BBC Microbit

Installation

Storage

Communications MB ←→ PC

Internet Access – remote control

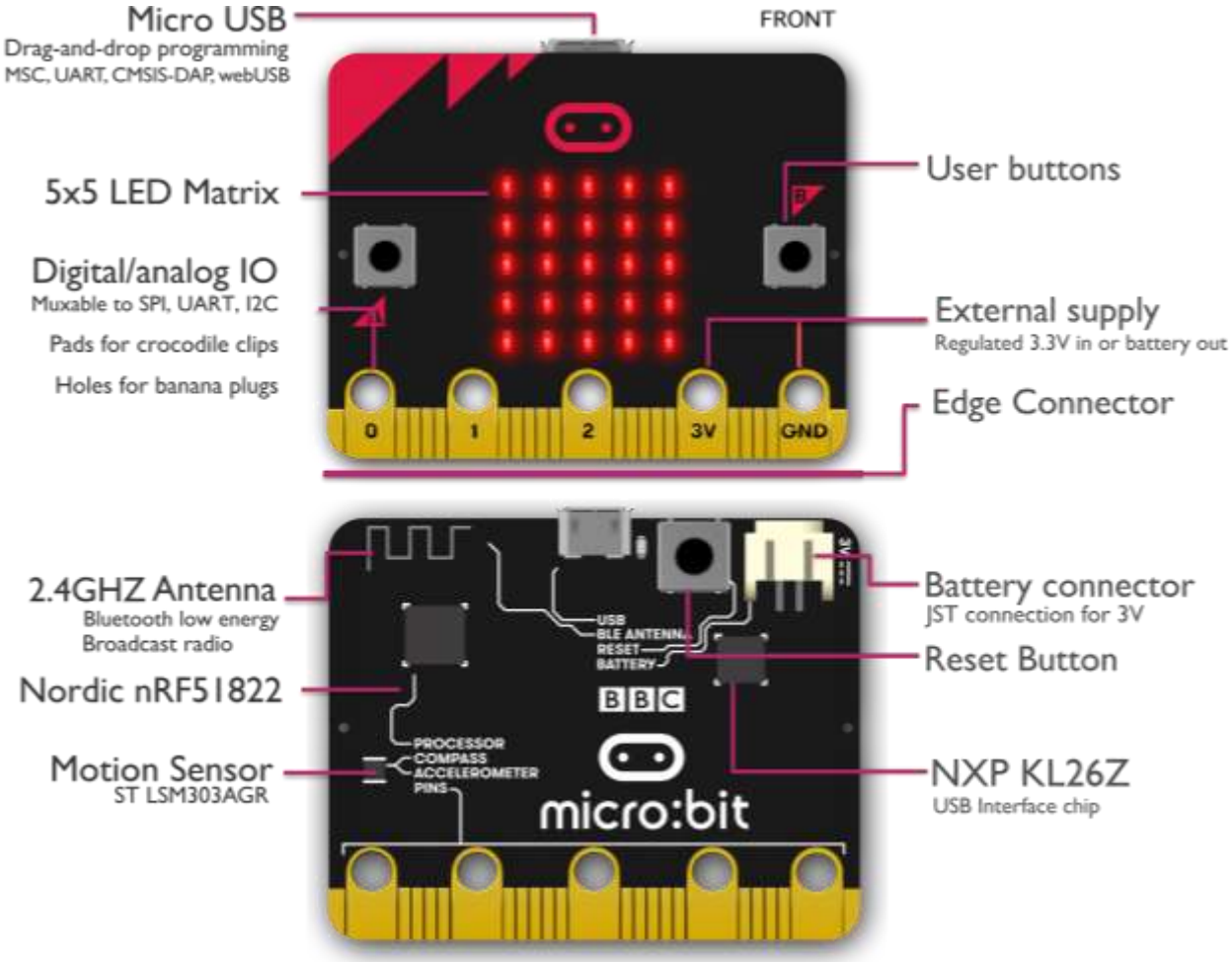
BBC Microbit

- **Much lower power than the Raspberry Pi or Arduino, , but look at all the sensors**
- **Biggest advantage:**
  - **Access to sensors, both standalone, and via PC**

Sensor suchs as :

- **Radio & Bluetooth antenna**
- **Processor & temperature sensor**
- **Compass**
- **Accelerometer and Gyroscope**
- **Input/Output pins**
- **Single and 25x LED for display and light sensing**
- **Reset button + 2 user Buttons**
- **Micro USB socket**
- **Battery socket**
- **USB interface chip**

BBC Microbit



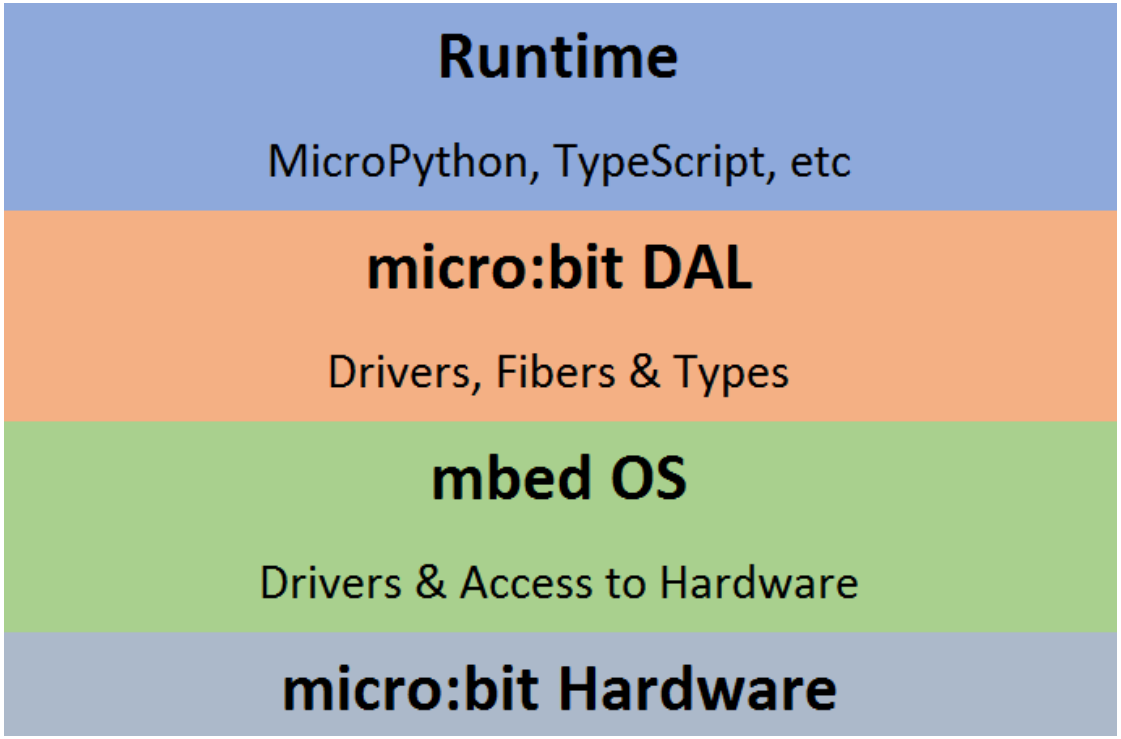
BBC Microbit – more detail

- When you connect the Micro:bit (MB) via USB, you will see an extra drive with some files on it.
  - If you hold down the reset button while connecting the USB, you will see a different set of files – including device.txt which contains the version numbers of all the software installed.

```
# DAPLink Firmware - see https://mbed.com/daplink
Unique ID: 0000000051864e45000a10070000004a0000000097969972
HIC ID: 97969901
Auto Reset: 0
Automation allowed: 1
Overflow detection: 0
Daplink Mode: Bootloader
Bootloader Version: 0243
Interface Version: 0249
Git SHA: b403a07e3696cee1e116d44cbdd64446e056ce38
Local Mods: 0
USB Interfaces: MSD
Bootloader CRC: 0x32eb3cfd
Interface CRC: 0xcdb7b2a3
Remount count: 0
```

Microbit Memory Map

- What is actually inside the ,e,pry pf the MB?



Microbit Memory Map

- **<u>Runtime</u>**
  - MicroPython, TypeScript and other languages

- **<u>Microbit Device Abstraction Layer (DAL)</u>**

Core – High-level components,

Types – Helper types such as ManagedString, Image, Event and PacketBuffer

Drivers – For control of a specific hardware component, such as Accelerometer, Button, Compass, Display, Flash, IO, Serial and Pin

Bluetooth – All the code for the Bluetooth Low Energy (BLE) stack that is shipped with the micro:bit

Microbit Memory Map

- **”mbed” Operating System (https://mbed.com)**
  - The software at the bottom of the stack is making use of the ARM mbed OS which is:
    - An open-source embedded operating system designed for the “things” in the Internet of Things (IoT).
    - mbed OS includes the features you need to develop a connected product using an ARM Cortex-M microcontroller.
    - mbed OS provides a platform that includes:
      - Security foundations.
      - Cloud management services.
      - Drivers for sensors, I/O devices and connectivity.
  - mbed OS is modular, configurable software that you can customize to your device and to reduce memory requirements by excluding unused software.

Microbit Memory Map

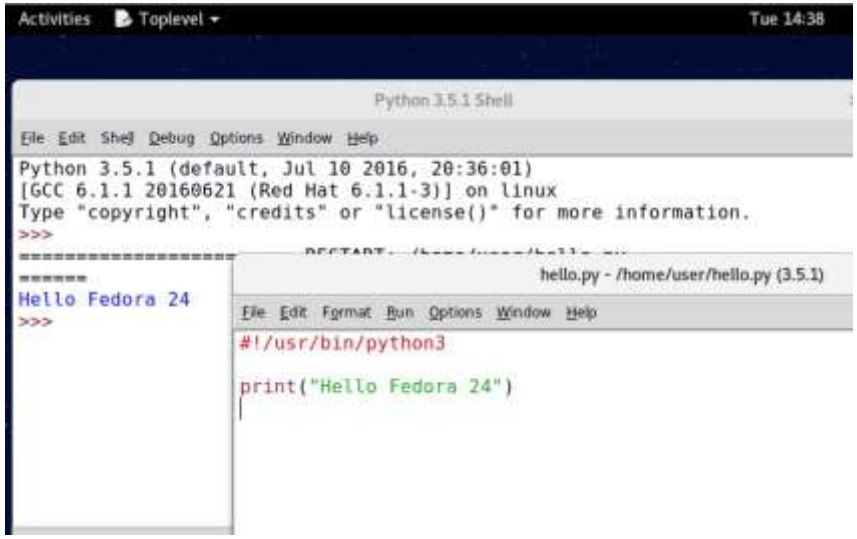
- **<u>”mbed”. Operating System</u>**
  - Lower level Device Drivers
  - Controlled Access
  - Security

- **<u>Hardware</u>**
  - **<u>3D Accelerometer, Gyro</u>**
  - **<u>Compass</u>**
  - **<u>25 LED’s</u>**
  - Pins

- More detail: https://mattwarren.org/2017/11/28/Exploring-the-BBC-microbit-Software-Stack/

Installation

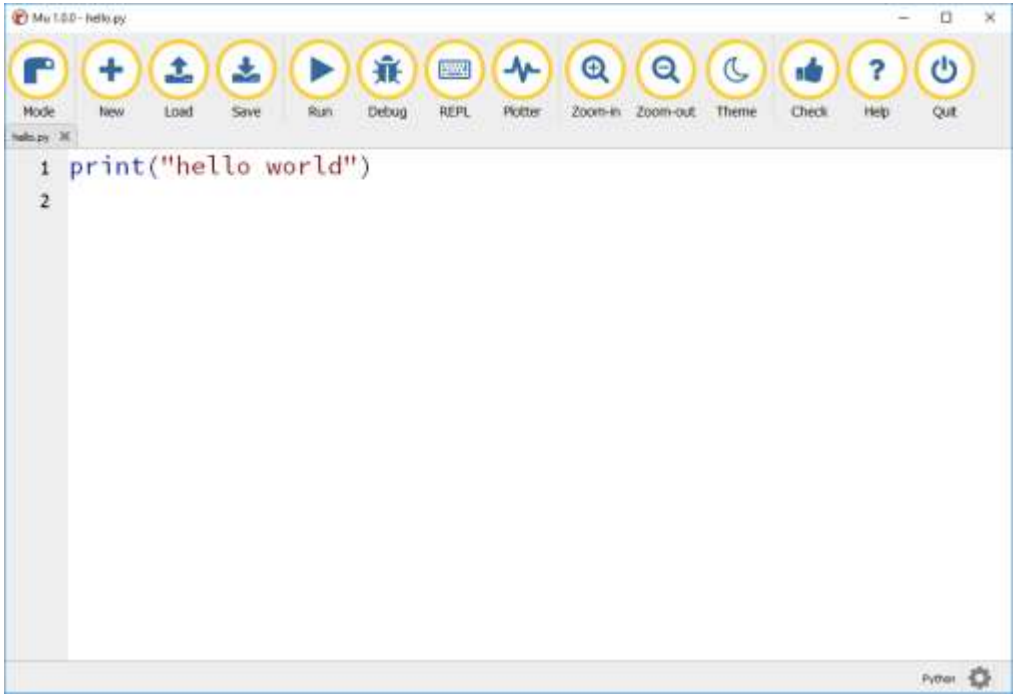
- Install Python from https://python.org
  - Latest version
  - Has its own IDLE editor
  - Also possible to link Visual Studio and Eclipse to thisversion
  - For Windows, be sure to select “include Python in path” in the installation proess



Installation

- Install Mu from
  https://codewith.mu/en/howto/1.0/install_with_python
  - Specific for BBC Microbit, but can also be used for PC Python

Mu includes Python, so you can skjip the next step if you select this editor



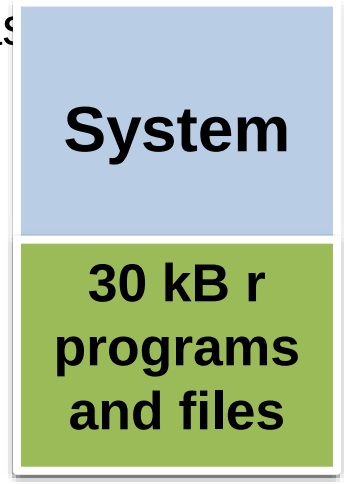
Installation - pip

- You will also need several packages
- You nstall new packages using “pip” or “pip3”
- Pip comes with Python, but is run at the command line
  - See youyr tutor for how to do this.

- Use pip to sinatll modules. In particular
  - pip install pyserial for access to serial ports
  - pip install microfs access the microbit file system
    - To find out where this was installed, type type within Python
      >>> import microfs
      >>> microfs.__file__
    - Close to the folder that is returned, you will find a program called **<u>ufs.exe</u>**,.

Storage

- The Micro:bit (MB) has an internal file system that allows you to run it in a variety of different ways.
  - You **<u>can</u>** ***flash*** the device using an editor such as
    Benefits
    - More complex programs are possible
    - More space available
    - More flexibility to control wake-up state
    - No information is kept between resets,
      - So start with clean slate.

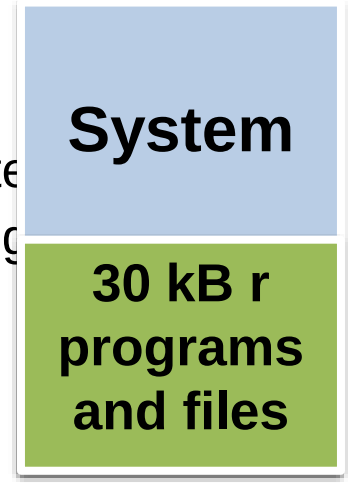


- This is how you have been using it in Grok
  - But there is another way

Storage

- The Micro:bit (MB) has an internal file system that allows you to run it in a variety of different ways.

  – You can copy a file called main.py onto it.
  <u>Benefits</u>:
    - This file will survive resets and power off.
    - You can also storage extra files on the file syste
    - Hardware such as timers can retain their setting
    - Actions can be performed remotely



  <u>Issues</u>
    - This REPLACES the file system every time you do it.
    - No information is kept between resets,
      – So start with clean slate.

Details: https://microbit-micropython.readthedocs.io/en/v1.0.1/tutorials/storage.html

Using ufs and microfs to locate files on MB

- On the command ine, you can type one of the following:
  - ufs ls List files
    main.py data.csv time.txt

    f = microfs.ls()
    print(f)
    [“main.py“,“data.csv“ “time.txt”]

  - ufs get main.py get file from MB to PC
    f = microfs.get(”main.py”)

  - ufs put main.py put file in current folder from PC
    microfs.put(“main.py”)

  - ufs rm data.csv delete file from MB file system
    microfs.rm(“data.csv”)

  – In addition microfs has the function
    com = microfs.get_serial()
  – This find the right serial port that the microbit is connected to and opens it. It returns a serial object which you can use to read/write from the port.

Using micropc to control the MB

- Get module microbitpc.py from Canvas (Week 10/Lab10)
- microbitpc contains python commands to control the microbit as follows

```
>>>  frommicrobitpc import *
>>> com = openMB()          Get the COM port tfor the MB
                            com is a serial object
>>> stopMB(com)             Stop the running program
>>> restart(com)            Restart  “main.py”
>>> resetMB(com)            Reset the MB, restarts “main.py”
>>> ln = readMB(cin, cb)    Read cb bytes from com to ln
>>> nc = wrteMB(com, ln)    Write ln to serial, nc bytes written
>>> closeMB(com)            Close serial port
```