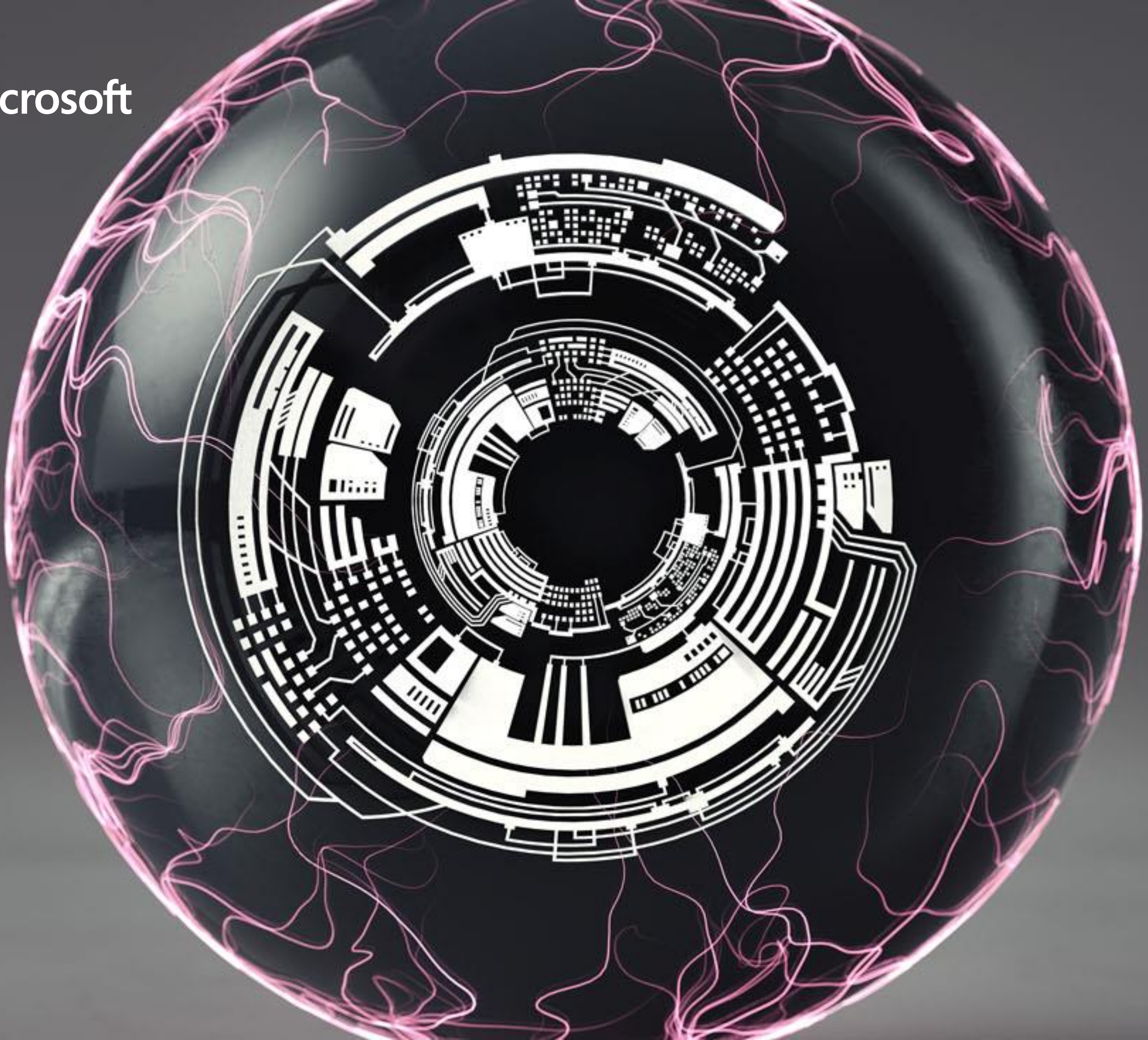
Microsoft



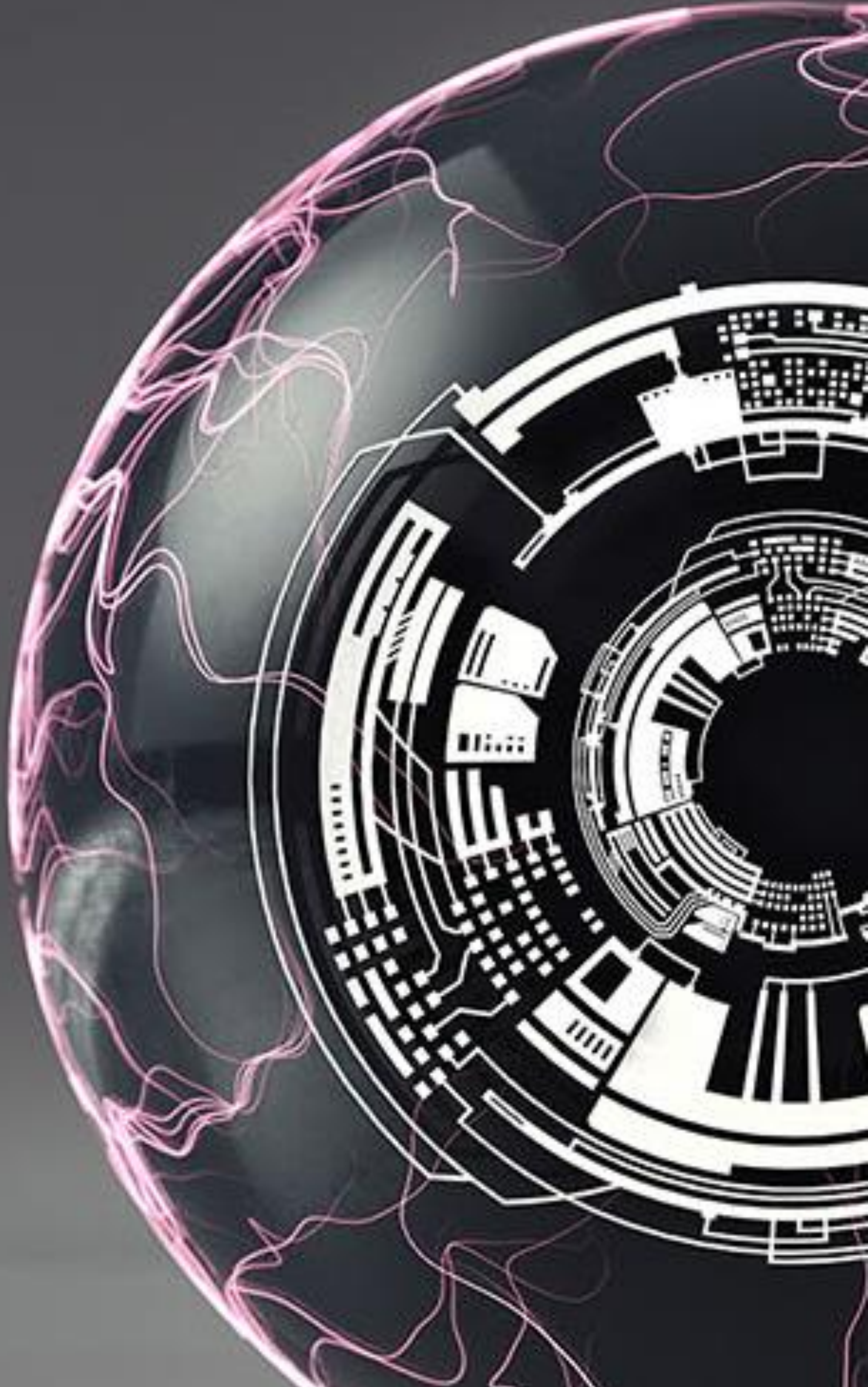
Microsoft Research Faculty Summit 2016

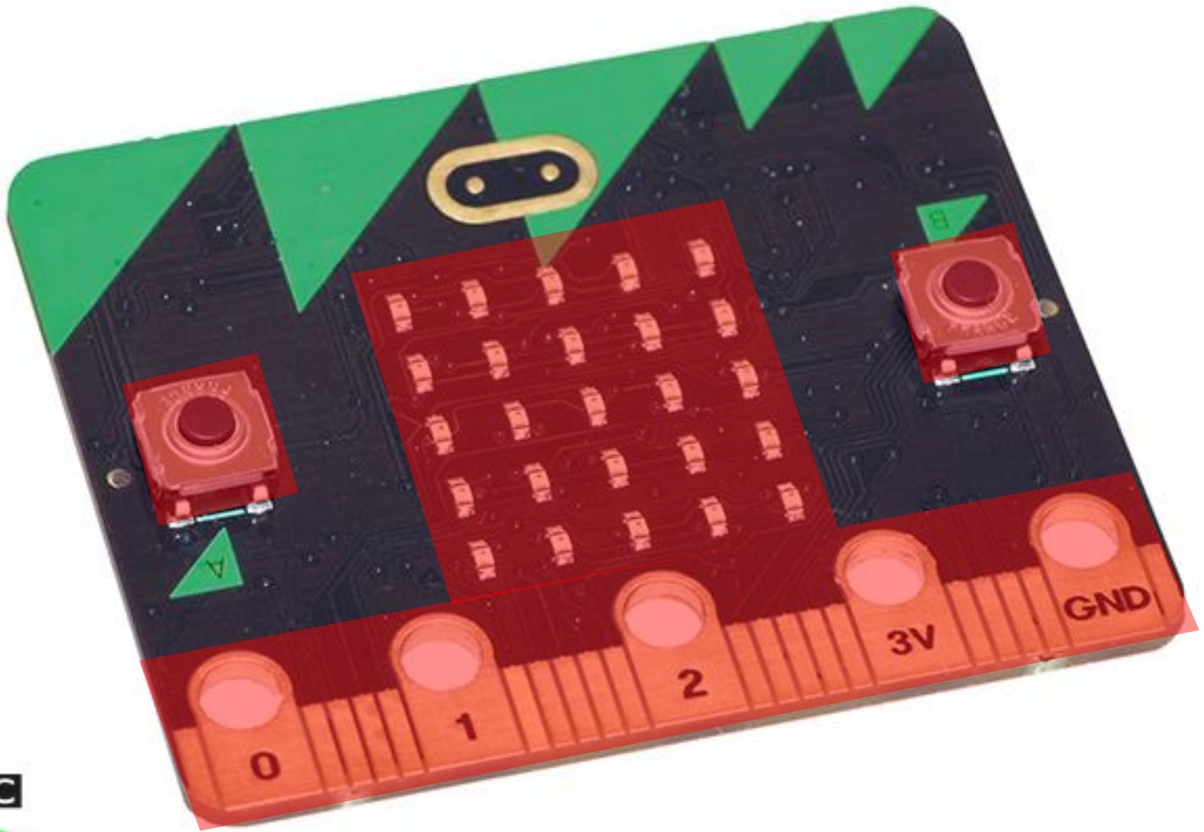
Microsoft

The micro:bit runtime Inside and Out

Joe Finney
Lancaster University, UK

joe@comp.lancs.ac.uk

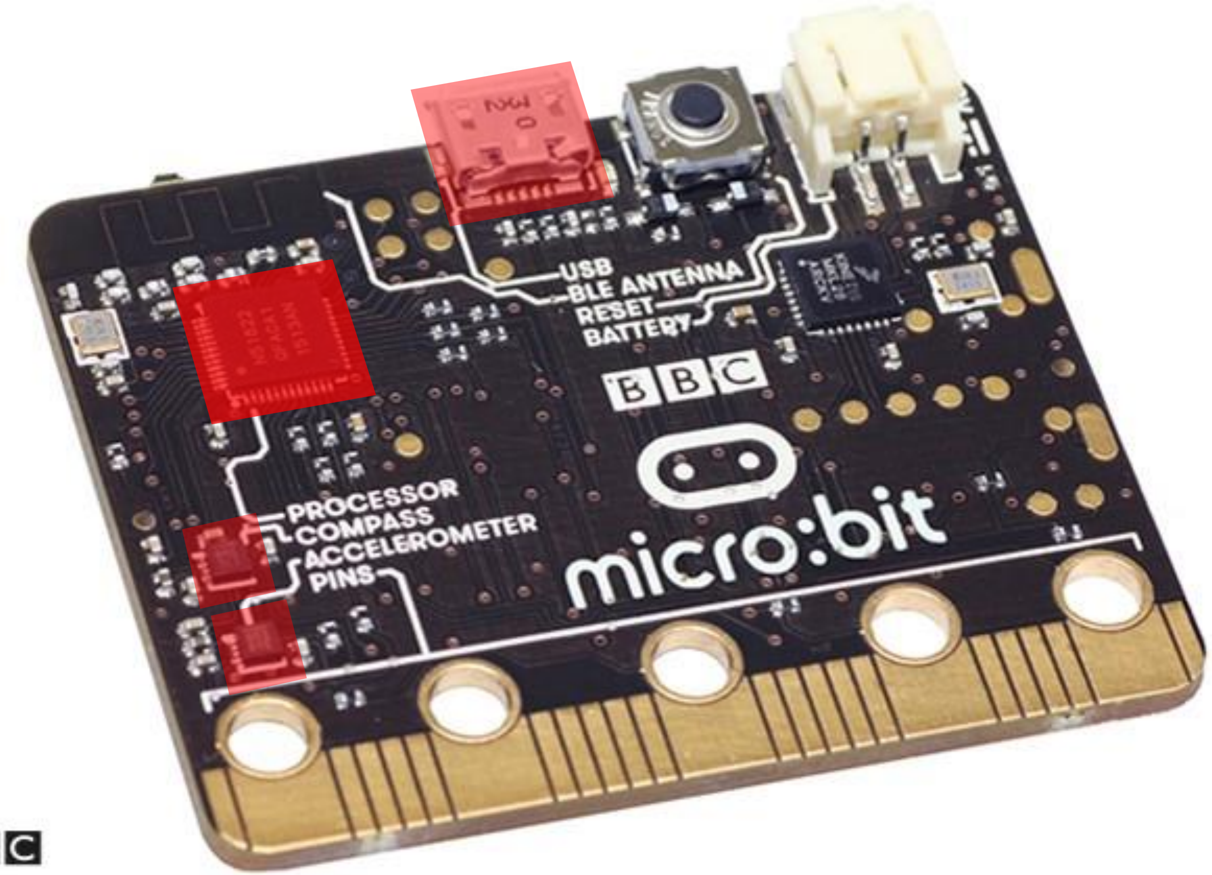




BBC micro:bit

- 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



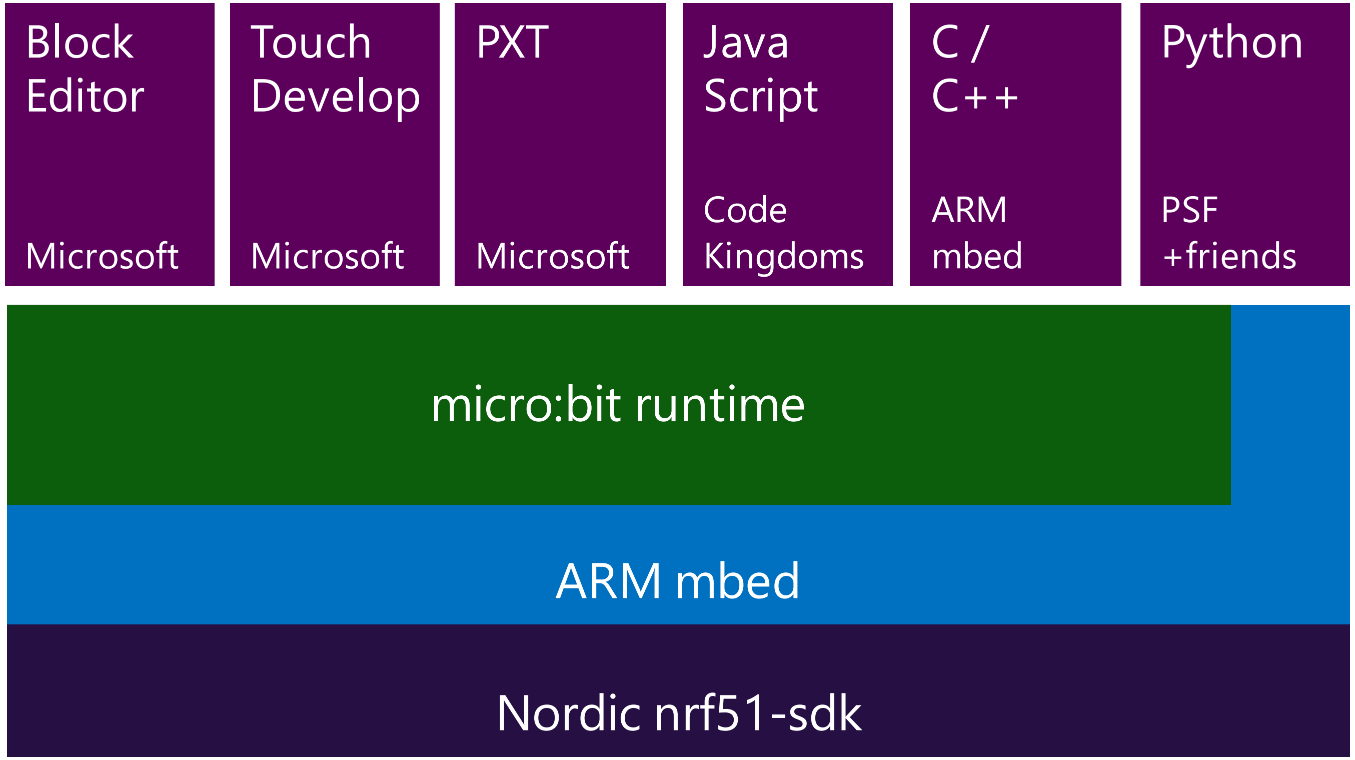
- 16MHz ARM Cortex M0
- 16KB RAM, 256K FLASH

- USB Storage/Serial/Debug

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

micro:bit runtime architecture

The micro:bit community encourages many languages…



Block Editor
Microsoft

Touch Develop
Microsoft

PXT
Microsoft

Java Script
Code Kingdoms

C / C++
ARM mbed

Python
PSF +friends

micro:bit runtime

ARM mbed

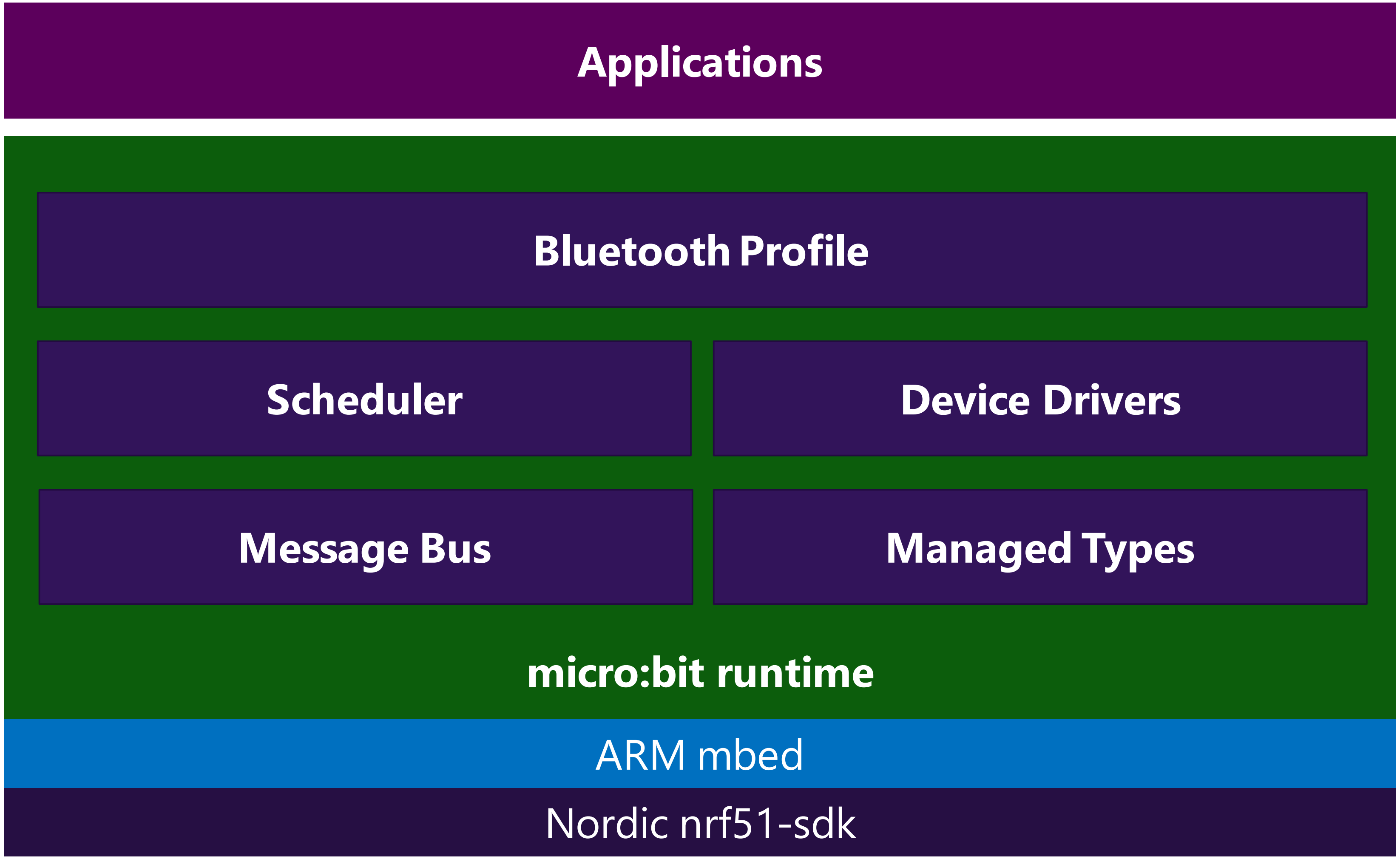
Nordic nrf51-sdk

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



Applications

Bluetooth Profile

Scheduler

Device Drivers

Message Bus

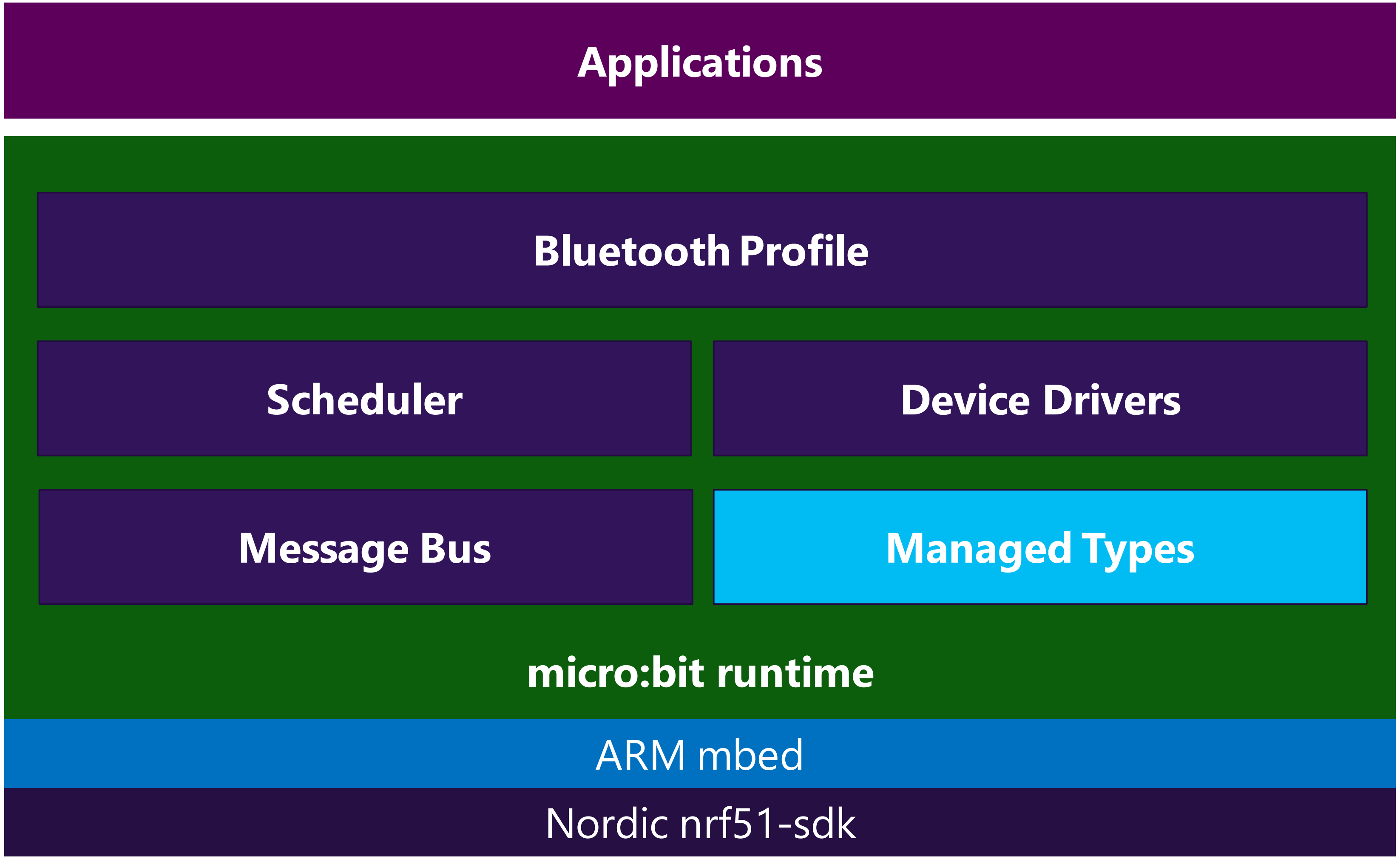
Managed Types

micro:bit runtime

ARM mbed

Nordic nrf51-sdk

micro:bit runtime architecture



Applications

Bluetooth Profile

Scheduler

Device Drivers

Message Bus

Managed Types

micro:bit runtime

ARM mbed

Nordic nrf51-sdk

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);
```

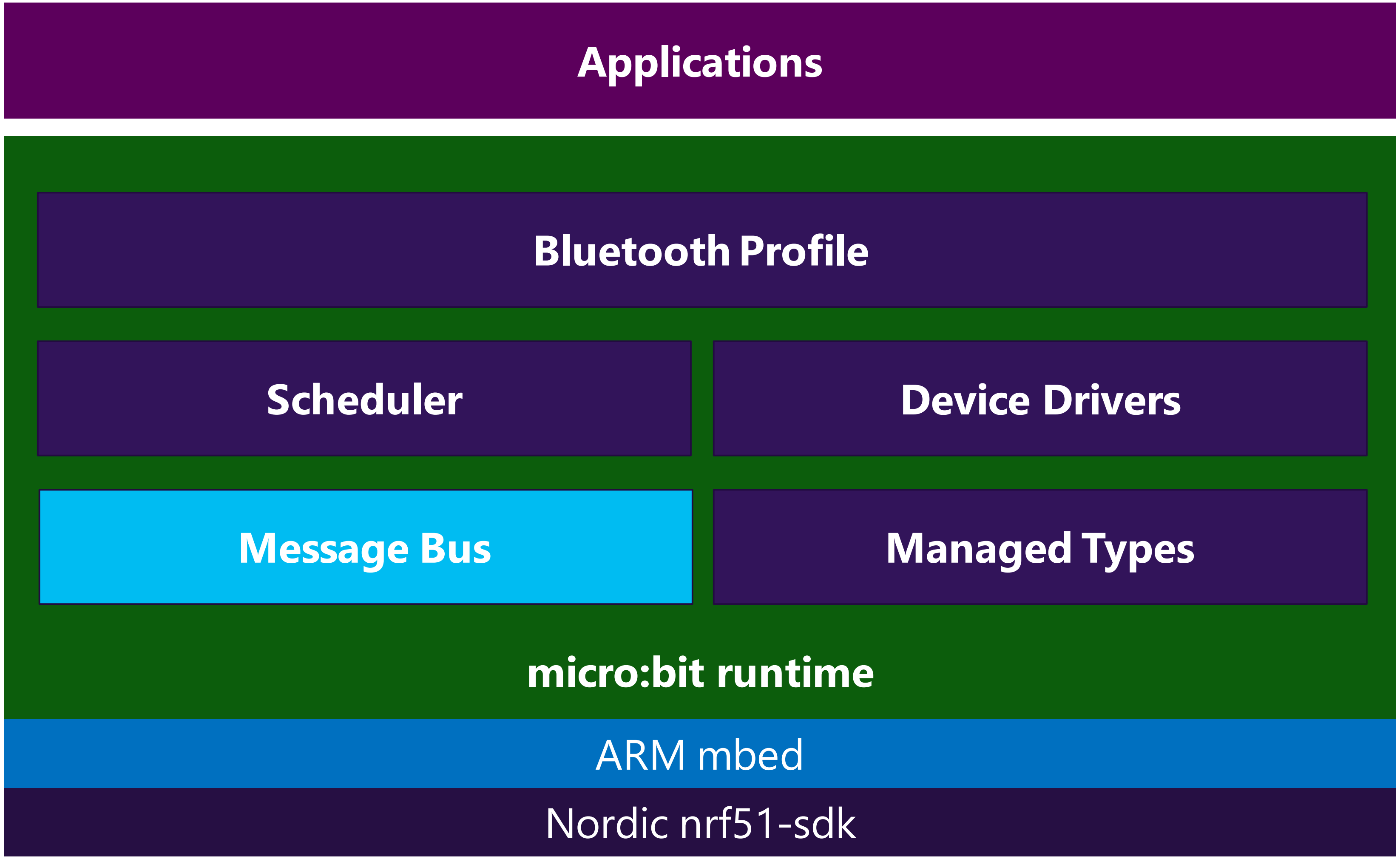
```
void
doSomething(ManagedString text)
{
    …
}
```

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



Applications

Bluetooth Profile

Scheduler

Device Drivers

Message Bus

Managed Types

micro:bit runtime

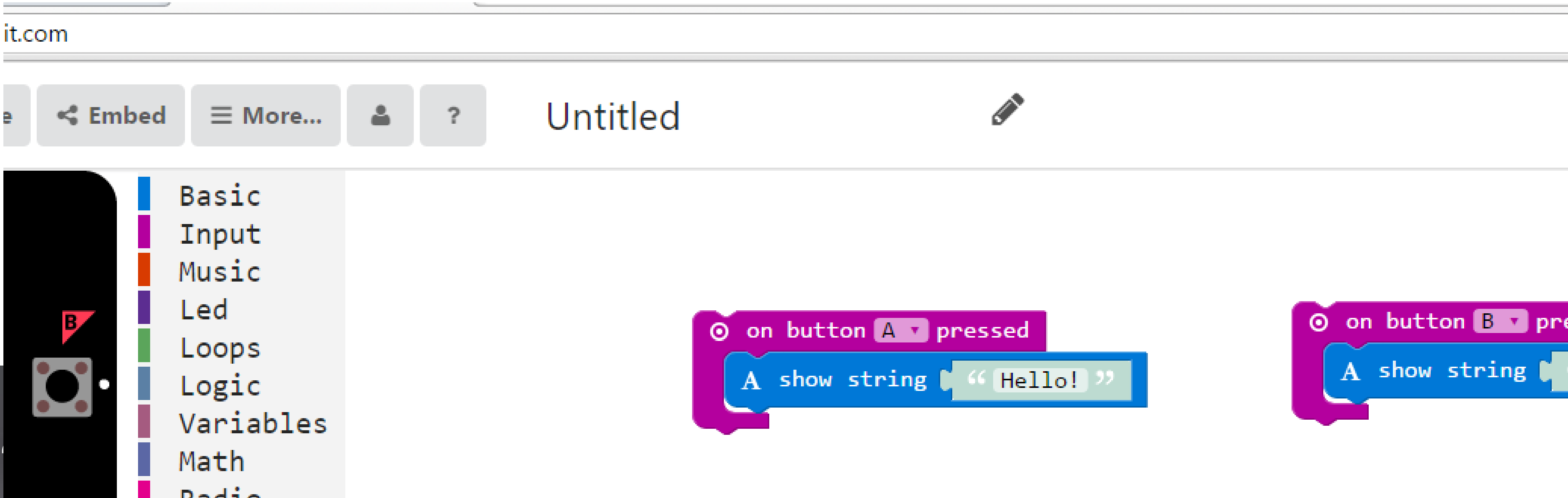
ARM mbed

Nordic nrf51-sdk

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.



```
on button A pressed
  show string "Hello!"

on button B pre
  show string
```

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…

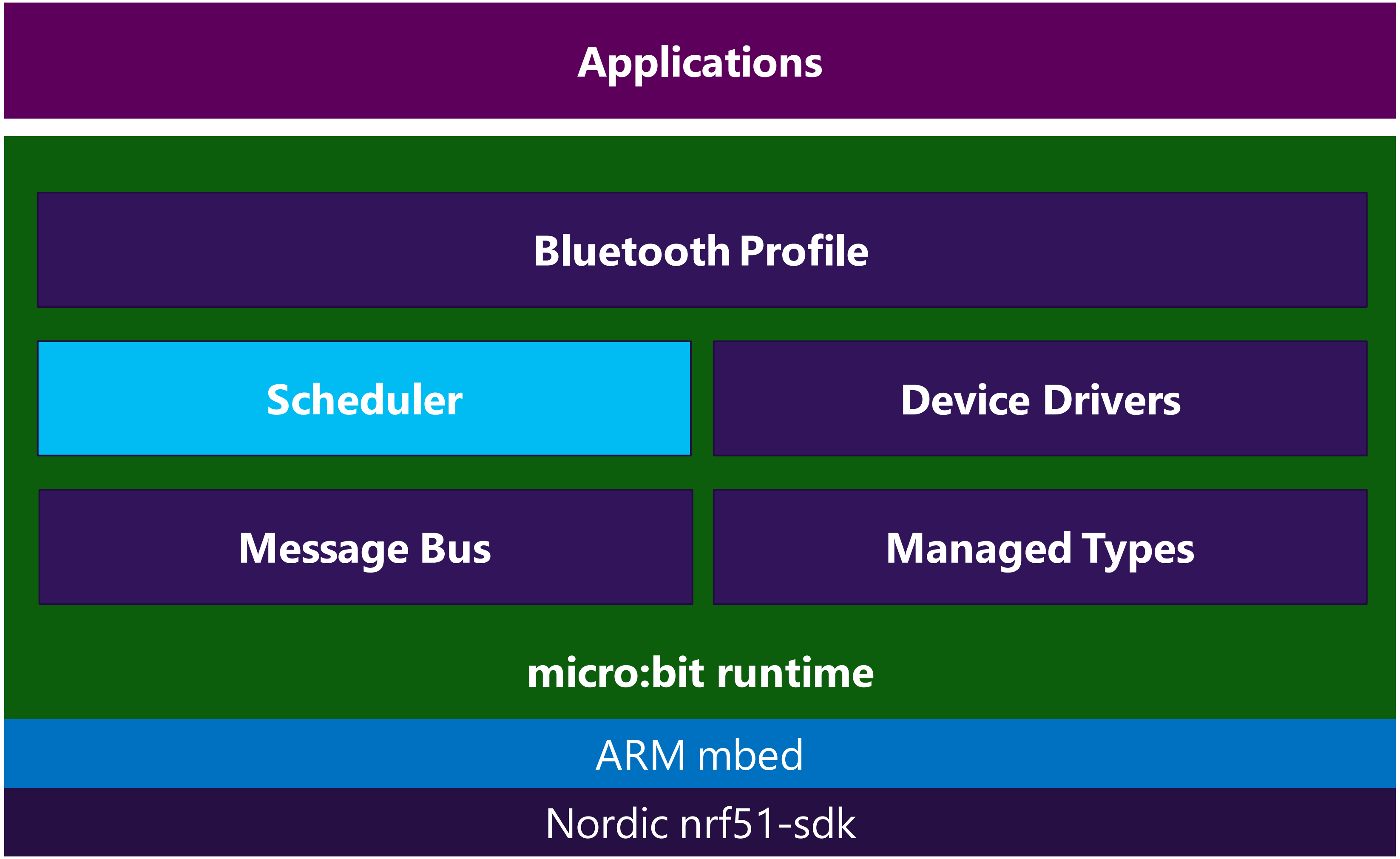
```
void onEvent(MicroBitEvent e)
{
    if (e.source == MICROBIT_ID_GESTURE) …
}

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



Applications

Bluetooth Profile

Scheduler

Device Drivers

Message Bus

Managed Types

micro:bit runtime

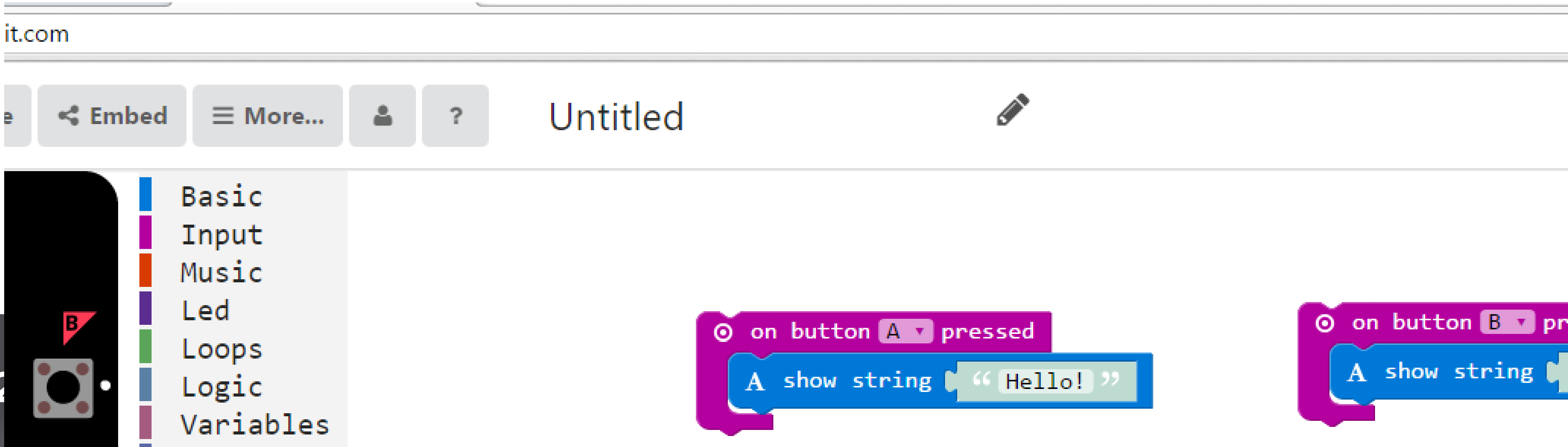
ARM mbed

Nordic nrf51-sdk

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…



```
on button A pressed
    show string "Hello!"
```

```
on button B pr
    show string
```

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.

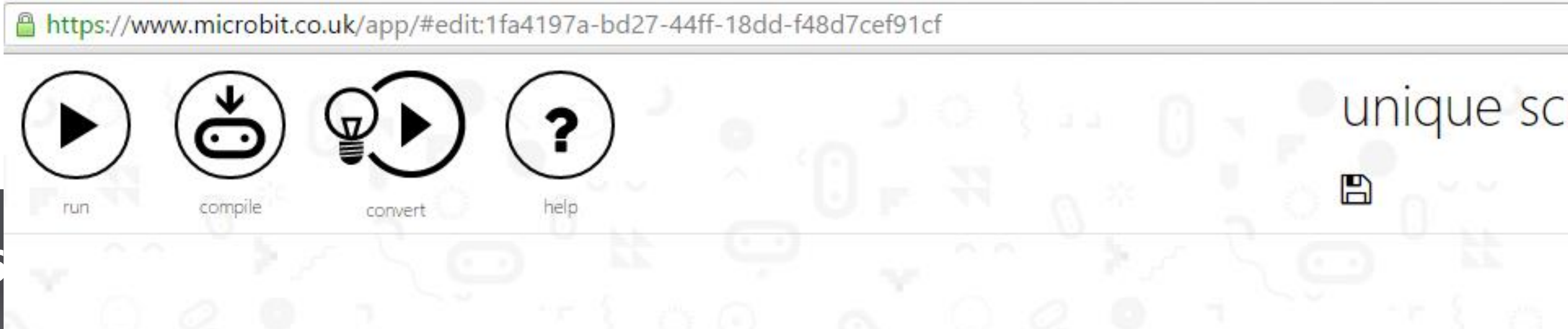
- Event handlers (by default) run in their own fiber*
- Effectively decoupling kids’ code from nasty interrupt context code.

- 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:

* Actually, this is a fib. We use a novel technique called fork-on-block to only create fibers if the code inside an event handler attempts to block the fiber... but that's for another day!

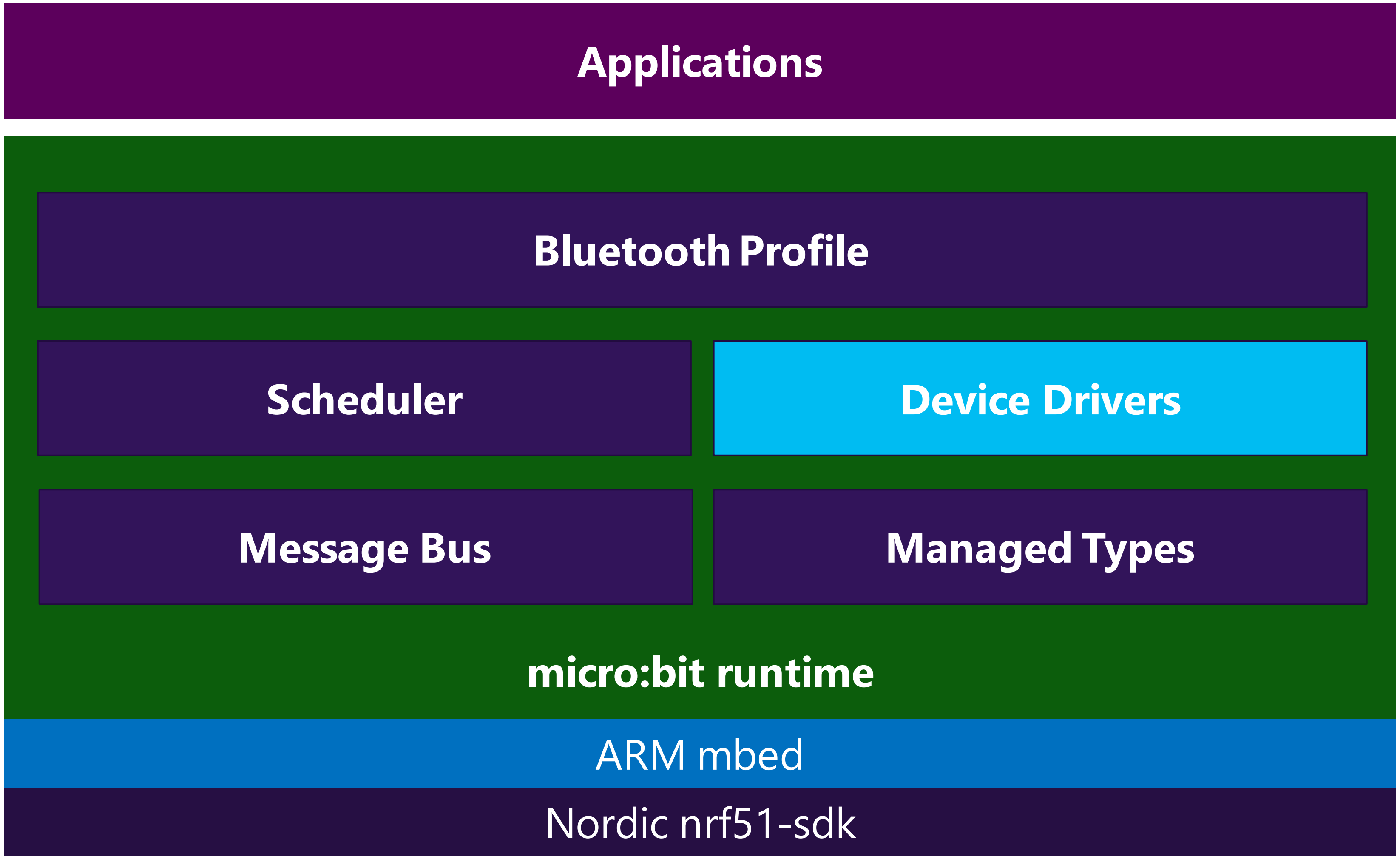
Fiber Scheduler: Providing Concurrent behaviour...

```
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);
```



https://www.microbit.co.uk/app/#edit:1fa4197a-bd27-44ff-18dd-f48d7cef91cf

micro:bit runtime architecture



Applications

Bluetooth Profile

Scheduler

Device Drivers

Message Bus

Managed Types

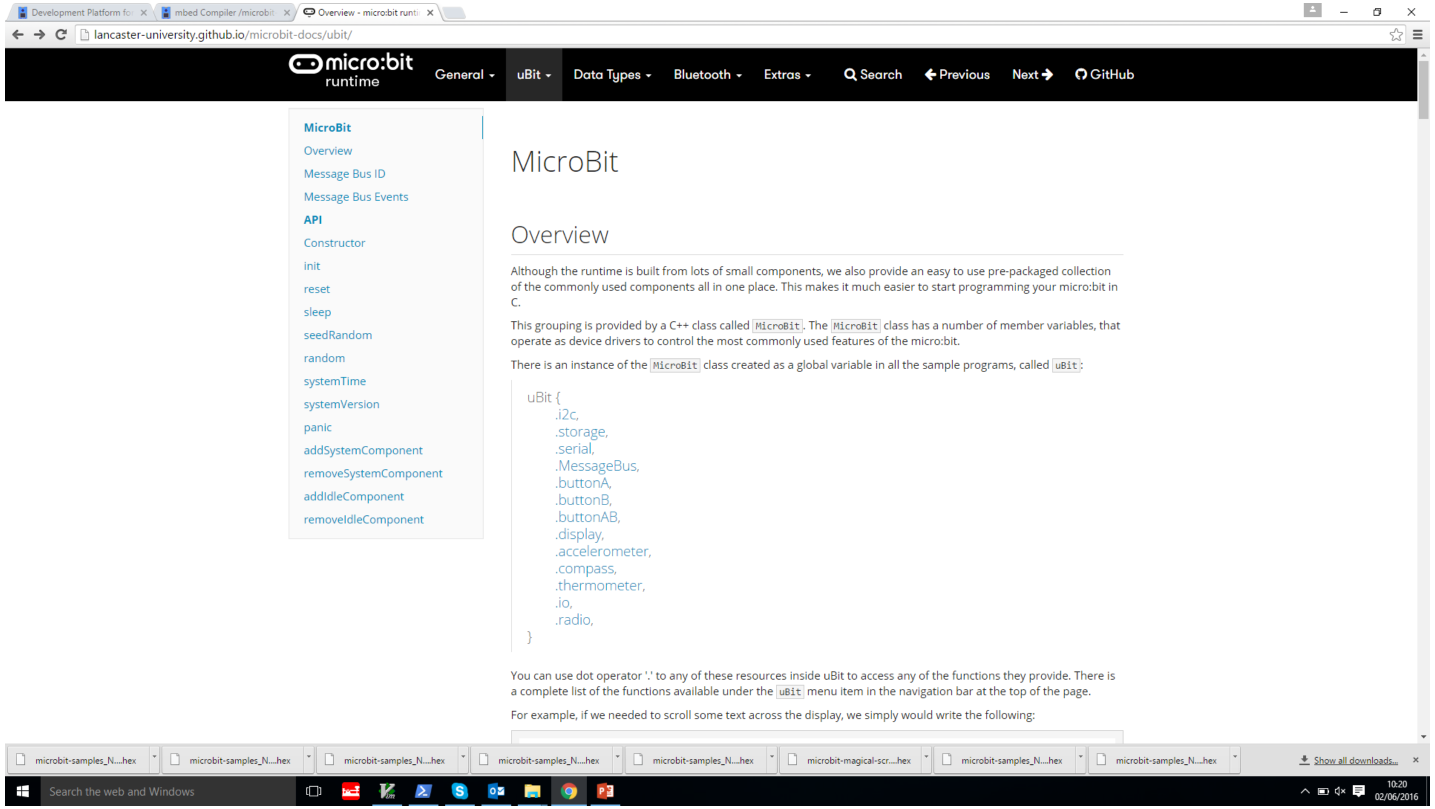
micro:bit runtime

ARM mbed

Nordic nrf51-sdk

Device Drivers

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



MicroBit

Overview

Although the runtime is built from lots of small components, we also provide an easy to use pre-packaged collection of the commonly used components all in one place. This makes it much easier to start programming your micro:bit in C.

This grouping is provided by a C++ class called `MicroBit`. The `MicroBit` class has a number of member variables, that operate as device drivers to control the most commonly used features of the micro:bit.

There is an instance of the `MicroBit` class created as a global variable in all the sample programs, called `uBit`:

```
uBit {
       .i2c,
       .storage,
       .serial,
       .MessageBus,
       .buttonA,
       .buttonB,
       .buttonAB,
       .display,
       .accelerometer,
       .compass,
       .thermometer,
       .io,
       .radio,
}
```

You can use dot operator '.' to any of these resources inside uBit to access any of the functions they provide. There is a complete list of the functions available under the `uBit` menu item in the navigation bar at the top of the page.

For example, if we needed to scroll some text across the display, we simply would write the following:

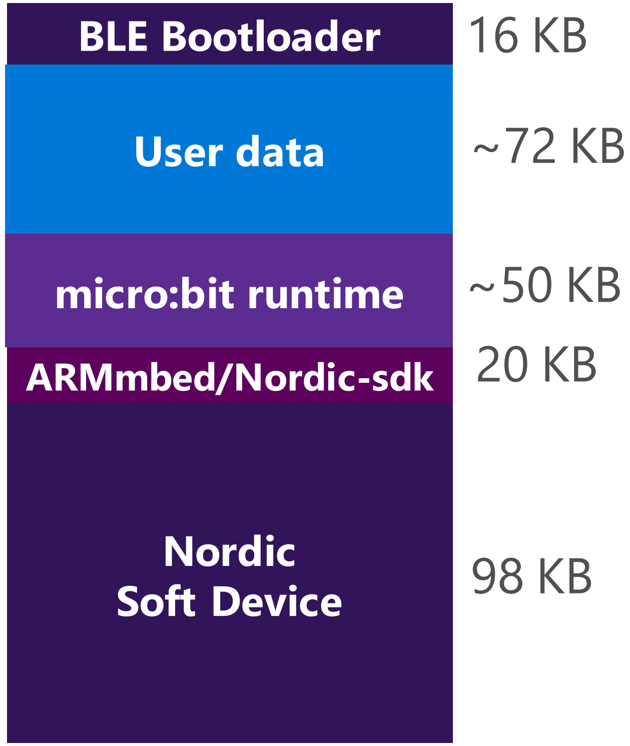
Device Drivers

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

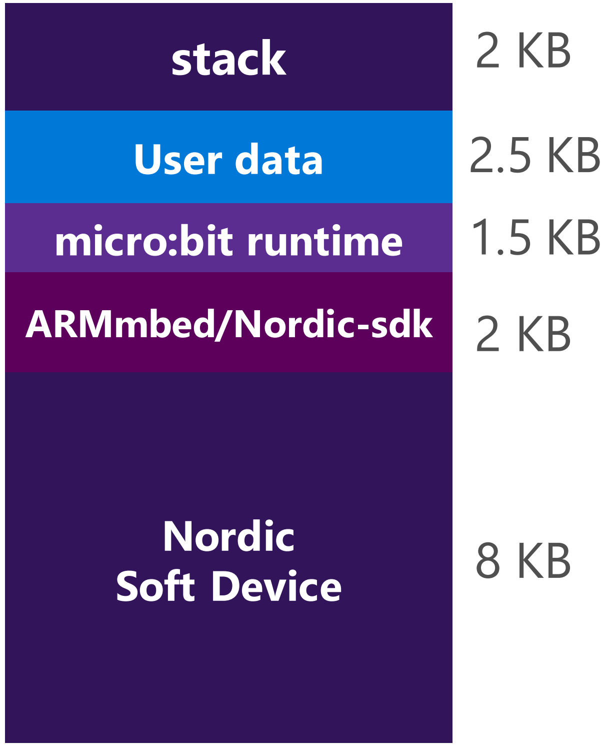
```
MicroBit uBit;

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…

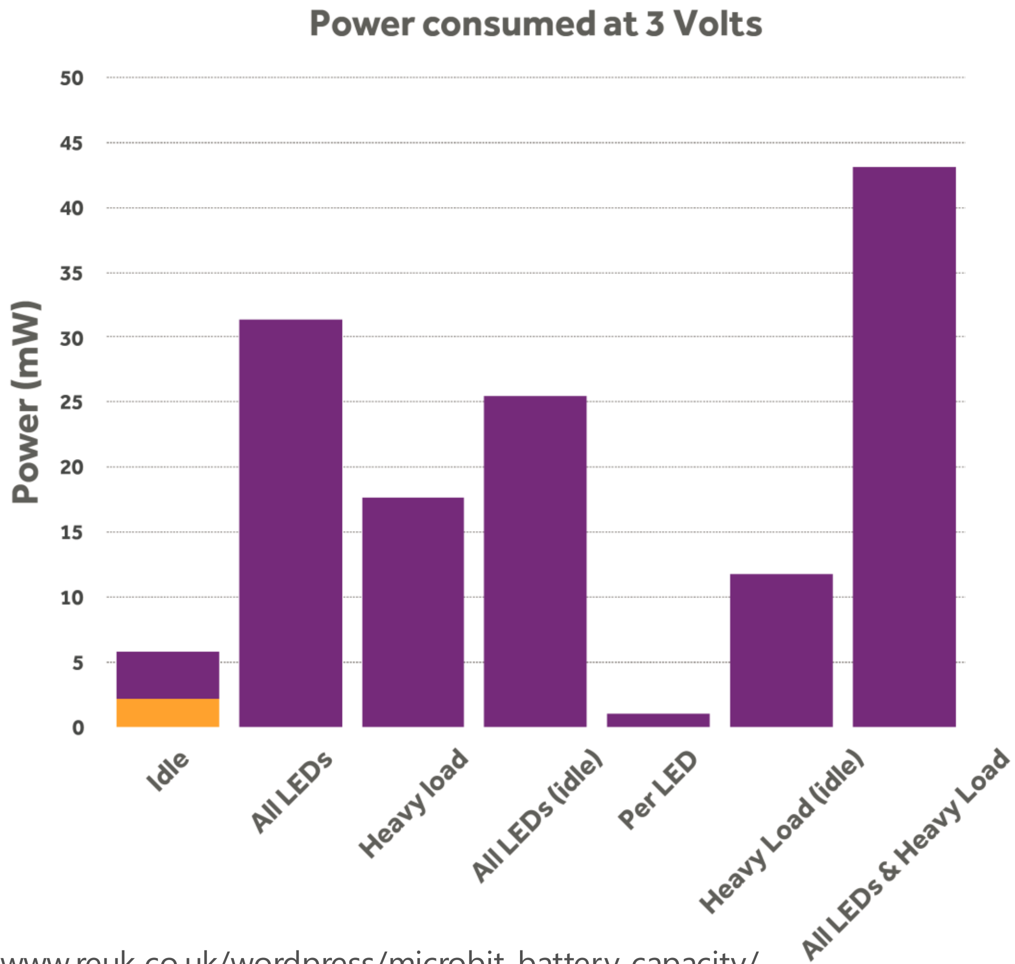


FLASH MEMORY



SRAM MEMORY

Power Efficiency



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

Pi 3 ~ 2000mW

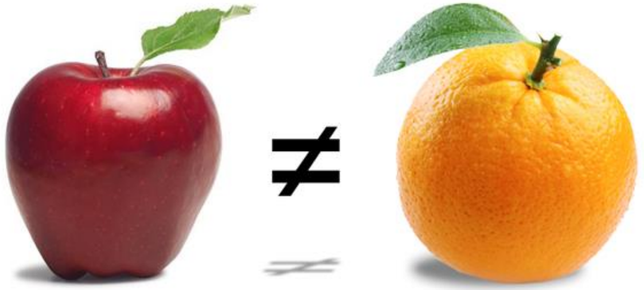
https://www.raspberrypi.org/help/faqs/

Pi Zero ~500mW

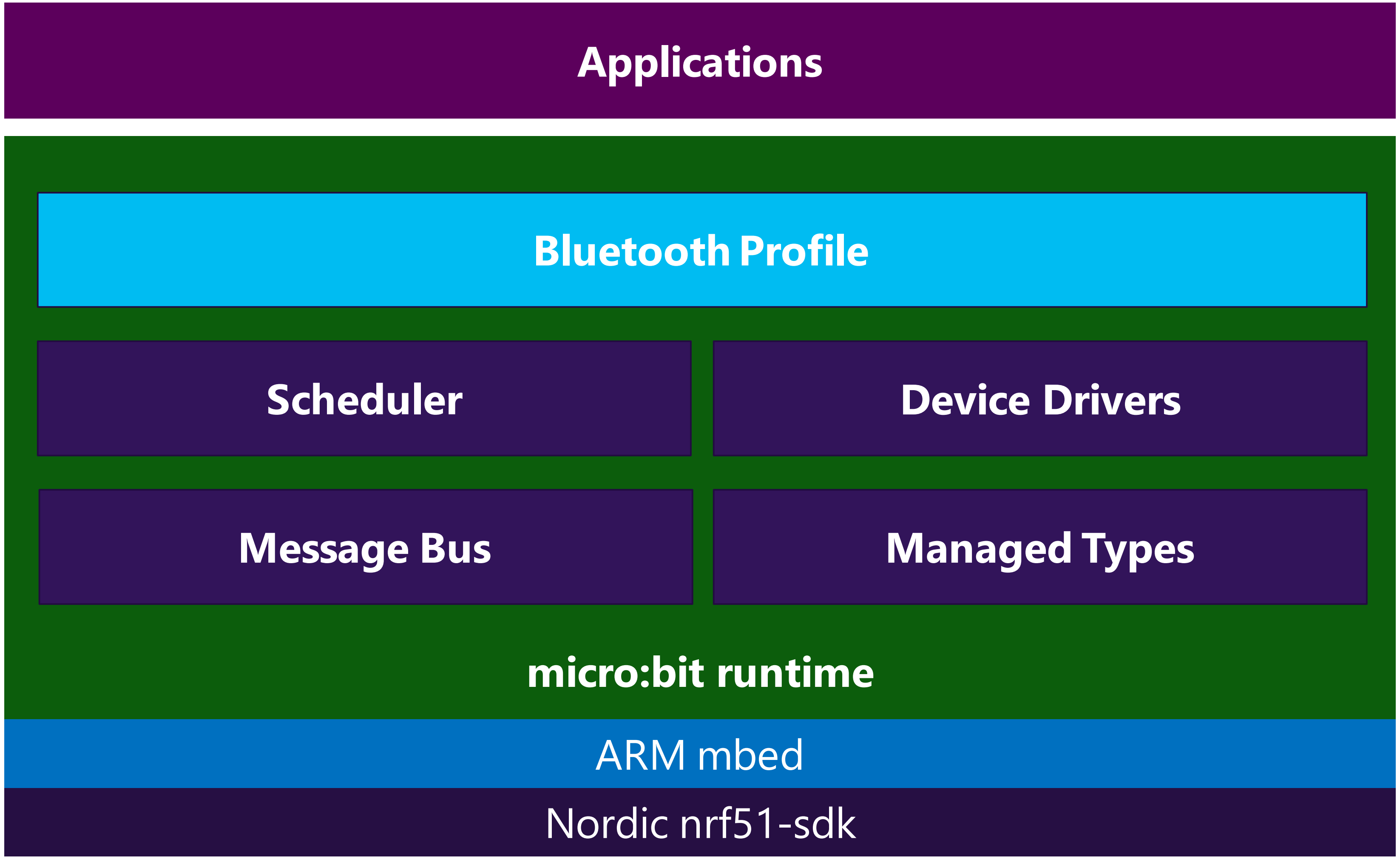
http://raspi.tv/2015/raspberry-pi-zero-power-measurements

Arduino Uno ~400mW

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



micro:bit runtime architecture



Applications

Bluetooth Profile

Scheduler

Device Drivers

Message Bus

Managed Types

micro:bit runtime

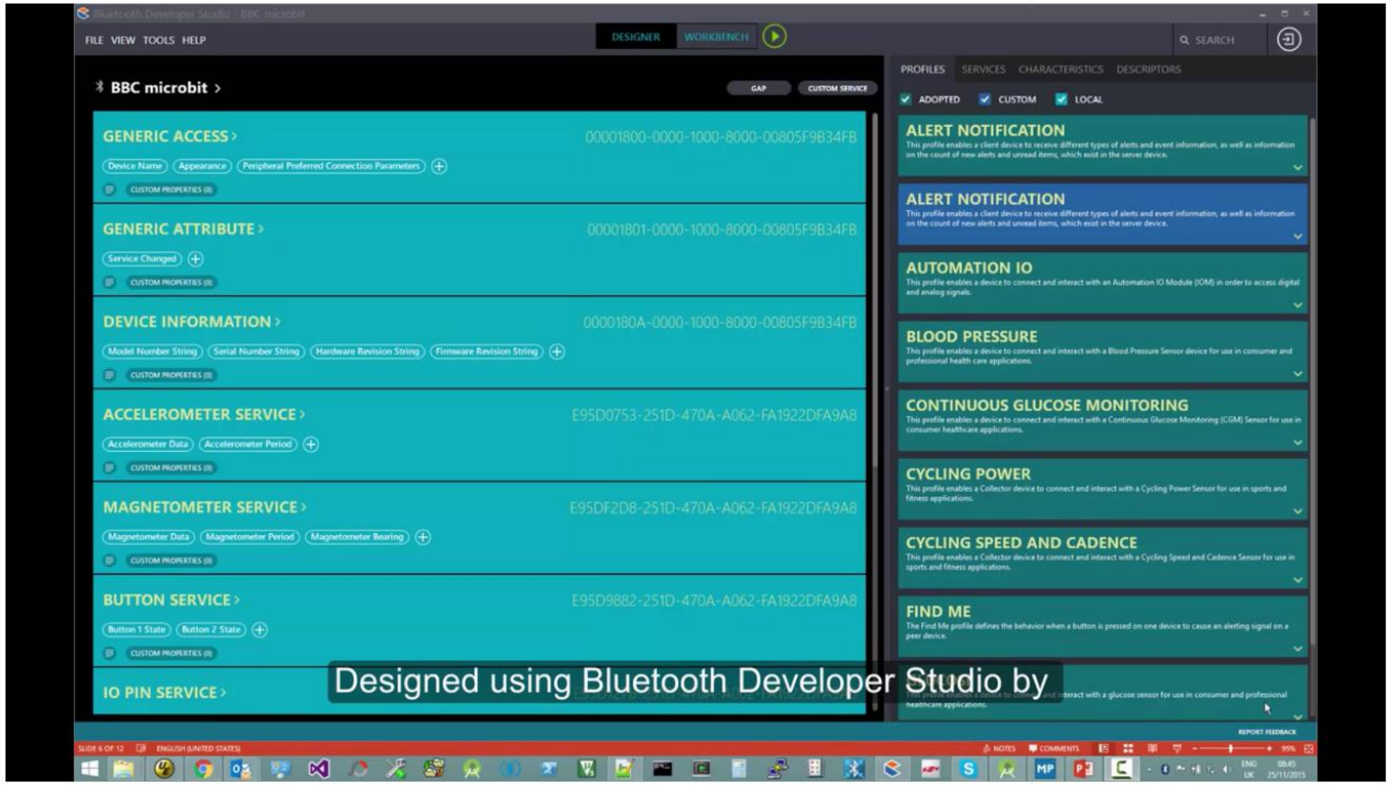
ARM mbed

Nordic nrf51-sdk

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)



GENERIC ACCESS > 00001800-0000-1000-8000-00805F9B34FB
Device Name | Appearance | Peripheral Preferred Connection Parameters
CUSTOM PROPERTIES (0)

GENERIC ATTRIBUTE > 00001801-0000-1000-8000-00805F9B34FB
Service Changed
CUSTOM PROPERTIES (0)

DEVICE INFORMATION > 0000180A-0000-1000-8000-00805F9B34FB
Model Number String | Serial Number String | Hardware Revision String | Firmware Revision String
CUSTOM PROPERTIES (0)

ACCELEROMETER SERVICE > E95D0753-251D-470A-A062-FA1922DFA9A8
Accelerometer Data | Accelerometer Period
CUSTOM PROPERTIES (0)

MAGNETOMETER SERVICE > E95DF2D8-251D-470A-A062-FA1922DFA9A8
Magnetometer Data | Magnetometer Period | Magnetometer Bearing
CUSTOM PROPERTIES (0)

BUTTON SERVICE > E95D9882-251D-470A-A062-FA1922DFA9A8
Button 1 State | Button 2 State
CUSTOM PROPERTIES (0)

IO PIN SERVICE >

ALERT NOTIFICATION
This profile enables a client device to receive different types of alerts and event information, as well as information on the count of new alerts and unread items, which exist in the server device.

ALERT NOTIFICATION
This profile enables a client device to receive different types of alerts and event information, as well as information on the count of new alerts and unread items, which exist in the server device.

AUTOMATION IO
This profile enables a device to connect and interact with an Automation IO Module (IOM) in order to access digital and analog signals.

BLOOD PRESSURE
This profile enables a device to connect and interact with a Blood Pressure Sensor device for use in consumer and professional health care applications.

CONTINUOUS GLUCOSE MONITORING
This profile enables a device to connect and interact with a Continuous Glucose Monitoring (CGM) Sensor for use in consumer healthcare applications.

CYCLING POWER
This profile enables a Collector device to connect and interact with a Cycling Power Sensor for use in sports and fitness applications.

CYCLING SPEED AND CADENCE
This profile enables a Collector device to connect and interact with a Cycling Speed and Cadence Sensor for use in sports and fitness applications.

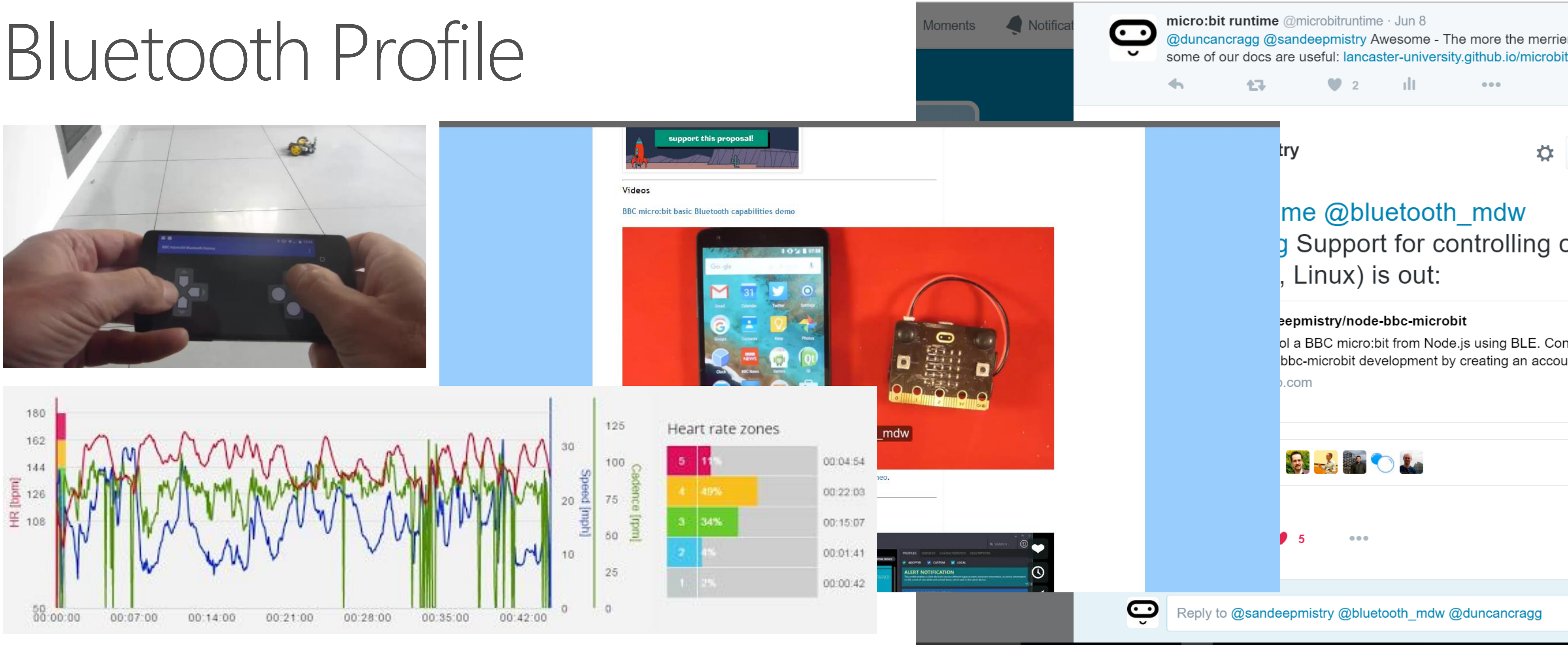
FIND ME
The Find Me profile defines the behavior when a button is pressed on one device to cause an alerting signal on a peer device.

…teract with a glucose sensor for use in consumer and professional healthcare applications.

Designed using Bluetooth Developer Studio by

microbit_profile_overview from Martin Woolley on Vimeo.

Bluetooth Profile



micro:bit runtime @microbitruntime · Jun 8

@duncancragg @sandeepmistry Awesome - The more the merrie

some of our docs are useful: lancaster-university.github.io/microbit

support this proposal!

Videos

BBC micro:bit basic Bluetooth capabilities demo

me @bluetooth_mdw

Support for controlling

, Linux) is out:

eepmistry/node-bbc-microbit

ol a BBC micro:bit from Node.js using BLE. Con

bbc-microbit development by creating an accou

.com

Reply to @sandeepmistry @bluetooth_mdw @duncancragg

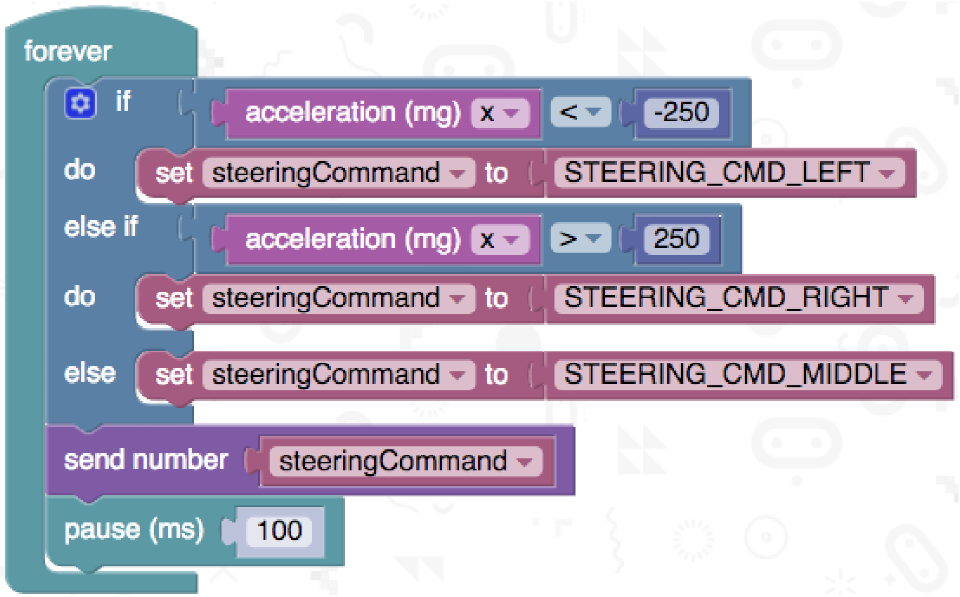
© Martin Woolley Bluetooth SIG

http://bluetooth-mdw.blogspot.co.uk/p/bbc-microbit.html

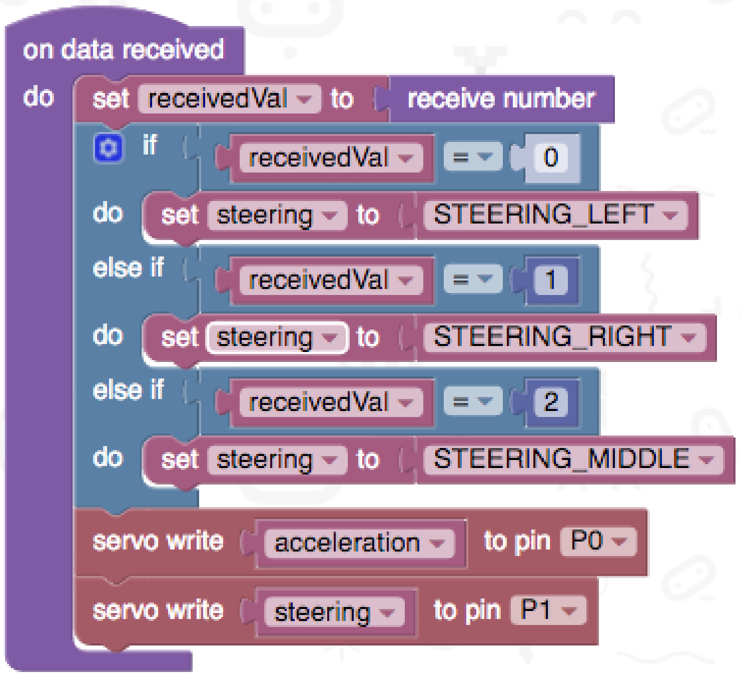
https://play.google.com/store/apps/details?id=com.bluetooth.mwoolley.microbitbledemo

MicroBitRadio

Simple, raw packet communications…

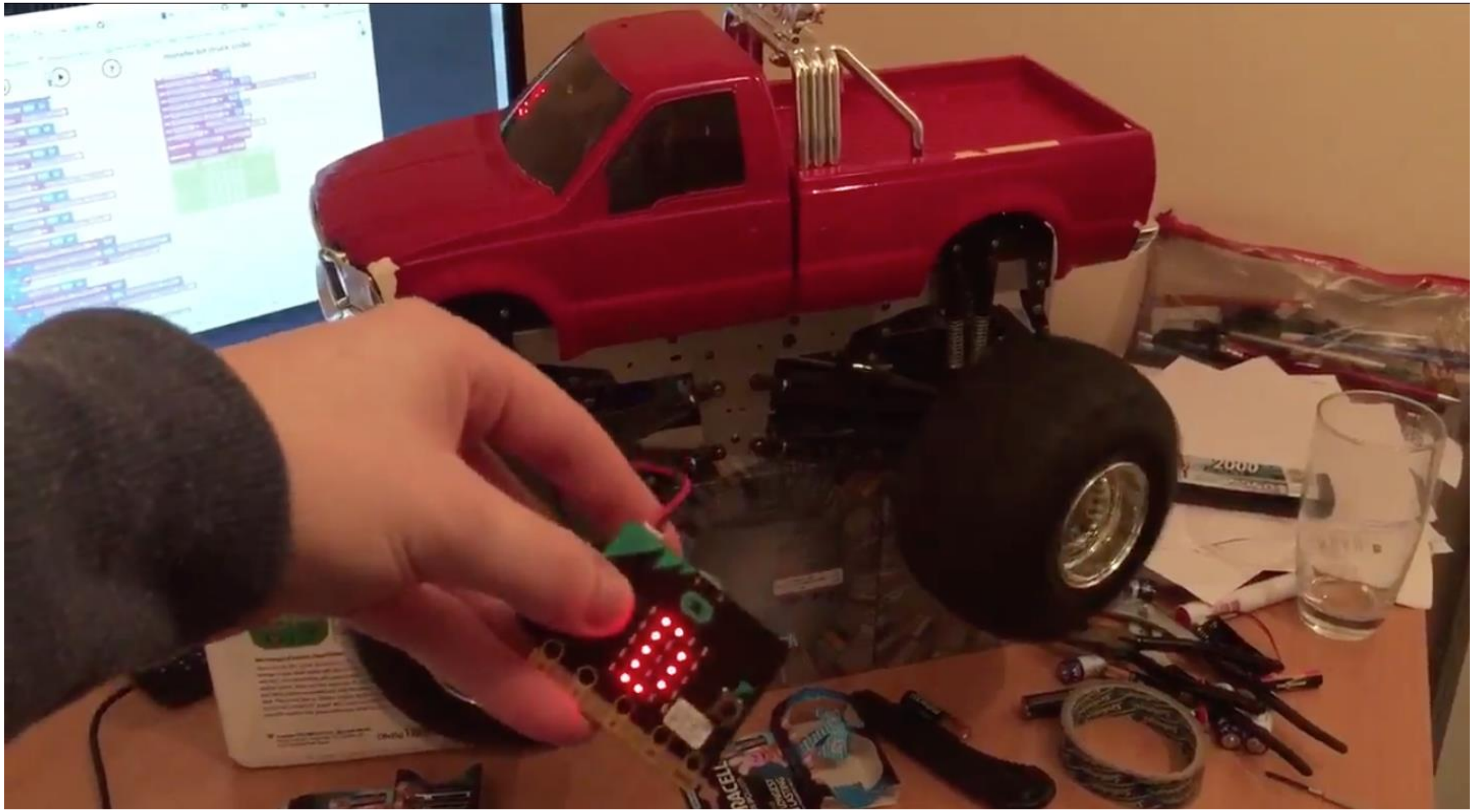


```
forever
    if  acceleration (mg) x  <  -250
    do  set steeringCommand to STEERING_CMD_LEFT
    else if  acceleration (mg) x  >  250
    do  set steeringCommand to STEERING_CMD_RIGHT
    else  set steeringCommand to STEERING_CMD_MIDDLE
    send number  steeringCommand
    pause (ms)  100
```



```
on data received
do  set receivedVal to receive number
    if  receivedVal  =  0
    do  set steering to STEERING_LEFT
    else if  receivedVal  =  1
    do  set steering to STEERING_RIGHT
    else if  receivedVal  =  2
    do  set steering to STEERING_MIDDLE
    servo write  acceleration  to pin  P0
    servo write  steering  to pin  P1
```

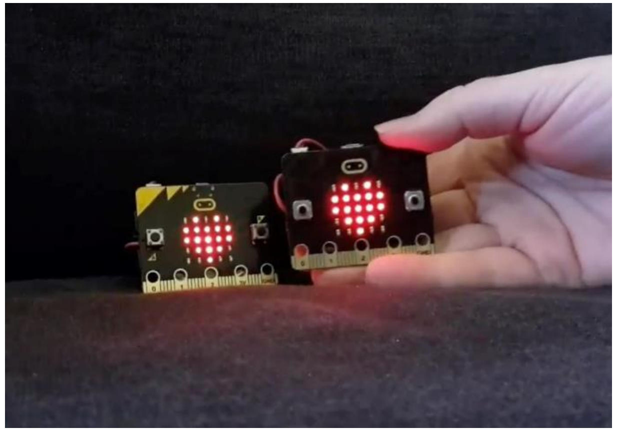
MicroBitRadio



@BBCMIDigital

RETWEETS 21

LIKES 40



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

BBC

micro:bit

Wanna go play?

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