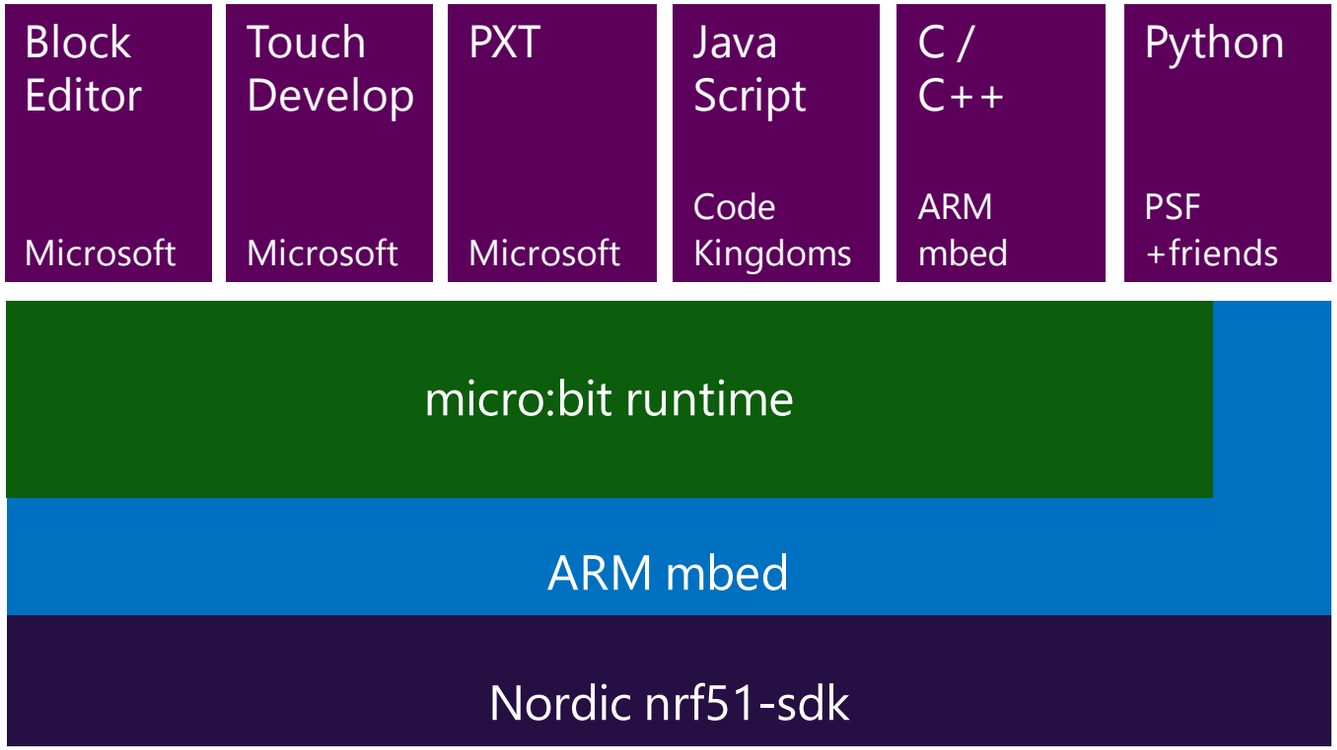
- USB Storage/Serial/Debug

- 3 axis accelerometer
- 3 axis magnetometer
- Temperature sensor
- Bluetooth Low Energy

BBC micro:bit

micro:bit runtime architecture

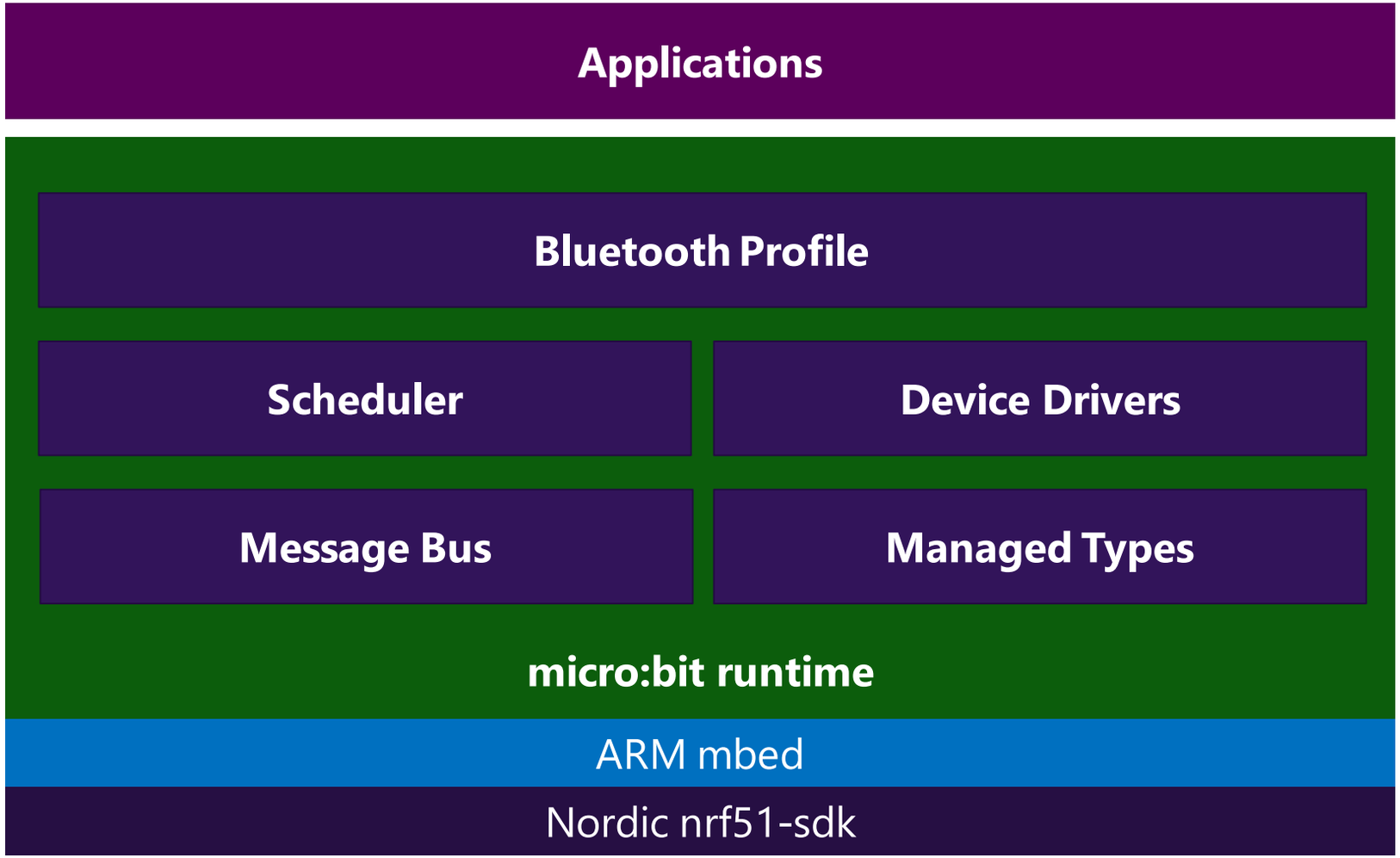
The micro:bit community encourages many languages…



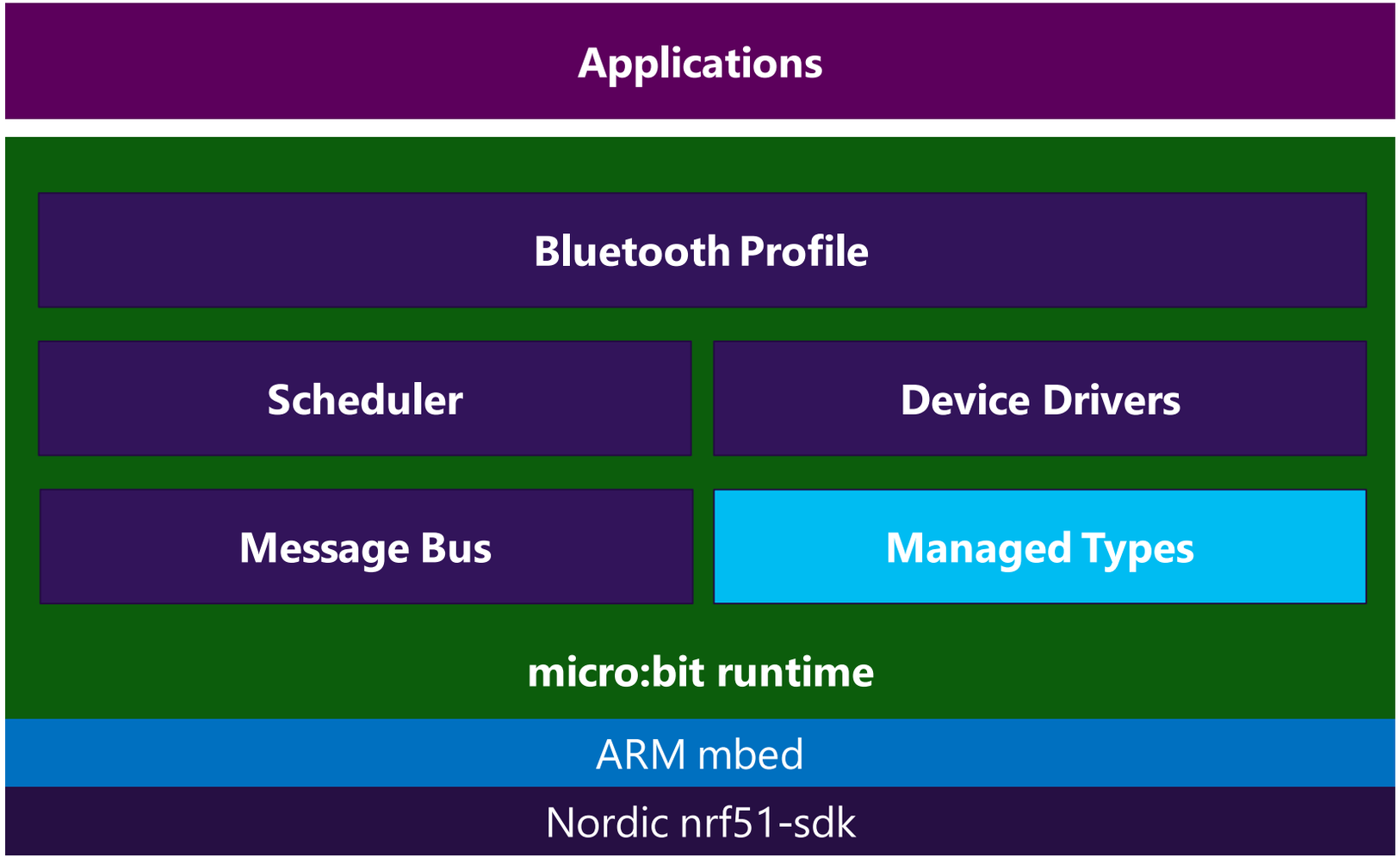
Introducing the micro:bit runtime

- Provides a Device Abstraction Layer for the micro:bit…
  - Open source C/C++ component based API
  - Designed with many requirements in mind:
  - High level language features (concurrency, eventing models and memory safety)
  - Native C/C++ friendliness
  - RAM efficiency
  - Power efficiency

micro:bit runtime architecture



micro:bit runtime architecture



Managed Types

- C is a great language for building software that works with hardware…
  - as it gives a lot of **power** to its users.

- Higher level languages are great for building applications
  - as they make it **easy, robust and simple** for the user.

Memory Management is a key distinction. e.g. take some classic C code:

```
char *s = malloc(10);

strcpy(s, "hello");
doSomething(s);
```

```
void
doSomething(char *text)
{
    …
}
```

who is responsible for freeing the data?

Managed Types

- Modern high level languages assume this is handled by their runtime - so we do!

- Commonly used data types (strings, images, packets) all have their own data type
- Uses **reference counting** to track when the data is used (simpler, but similar principle to JVM, CLR)
- Transparent to users and high level languages. Feels like a higher level language…

```
ManagedString s = "hello";

doSomething(s);
```

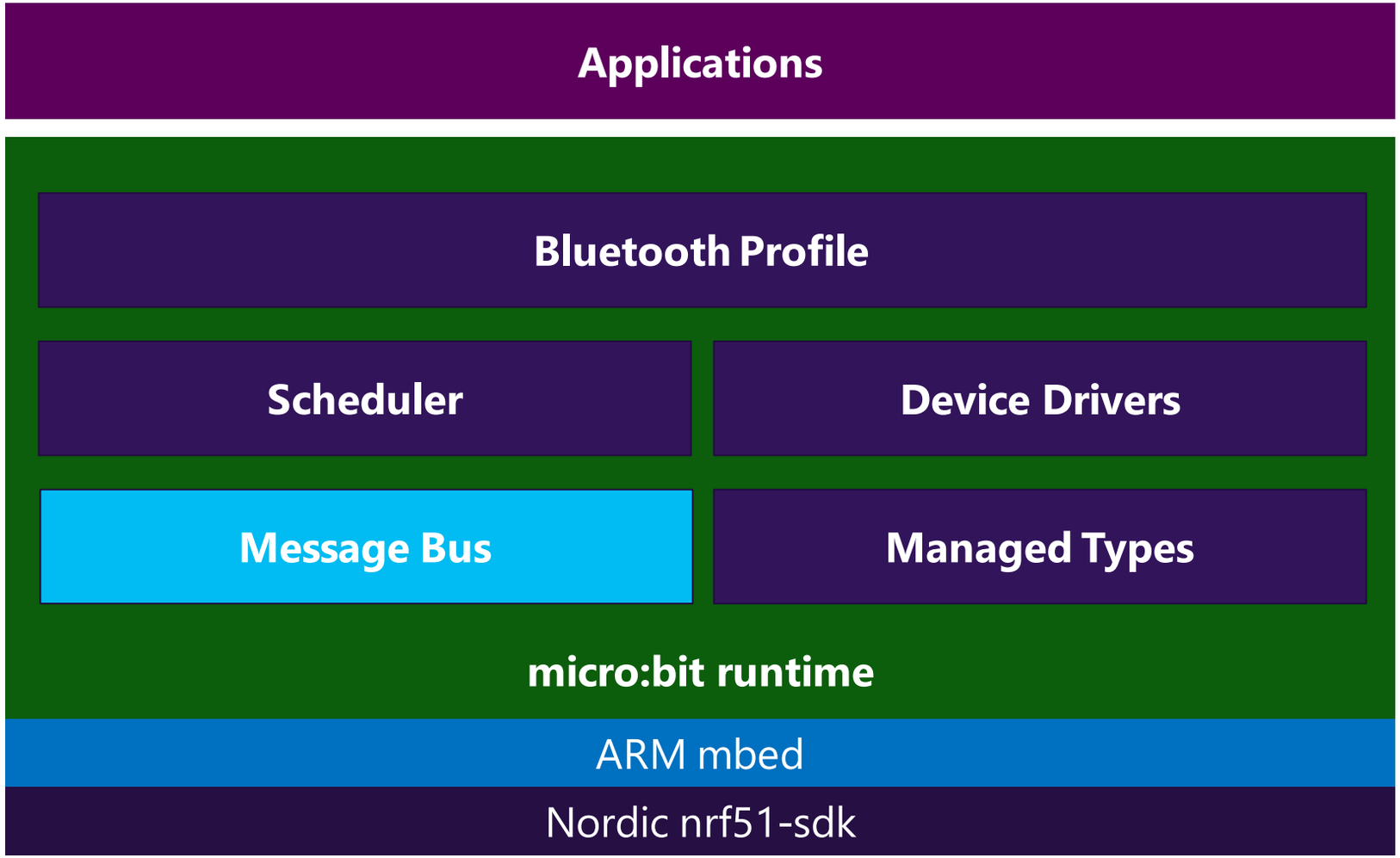
```
void
doSomething(ManagedString text)
{
    …
}
```


Managed Types

- Higher level languages can then more easily map onto the runtime.
- It also provides a clean, easy to use API for C/C++ users:

```
ManagedString s, t, message, answer;

s = “hello”;
t = “world”;

message = s + “ “ + t;

answer = “The answer is:” + 42;

if (message == answer)
    …
```

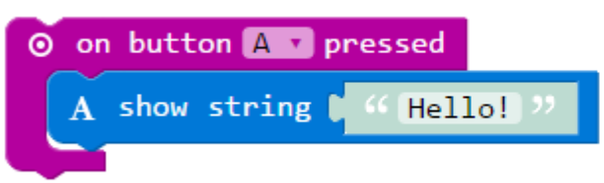

micro:bit runtime architecture

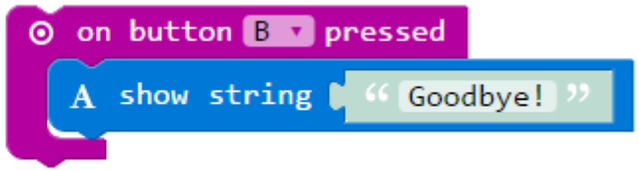


Eventing and the Message Bus

- Many languages support the concept of events.

- This is also something that kids find familiar from visual languages such as Scratch.
- And something that lends itself to embedded systems too… e.g.





Eventing and the Message Bus

- The micro:bit runtime contains a simple yet powerful extensible eventing model

- Events are themselves a very simple managed type.
- Contain two numeric values: a **source** and a **value**.

- Every component in the runtime has a unique ID – the source of an event.
- Each component can then create ANY value with that ID as a source at any time:

```
MicroBitEvent e(MICROBIT_ID_GESTURE, MICROBIT_ACCELEROMETER_EVT_SHAKE);
```

```
#define MICROBIT_ID_GESTURE                     27
#define MICROBIT_ACCELEROMETER_EVT_SHAKE        11
```


Eventing and the Message Bus

- The **MessageBus** then delivers events to any code that registers an interest.

- Functions can be either plain C functions, or C++ methods.
- Wildcard values can also be used to capture lots of events at once.
- There's also a matching **ignore** function in case you want to stop receiving events.

```
void onShake(MicroBitEvent e)
{
    // do something cool here!
}

int main()
{
    uBit.messageBus.listen(MICROBIT_ID_GESTURE, MICROBIT_ACCELEROMETER_EVT_SHAKE, onShake);
}
```


Eventing and the Message Bus

- The **MessageBus** then delivers events to any code that registers an interest.

- Functions can be either plain C functions, or C++ methods.
- Wildcard values can also be used to capture lots of events at once.
- There's also a matching **ignore** function in case you want to stop receiving events.

```
void onGesture(MicroBitEvent e)
{
    if (e.value == MICROBIT_ACCELEROMETER_EVT_SHAKE) …
}

int main()
{
    uBit.messageBus.listen(MICROBIT_ID_GESTURE, MICROBIT_EVT_ANY, onGesture);
}
```


Eventing and the Message Bus

- The **MessageBus** then delivers events to any code that registers an interest.

- Functions can be either plain C functions, or C++ methods.
- Wildcard values can also be used to capture lots of events at once.
- There's also a matching **ignore** function in case you want to stop receiving events…

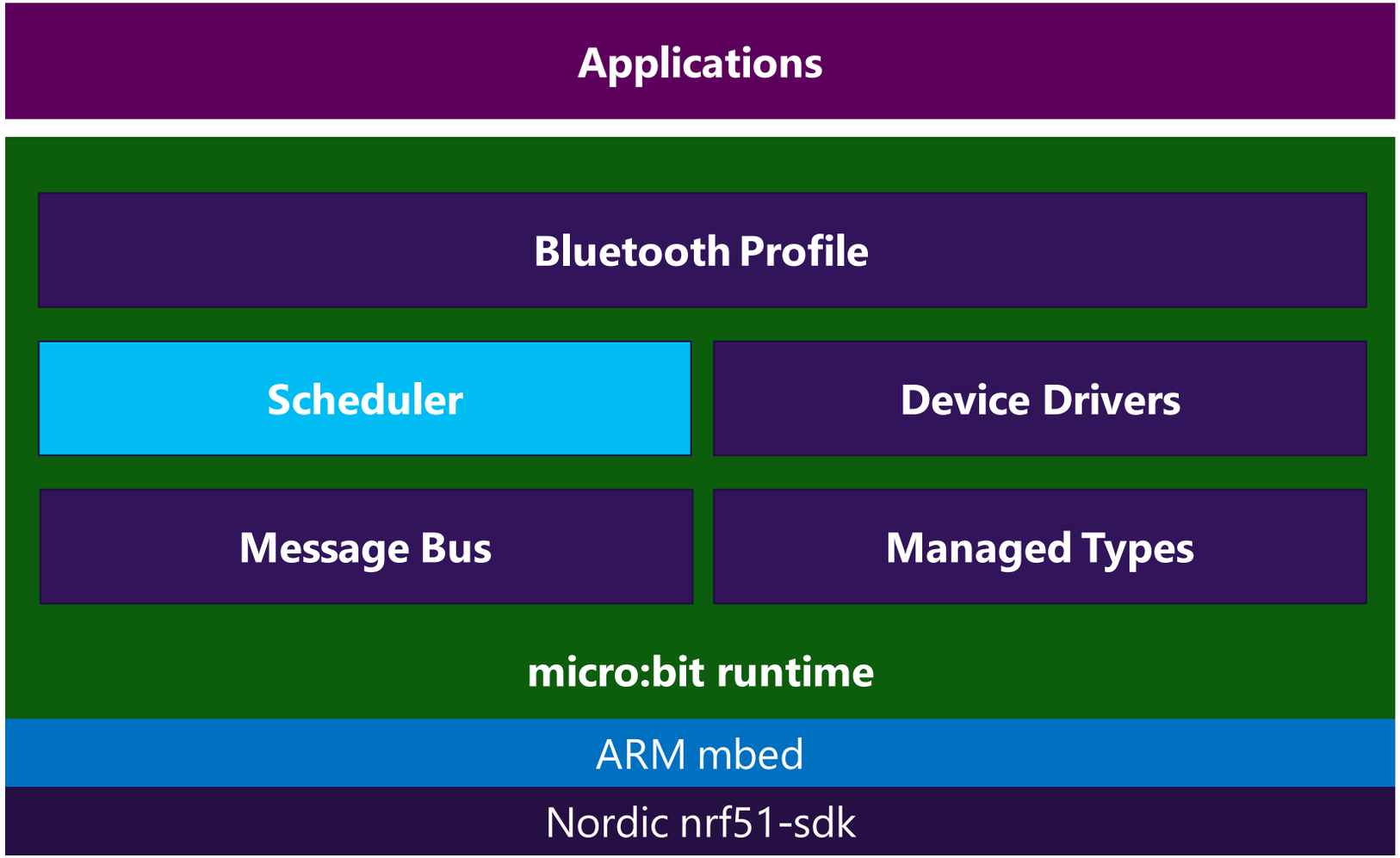
```
void onEvent(MicroBitEvent e)
{
    if (e.source == MICROBIT_ID_GESTURE) …
}

int main()
{
    uBit.messageBus.listen(MICROBIT_ID_ANY, MICROBIT_EVT_ANY, onEvent);
}
```


Eventing and the Message Bus

- The runtime generates a range of events application can build on.
  - Users can also define their own events easily… just numbers!

```
#define MICROBIT_ACCELEROMETER_EVT_TILT_UP              1
#define MICROBIT_ACCELEROMETER_EVT_TILT_DOWN            2
#define MICROBIT_ACCELEROMETER_EVT_TILT_LEFT            3
#define MICROBIT_ACCELEROMETER_EVT_TILT_RIGHT           4
#define MICROBIT_ACCELEROMETER_EVT_FACE_UP              5
#define MICROBIT_ACCELEROMETER_EVT_FACE_DOWN            6
#define MICROBIT_ACCELEROMETER_EVT_FREEFALL             7
#define MICROBIT_ACCELEROMETER_EVT_SHAKE                11

#define MICROBIT_BUTTON_EVT_DOWN                        1
#define MICROBIT_BUTTON_EVT_UP                          2
#define MICROBIT_BUTTON_EVT_CLICK                       3
#define MICROBIT_BUTTON_EVT_LONG_CLICK                  4
#define MICROBIT_BUTTON_EVT_HOLD                        5
#define MICROBIT_BUTTON_EVT_DOUBLE_CLICK                6

#define MICROBIT_RADIO_EVT_DATAGRAM                     1
```

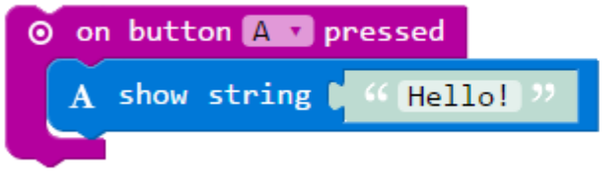

micro:bit runtime architecture

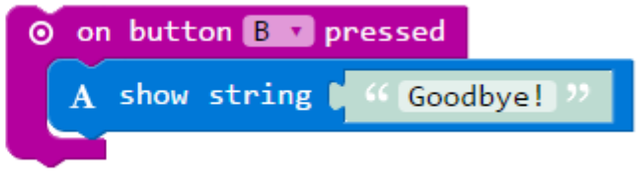


Fiber Scheduler: Providing Concurrent behaviour…

…or at least *apparently* concurrent behaviour!

- Take this simple example again. What behaviour would you expect?
- Given that show string will scroll the given text on the 5x5 matrix display…





Fiber Scheduler: Providing Concurrent behaviour…

- Fibers can be created at any time, and execute independently
- By design, a **non pre-emptive** scheduler to reduce potential race conditions.
- Fibers can sleep, or block on events on the MessageBus
- Anytime there's nothing to do… processor enters a power efficient sleep

```
void doSomething()
{
    while(1)
    {
        uBit.display.print('A');
        uBit.sleep(100);
    }
}
```

```
void doSomethingElse()
{
    while(1)
    {
        uBit.display.print('B');
        uBit.sleep(100);
    }
}
```


Fiber Scheduler: Providing Concurrent behaviour…

- A **RAM optimised** thread scheduler for Cortex processors.

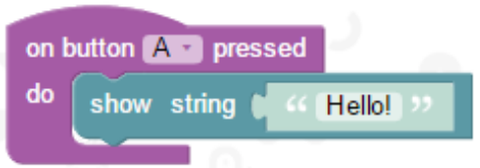
- We adopt a stack duplication approach
- Keeps RAM cost of fibers low, at the expense of CPU time
- Each fiber typically costs ~200 bytes.

- Event handlers (by default) run in their own fiber*
- Effectively decoupling kids' code from nasty interrupt context code.

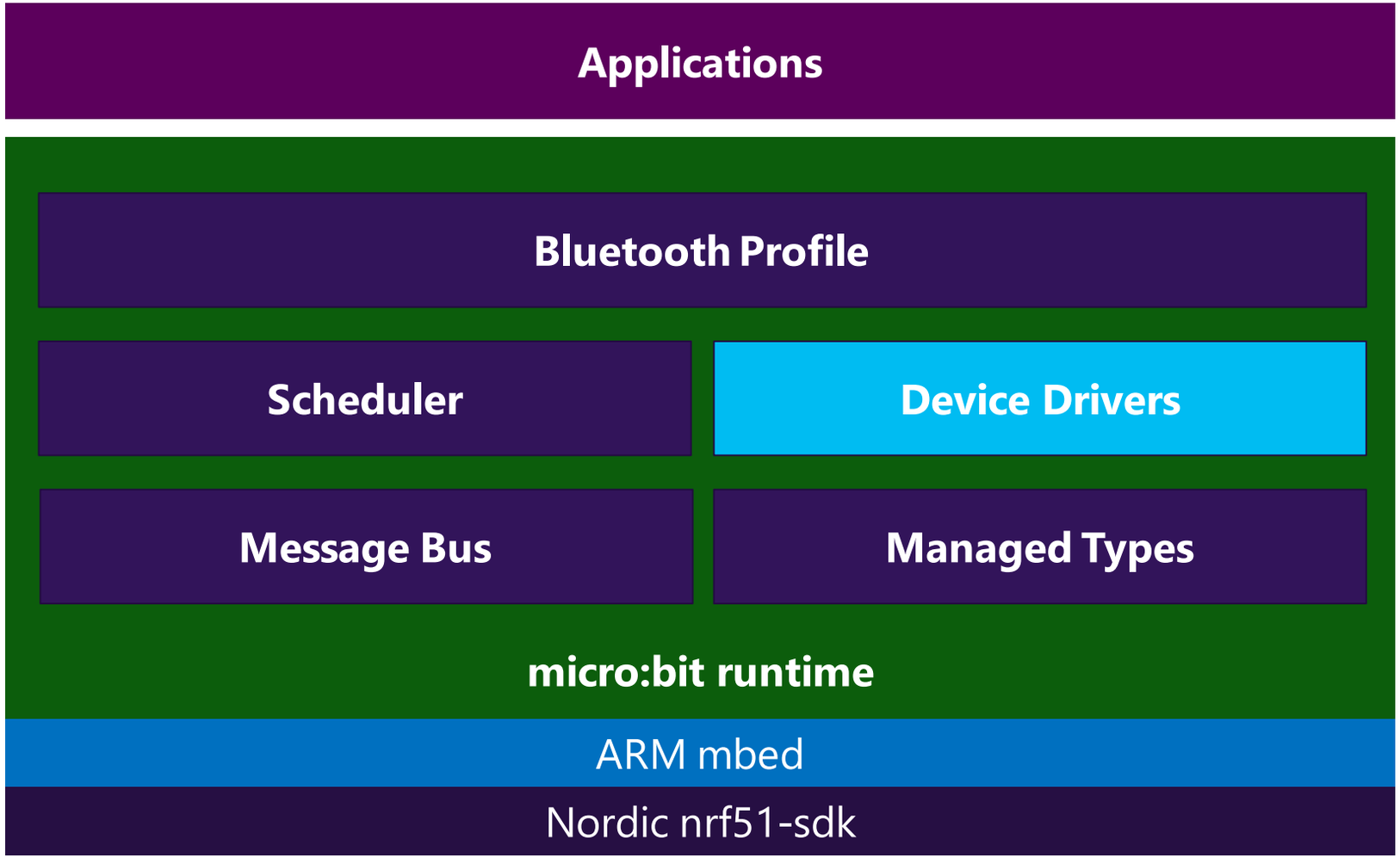
- Functions (e.g. scroll text) can block the calling fiber until the task completes…
- …and event handlers can safely execute users code without risk of locking out the CPU…
- …so our blocks program can simply and efficiently translate to this:

Fiber Scheduler: Providing Concurrent behaviour…

```
void onButtonA()
{
    uBit.display.scroll(“hello”);
}

void onButtonB()
{
    uBit.display.scroll(“goodbye”);
}

// Then in your main program...

uBit.messageBus.listen(MICROBIT_ID_BUTTON_A, MICROBIT_BUTTON_EVT_CLICK, onButtonA);
uBit.messageBus.listen(MICROBIT_ID_BUTTON_B, MICROBIT_BUTTON_EVT_CLICK, onButtonB);
```

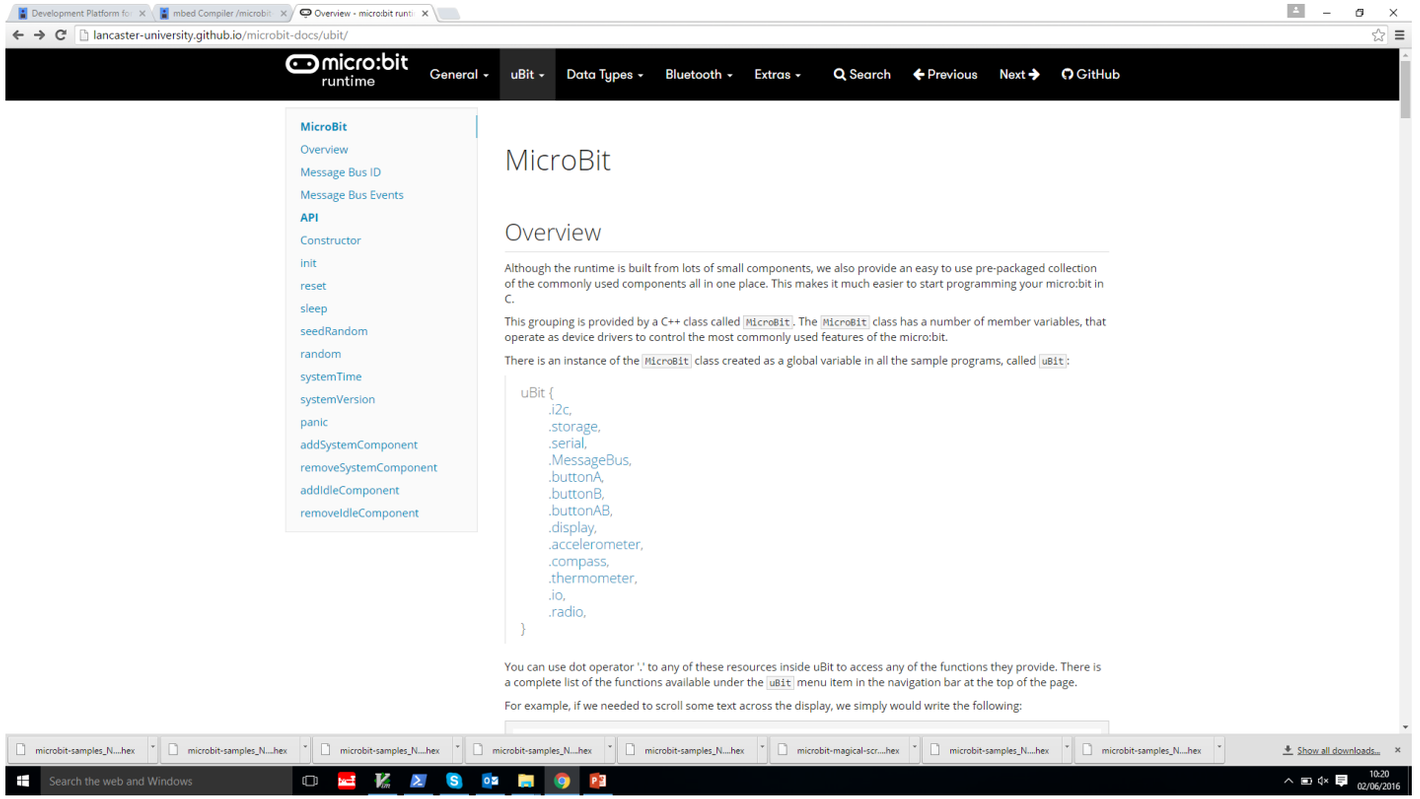


micro:bit runtime architecture



Device Drivers

- Each hardware component is supported by a corresponding C++ software component:
  - MicroBitAccelerometer
  - MicroBitButton
  - MicroBitMultiButton
  - MicroBitCompass
  - MicroBitDisplay
  - MicroBitIO
  - MicroBitLightSensor
  - MicroBitRadio
  - MicroBitSerial
  - MicroBitStorage
  - MicroBitThermometer



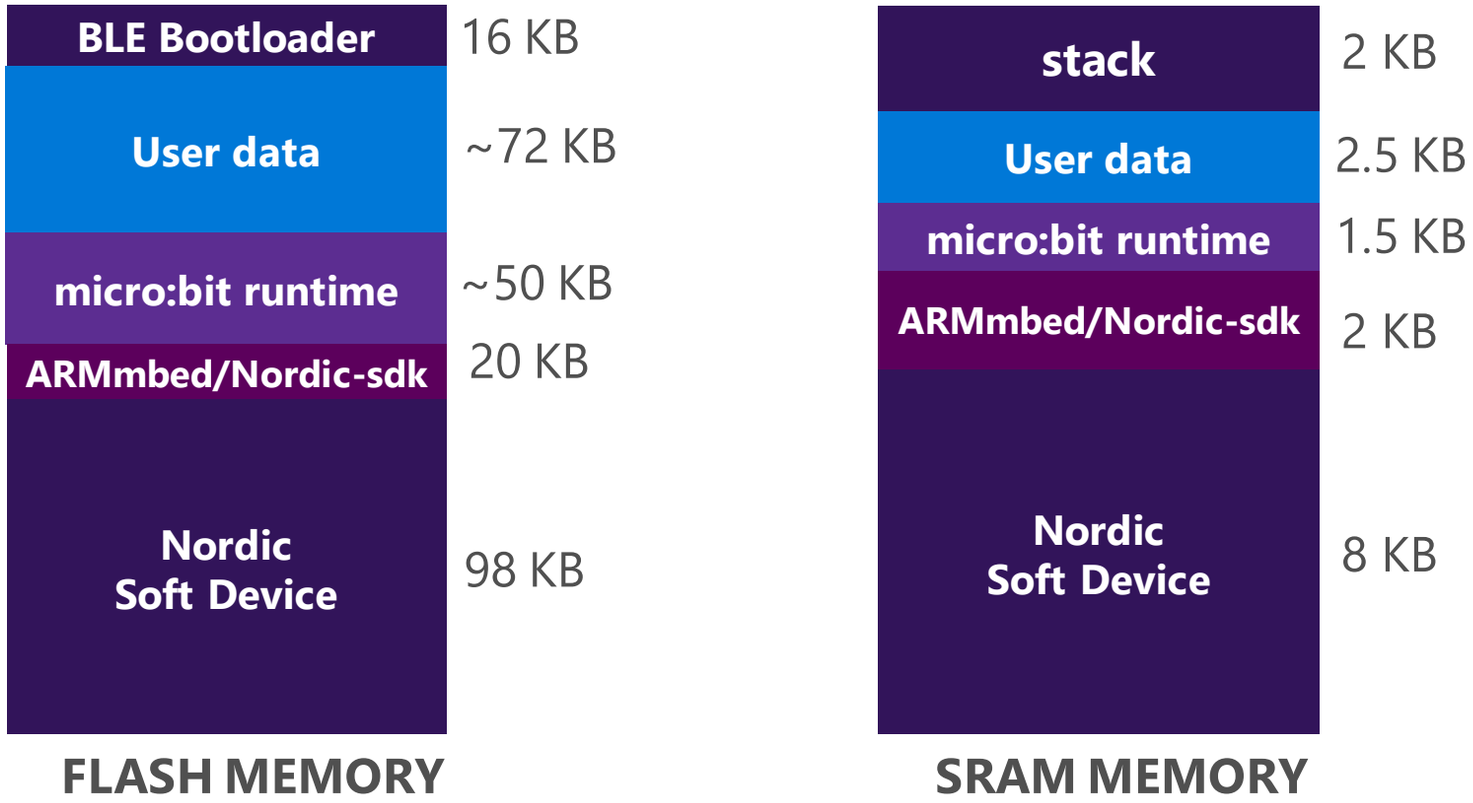
Device Drivers

- Complexity of fine grained initialization too great for most high level languages…
- So we wrap the common set of components together:

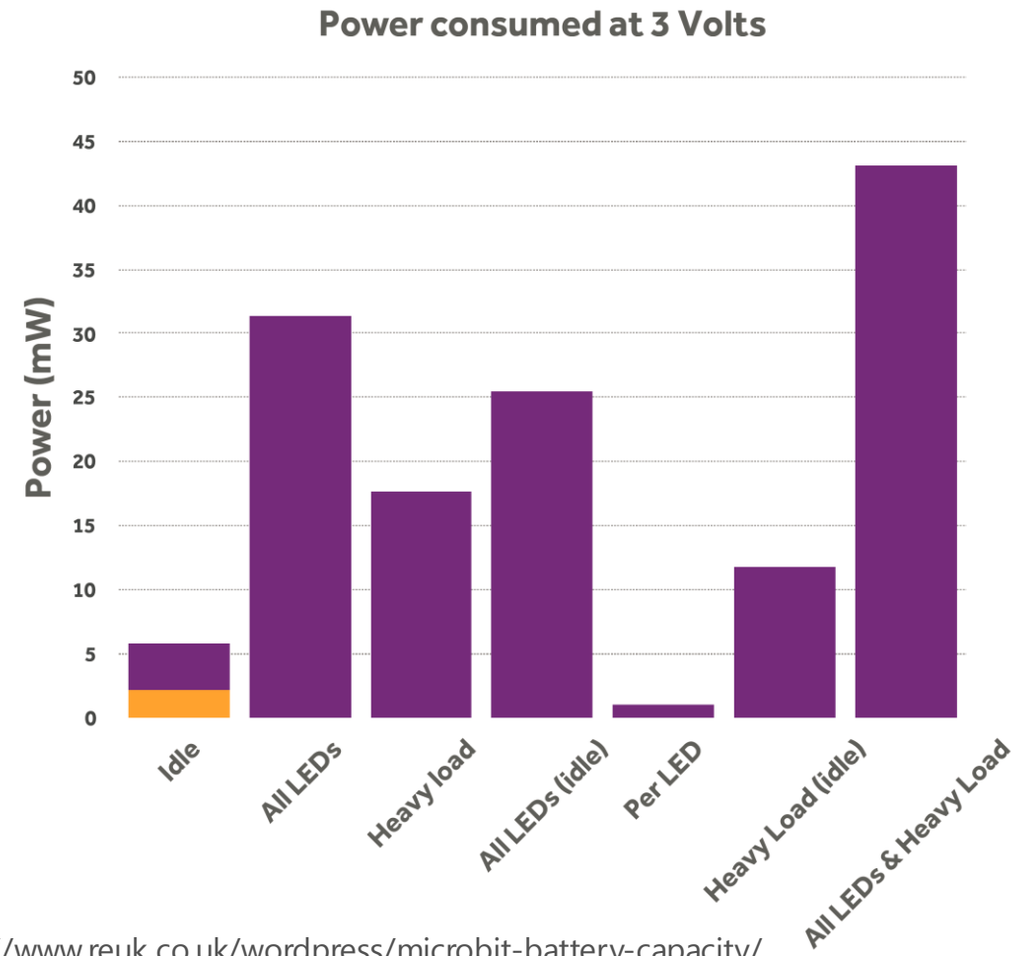
```
MicroBit uBit;

int main()
{
    // initialise runtime
    uBit.init();

    // code!
    uBit.display.scroll("Hello World!");
}
```


Memory Footprint

- micro:bit has 16Mhz Nordic nrf51822 CPU (32 bit Cortex M0)
- 256 KB FLASH memory, 16KB SRAM…



Power Efficiency



http://www.reuk.co.uk/wordpress/microbit-battery-capacity/

Pi 3 ~ 2000mW

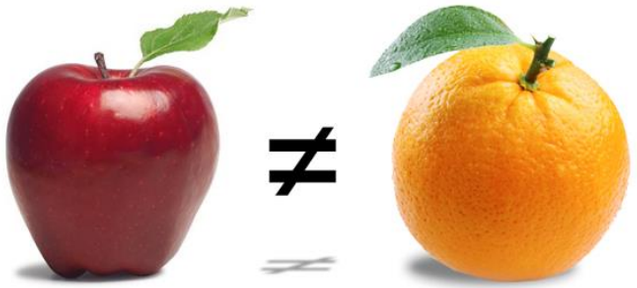
https://www.raspberrypi.org/help/faqs/

Pi Zero ~500mW

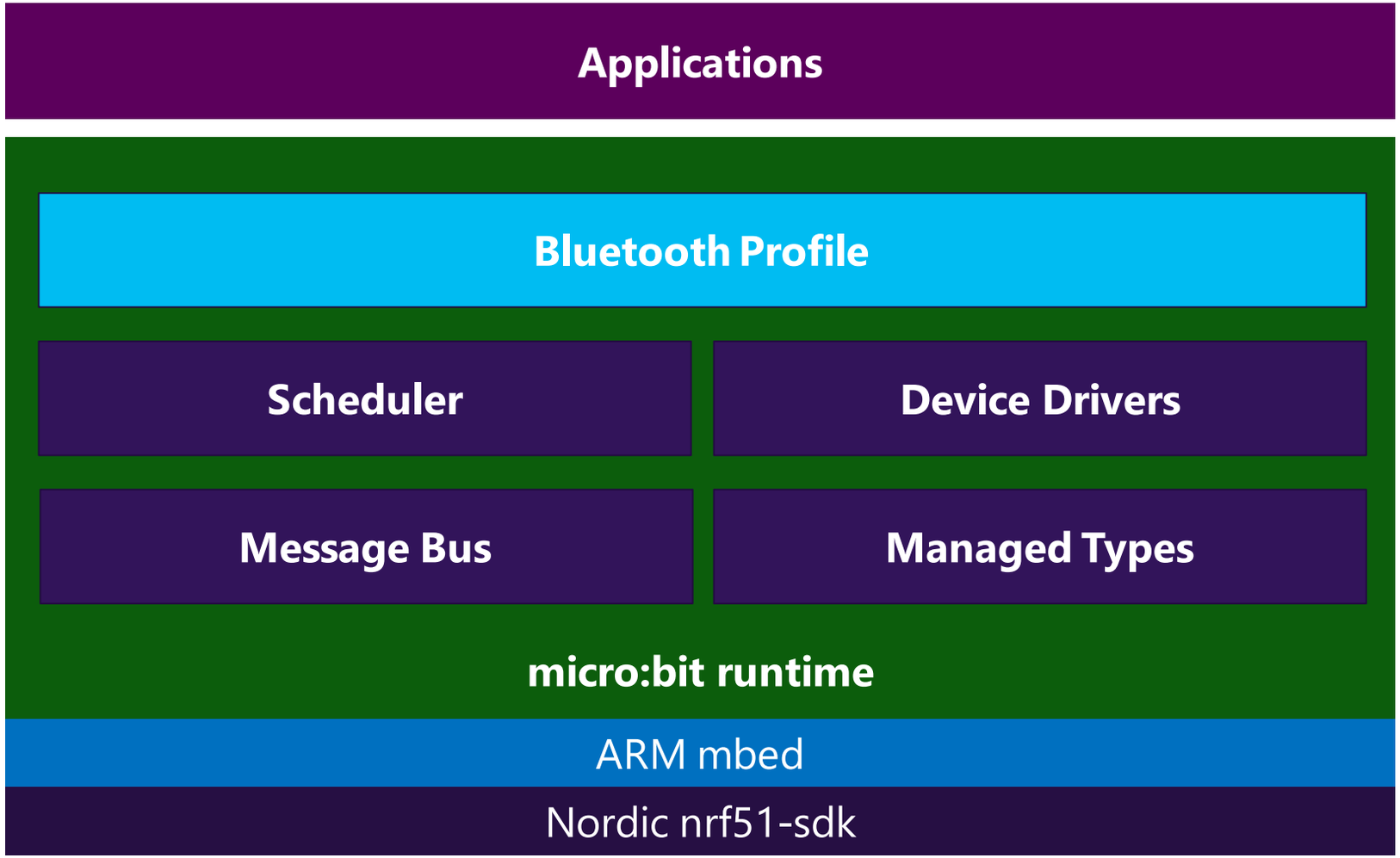
http://raspi.tv/2015/raspberry-pi-zero-power-measurements

Arduino Uno ~400mW

http://forum.arduino.cc/index.php?topic=135872.0



micro:bit runtime architecture

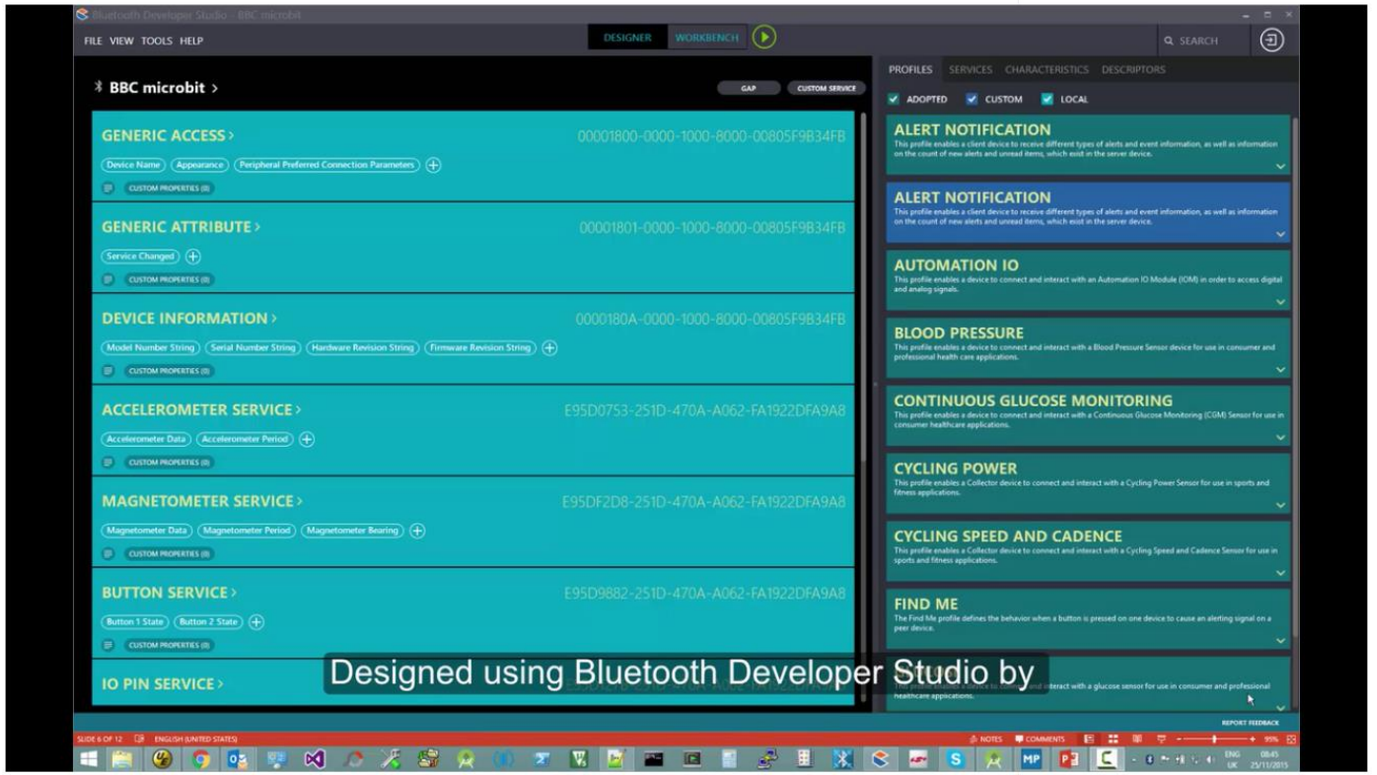


Bluetooth Profile

- Each driver component also mapped as RESTful Bluetooth API…

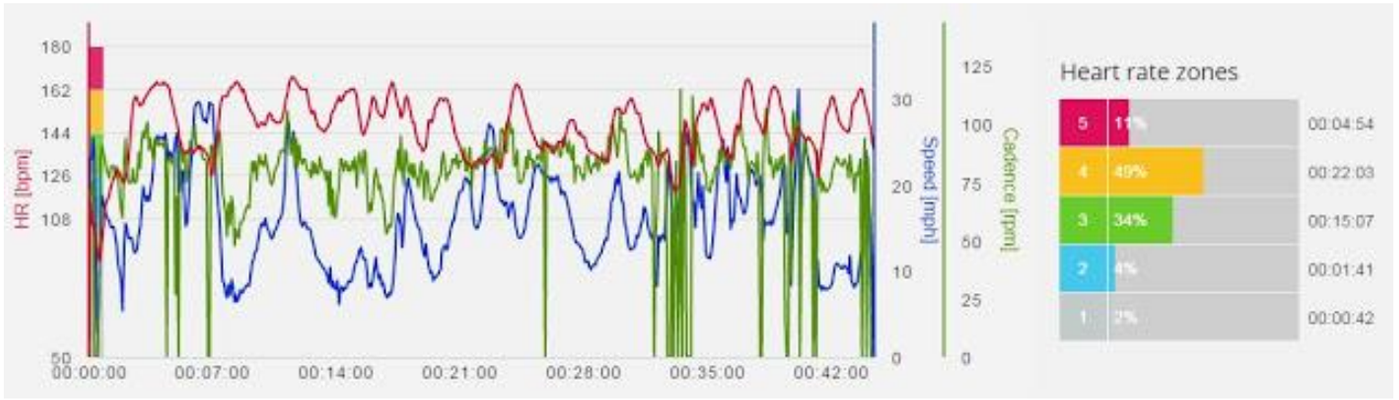
  - MicroBitAccelerometerService
  - MicroBitButtonService
  - MicroBitMagnetometerService
  - MicroBitLEDService
  - MicroBitIOPinService
  - MicroBitTemperatureService
  - MicroBitEventService
  - UARTService
  - DeviceFirmwareUpdate

  - Keyboard HID (coming soon)
  - iBeacon/Eddystone (coming soon)



microbit_profile_overview from Martin Woolley on Vimeo.

Bluetooth Profile



© Martin Woolley Bluetooth SIG

micro:bit runtime @microbitruntime · Jun 8
@duncancragg @sandeepmistry Awesome - The more the merrier guys. In case some of our docs are useful: lancaster-university.github.io/microbit-docs/

2

Sandeep Mistry
@sandeepmistry

Follow

@microbitruntime @bluetooth_mdw @duncancragg Support for controlling one from Node.js (OS X, Linux) is out:

sandeepmistry/node-bbc-microbit
Control a BBC micro:bit from Node.js using BLE. Contribute to node-bbc-microbit development by creating an account on GitHub.
github.com

RETWEETS 3 LIKES 5

10:11 AM - 10 Jul 2016

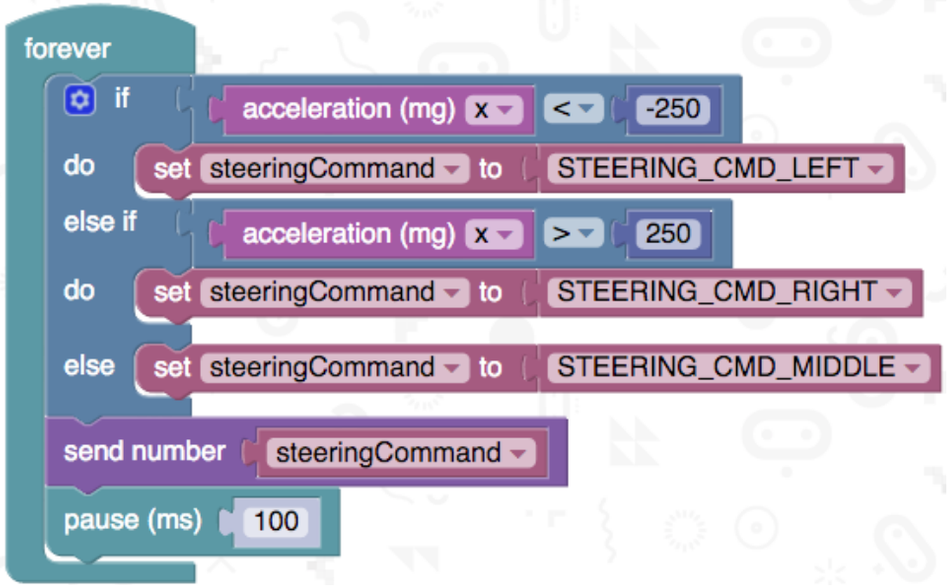
3 5

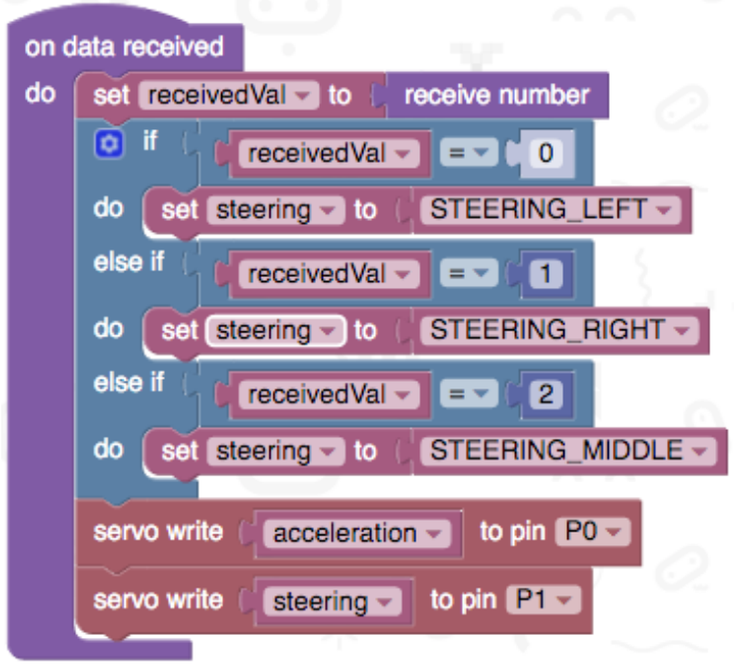
Reply to @sandeepmistry @bluetooth_mdw @duncancragg

http://bluetooth-mdw.blogspot.co.uk/p/bbc-microbit.html
https://play.google.com/store/apps/details?id=com.bluetooth.mwoolley.microbitbledemo

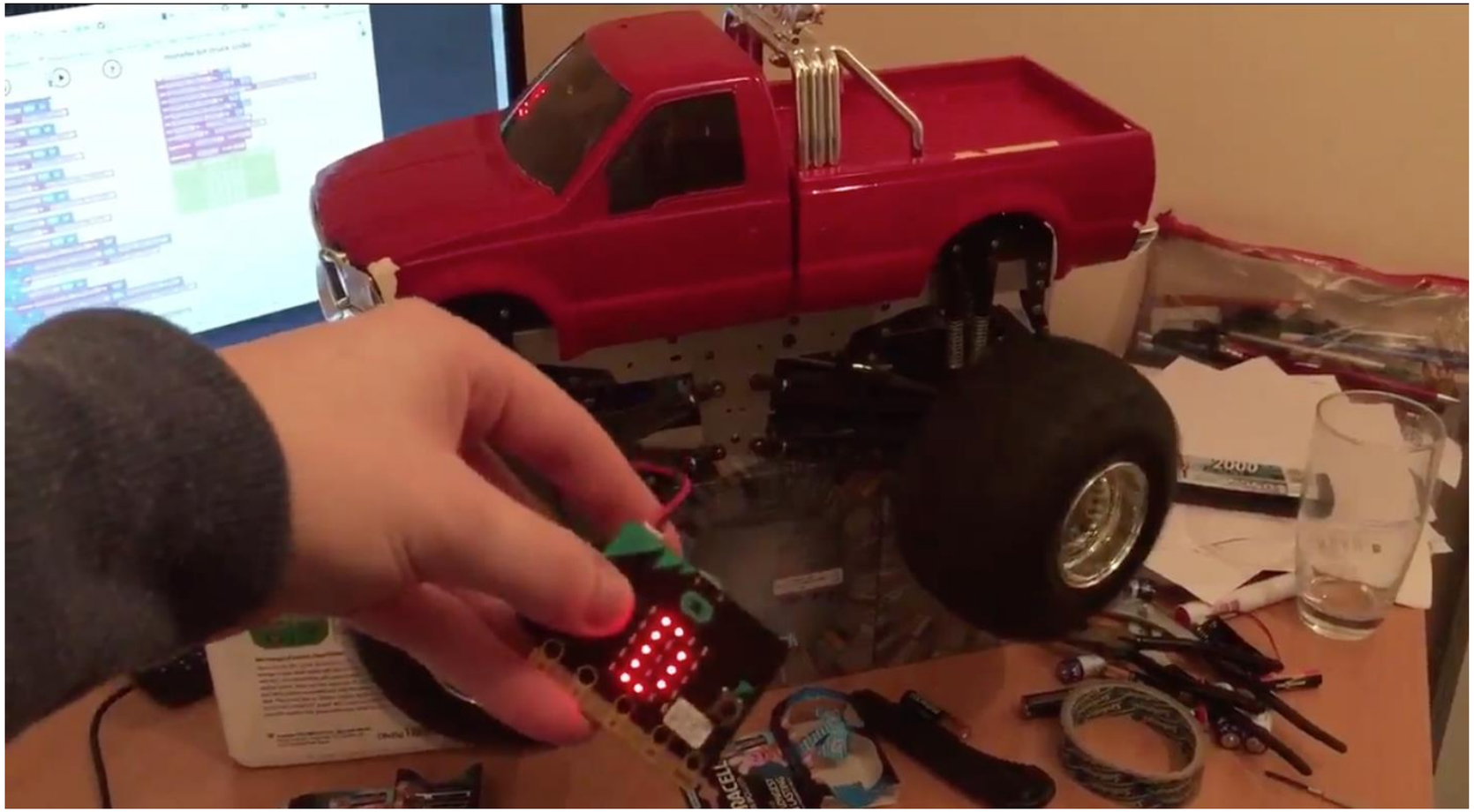
MicroBitRadio

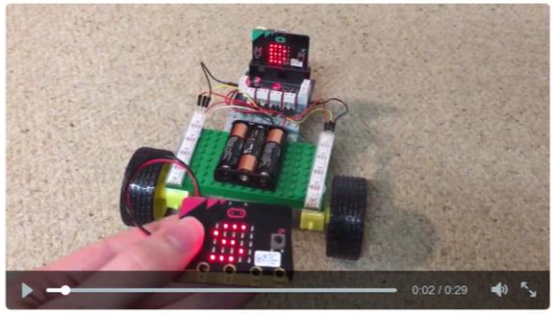
Simple, raw packet communications…

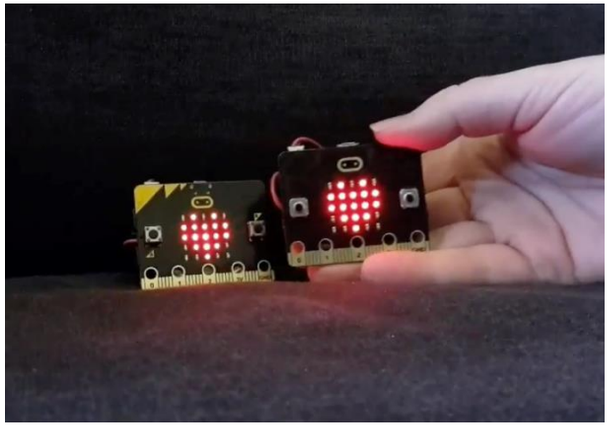




MicroBitRadio







Coming soon to a micro:bit near you

- Features currently under development…
  - On chip file system, exposed through USB interface
  - End-to-end IoT interfaces
  - Platform independence

http://lancaster-university.github.io/microbit-docs/

https://developer.mbed.org/platforms/Microbit/

https://codethemicrobit.com/

https://www.microbit.co.uk/

@microbitruntime

lancaster-university/microbit-dal

Lancaster University