The BBC micro:bit and amateur radio

The micro:bit is a tiny low cost single board computer that the BBC has developed to inspire children to learn more about programming [1, 2, 3, 4]. It includes a high performance microcontroller (PIC), has USB and Bluetooth functionality, as well as having a built-in magnetometer and accelerometer.

This article is intended to introduce how the BBC micro:bit will be useful in developing all sorts of interesting projects, especially for young people. I hope it will also be useful for amateur radio club activities. This article shows some simple amateur radio projects. It's not about examples of great coding – I have opted a simple 'get you started approach' – you can, of course, develop the code as much as you wish. **Photo 1** shows a general view of the BBC micro:bit and **Figure 1** identifies the major items.

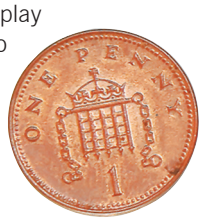
The BBC micro

In the 1980s the BBC introduced a microcomputer to teach basic programming skills through their TV series *The Computer Programme*. Along with the ZX81 and various other iconic computers many were inspired to take up a career in programming. Unfortunately, the decades that followed were a poor time for learning programming, with many information and communication technology (ICT) courses simply focusing on how to handle a spreadsheet and write a Word document.

In 2016 the BBC launched the BBC micro:bit to stimulate another explosion of programming enthusiasm and skill in children [1]. The device is now available to buy and so has opened up the devices to a wider audience (eg through the pi-store [2] or on eBay for around £12-14).

The micro:bit is a small PCB that has an edge connector allowing access to many of the microcontroller pins. Three of the chip ports (port 0, 1 and 2), the 0V and +3V are accessible via 4mm holes on the PCB that fit standard banana plugs. A multiway edge connector block can be used to make it easier to connect to the other pins.

There are three switches: switch A and B on the front and a reset switch on the back. The front of the PCB also includes a 5 x 5 pixel LED display that can be used to draw images or scroll messages. The LEDs can not only be switched on and off but can also be adjusted in brightness.



Elektor magazine has developed an edge connector PCB that also includes thermometer and barometer devices, accessible via **[5]**. With their software it converts the micro:bit into a tiny weather station, which might be useful for /P or for contest events. If you don't want to use the weather station parts it can still form a useful edge connector to get at the ports of the micro:bit, see **Photo 2**.

BBC micro:bit specifications

- PCB size 43 x 52mm
- Nordic nRF51822 16MHz 32-bit ARM Cortex-M0 microcontroller, 256kB flash memory, 16kB static RAM, 2.4GHz Bluetooth Low Energy wireless networking
- NXP/Freescale KL26Z 48MHz ARM Cortex-M0+ core microcontroller
- 3-axis accelerometer sensor
- 3-axis magnetometer sensor (to act as a compass / metal detector)
- microUSB connector
- battery connector
- display consisting of 25 LEDs in a 5×5 array
- I/O includes five x 4mm banana plug ring connectors and 23-pin edge connector, three PWM outputs, 17 GPIO pins, six analogue inputs, serial I/O, SPI, and I²C.

The microcontroller on the micro:bit only has only a relatively small amount of memory space to play with compared to more advanced ICs, however there is still a lot of scope for useful and interesting projects.

Power requirement

The micro:bit can be powered from the 5V from a USB connection (or other USB type power supply) or by 3V (2 x AAA) via the built-in connector. The microcontroller

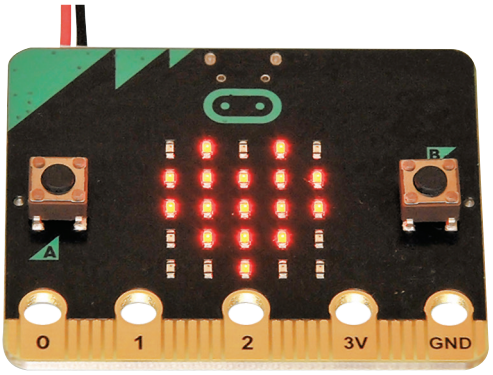


PHOTO 1: The BBC micro:bit with a penny for scale.

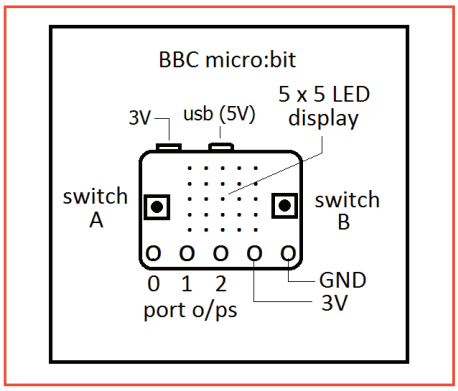


FIGURE 1: Diagrammatic view of the BBC micro:bit.

actually runs on 3V so there is an internal regulator / step-down IC. When the micro:bit is running, its LED display takes very little current, and in many circuits takes just a few mA so the power requirements are perfect for battery powered projects.

Free programming online

The BBC Micro:bit website allows you to program the board in a variety of different languages and styles including Javascript, Microsoft Block Editor, Microsoft Touch develop, and Python (called micro Python). It's all free online [1, 3, 4, 5, 6].

In principle all you have to do to program the micro:bit is to write your program online (or on your computer, see *Mu* editor described later) using one of the options and let the interface convert this into a hex file that can be sent ('flashed') to the micro:bit [1, 3]. You simply plug the micro:bit into a USB port on your computer and the

computer handles everything [1, 2, 3, 4, 5]. While the file is being downloaded to the micro:bit a yellow LED on the back of the PCB rapidly flashes. Once the download has been successful the program automatically runs on the micro:bit, powered from the 5V supply from the USB cable. If there is an error, the micro:bit scrolls a message on its LED display – usually telling you the line number where there may be a problem.

Python and Mu

You can also download the free Mu application (it does not need installing) that you can use to write micro Python programs on your PC (rather than online) and 'flash' them to your BBC micro:bit board [5]. I have chosen to use the Mu editor in this article. There are a lot nice tutorials to help you start learning Python and a good PDF is shown in the references [4]. Sometimes when you are debugging a circuit, changing the code and so flashing it to the micro:bit time after time, you may get an error. I found that if I cancelled the box that came up (it may take a few clicks), it finally sent the code to the micro:bit all OK.

Where are my files?

The files are sent directly to the micro:bit via the software. When using the Mu editor on my PC [5] I found the micro:bit folder by going from the C drive to →users→???→python (where ??? is a name that is specific for your file system, eg your computer name). Once the program is working I usually go to this folder and copy a version into a separate folder in my Documents folder. If you don't change the file name, any changes you make will be saved over the top of the last version of the software, so it's worth saving versions using different file names to suit.

Notes on programming

You need to type in the code carefully. The Mu editor is very particular about spaces, as they are important for the whole structure and flow of the code. The two sets of code shown in **Figure 2** are identical apart from a single extra space (on line 5) – but it's enough to stop the program from working! This particular program will flash an LED (and series resistor) connected between port 13 and ground.

The programs described in this article are all very simple and won't win any awards for programming style, but they will get you started. This approach has obvious limitations; when I tried to extend the 'letters' Morse code program (see later) to include the numbers and signs all together, I found it would not 'flash' to the micro:bit, even though I knew each separate program worked OK. I guess there were simply too many 'if' statements, so better quality coding style is needed (probably making use of Python 'dictionaries' etc). I have added notes in the code to help explain. These note are for people who have at least gone through the basic Python tutorials [1, 3, 4, 5].

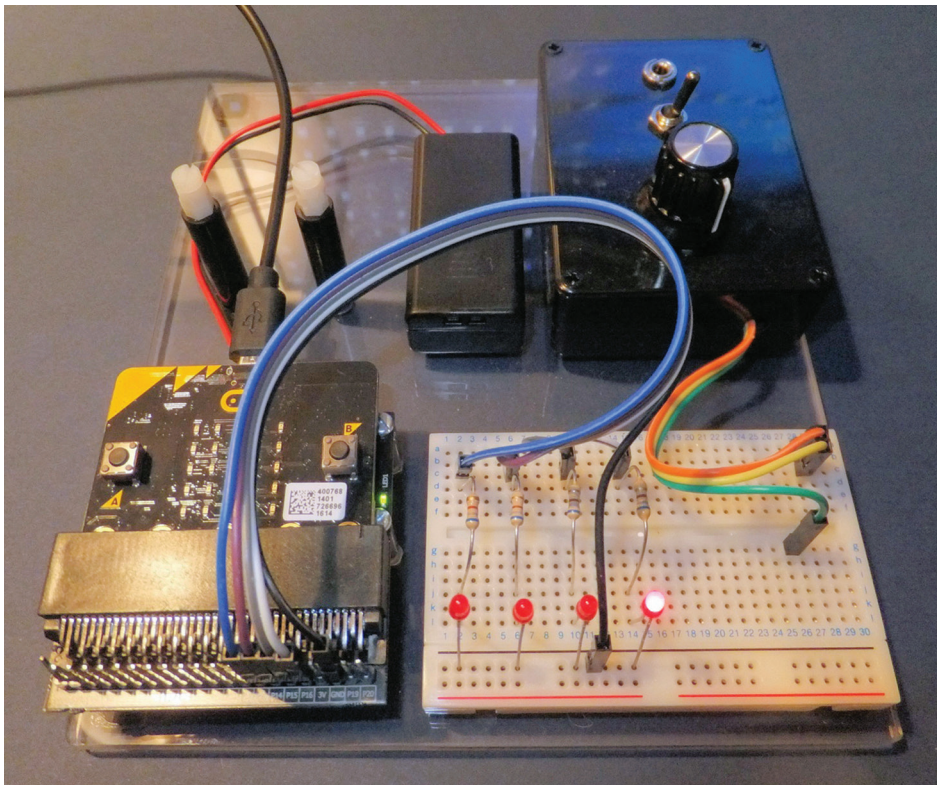


PHOTO 2: My experimental micro:bit setup showing the edge connector, breadboard and battery holder (2 x AAA). The black plastic box wires to a stereo 3.5mm socket for headphones, a 1k volume pot and switch for mono / stereo. The two plastic post on the left with two nylon bolts can be used to support the micro:bit via the 4mm holes if the edge connector is not needed.

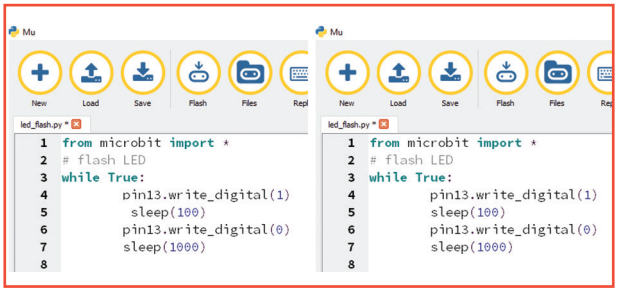


FIGURE 2: The extra space on line 5 of the left hand example is sufficient to stop the program working.

Demo / example programs

I have created some demo programs so you can quickly get an idea what the micro:bit can do [5, 7]. For examples there is a program that goes through some of the images the micro:bit can display as well as a simple clock and compass direction images. Load them up into Mu and 'flash' them into the micro:bit to see what they do. If you make any changes to the program it will save them over the top of the original file, so it's worth saving a copy of the original somewhere if you want to go back to it later.

Project 1: micro:bit controls LEDs

The micro:bit board comes with a built in 5 x 5 (= 25) LED display, so it can obviously control LEDs. The code shown in **Figure 2** (the example of the extra space) shows how easy it is to make an LED flash on port 13 say. The edge connector on the PCB has many of the output ports of the microcontroller assessable for you to use. Some are used to control the on-board 5 x 5 display so we will use some of the other 'free' ports.

Jonathan Hare, G1EXG
j.p.hare@sussex.ac.uk

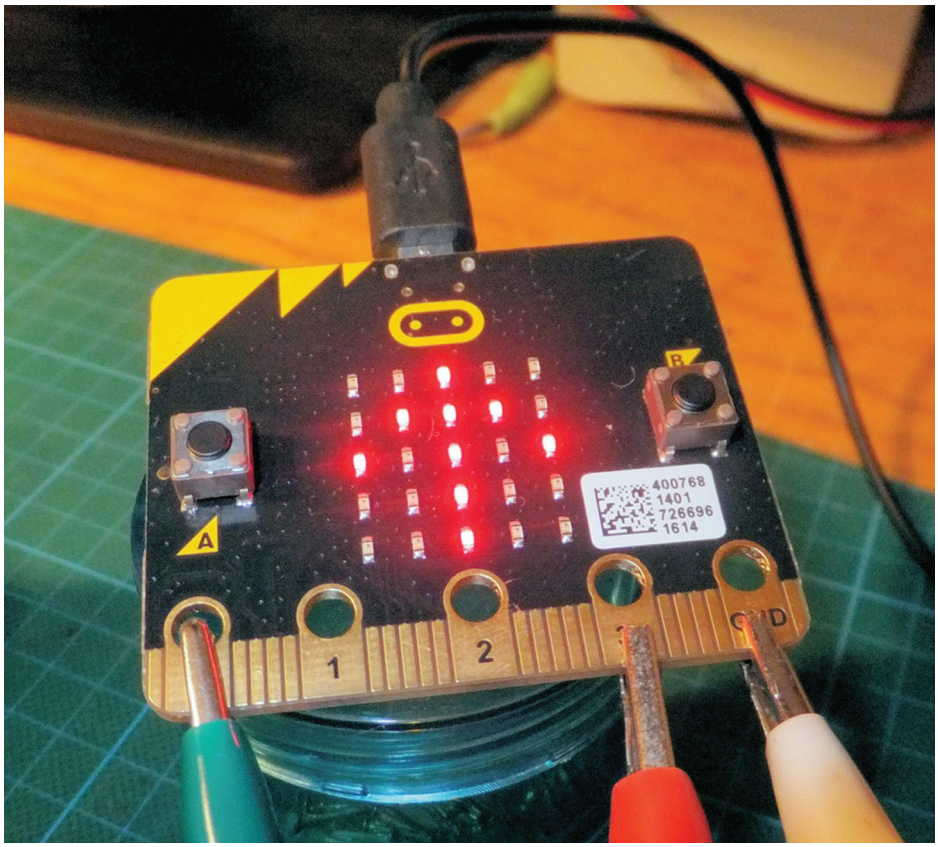


PHOTO 3: Rotator controller readout display (arrow pointing 'north').

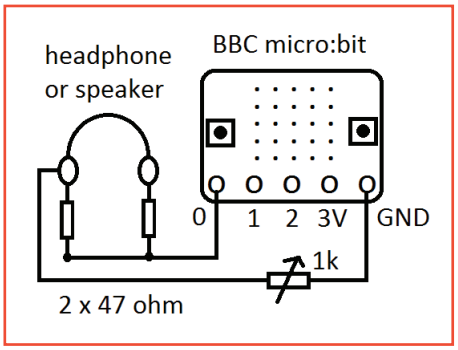


FIGURE 3: Morse demo circuit.

I wired four LEDs to port 13 to port 16 using series resistors (680 ohms) between each pin and LED. You can download the code and take a look at it: it simply lights the LEDs so they chase one after the other up and down the line of LEDs (see my video [7]). You can see that once you have the basic Python code set up it's as easy as 'pin13.write_digital(1)' to set a port high (3V) or 'pin13.write_digital(0)' to set low (0V). These outputs can turn things like LEDs on and off but also provide pulse width modulation (PWM) to create analogue outputs and power control etc.

Project 2: Morse on a micro:bit

I have created three very simple Morse code programs that you can use to practice listening to Morse code characters: 1) Morse code letters 2) numbers and 3) signs.

The Morse code audio appears on port 0 (pin 0 on the micro:bit, and the earth 0V (GND) of course). This can drive a small loudspeaker (preferably high impedance) or headphones. I fitted a 47 ohm resistor in series with each headphone insert, then put them in parallel and the used a series 1k pot as a crude volume control (see **Figure 3**) via the headphones screened earth connection.

The format is the same with all three programs. The program plays a random Morse character and then leaves a 1 second gap for you to try and recognise the character. The LED displays the character for a second, then clears and the next character plays. This continues for as long as it's powered. The Letters program plays all 26 letters of the alphabet, numbers plays 0, 1…9 and the Signs program plays the Morse code for . , ? = / + and *.

Brief explanation of the Morse 'Numbers' Python code

You can download the programs from my website [5] into the MU editor but you should also open in Notepad (which all Windows machines have). Here is a brief explanation of the 'numbers' Morse code program (a part of which is shown in **Figure 4**). It starts with the 'import' statements, which instructs extra code to be included so we can make use of the 'random' and 'music' functions later on in the code. The main code is held in a 'While' loop that makes sure that the program keeps repeating itself rather than just running through once.

The code then generates a random number between 0 and 9 using 'random.randrange(9)'. If this generates (say) 3, then the 'if' statement will move the code along so it plays 3 in Morse code. This is done using 'music.play ("xx")'. If xx is C5:1 the micro:bit will play a C note (of a certain pitch – '5') for a period of one unit – a 'dit' (dot). If we use C5:3 we get a C for three units – a 'dah' (dash). "R.1" creates a gap of one unit between the 'dits' and 'dahs'. 'music.set-tempo (bpm=yyy)' sets the speed of the Morse: 180 is about 12 words per minute. A lower value gives slower Morse, and a higher value, faster.

Note the difference in Python between = (used to assign a value eg bpm=120) and == (used to see if something is equal to something else, eg if answer == 4…) [4].

Project 3: callsign identification

The Worthing Radio Club meets Wednesdays at Lancing Parish Hall. It is really too small a room to have a PA system to help those with hearing difficulties, as it would be uncomfortable for those who can hear well. One of the club members had the idea of asking all those with hearing problems to bring along their 70cm handheld transceivers with an earpiece. Now when we have a presentation the (licenced) speaker has a small 70cm transmitter running low power into a small antenna. This signal can then be picked up by the members' handheld radios and they can adjust their own volume as they require.

To be legal, I used a micro:bit to send the club callsign in Morse every 10 minutes. The LED display shows the current character being sent. The initial tests used 3V (two AAA's) to power the micro:bit but a simple Zener regulator circuit from the handheld supply would also do. I low-pass filtered the micro:bit output and fed the attenuated audio into the mic. line to the transmitter. It sends the club callsign every 10 minutes, but you can of course modify the code to send your own club callsign. Such a simple device could be used as the basis of simple beacons or test transmitters etc.

Project 4: rotator controller readout

Some of the micro:bit ports can accept analogue voltage inputs (0 – 3V), converting them to a value between 0 – 1023. Here I make use of this in a very simple and basic

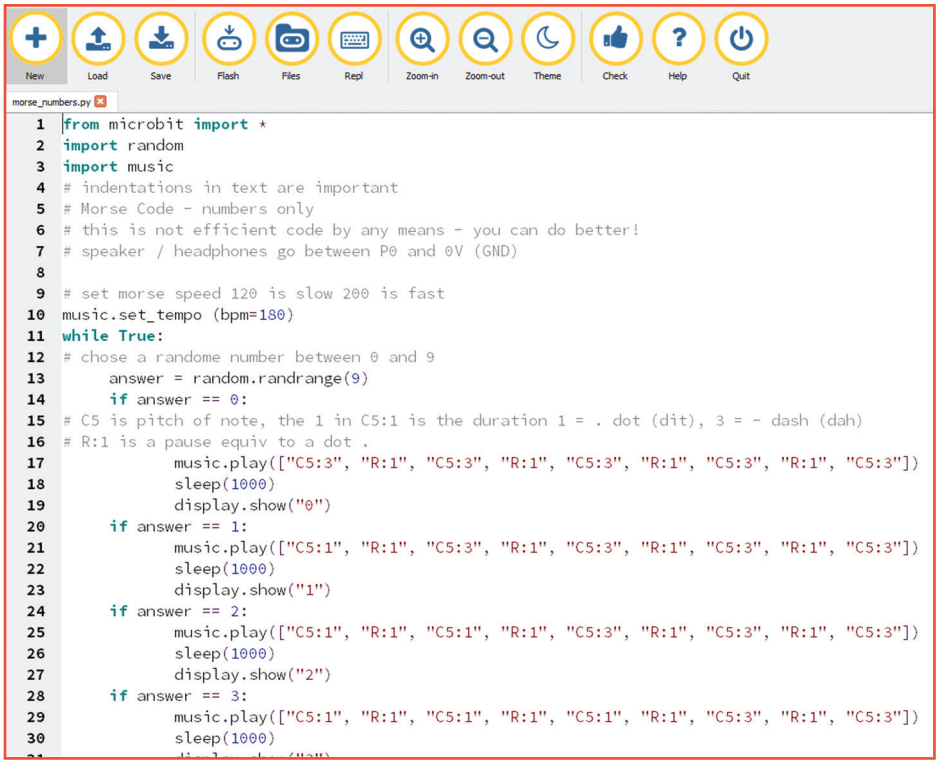


FIGURE 4: A section of the Numbers program.

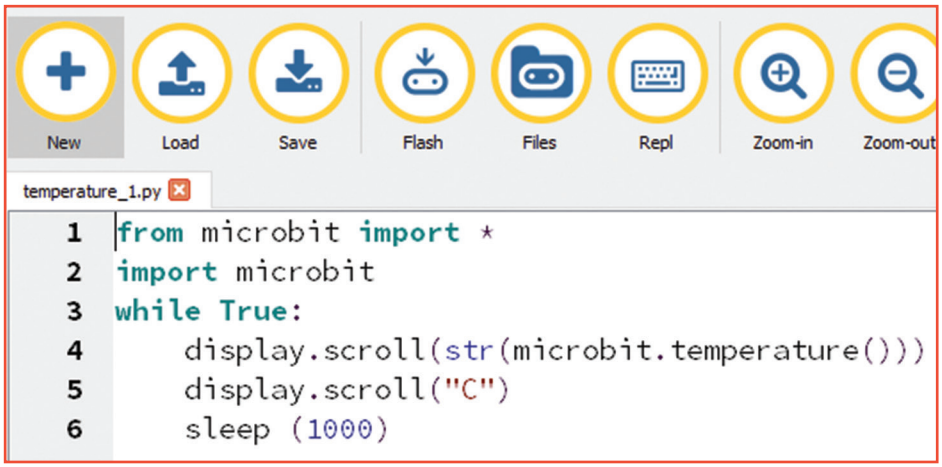


FIGURE 5: Code to scroll the micro:bit temperature across the screen every second.

antenna rotator display. If you are controlling something like a 2 element beam, which is not too directional, this display would be adequate. I often turn my beam into the direction of a station and then I peak the position for maximum signal. In this case a very accurate beam heading is not required – just a rough idea of the beam heading. As the rotator moves around its rough position is mirrored on the micro:bit display. I have put together a video about the micro:bit and you can see the rotator display in action on my YouTube video [7].

Many rotators have a potentiometer built into the rotator housing that provides position information voltage back to the controller down in the radio room. I used the 3V supply from the micro:bit to go to the 'ends' of the rotator potentiometer and sent the wiper signal to the micro:bit analogue input (port 0). It would be worth putting a low pass filter (eg a 22k resistor and 0.22μF capacitor) on the micro:bit input to reduce mains pickup and RF voltages causing problems.

The code converts the voltage reading into the corresponding arrow images showing the eight positions of the compass: N, NE, E, SE, S, SW, W, NW… and back to N again. The software simply checks the range of the voltage and displays the appropriate arrow, ie if the voltage input is between 0 and 64 show 'north' or UP arrow, 65 to 192 for NE etc. Each step is about 128 units except the two north headings that share 64 each 'end'. I haven't built in any hysteresis when checking the voltage so the display may jitter around if the rotator position just happens to be on transition from one compass heading to the neighbour (ie if the wind is blowing the rotator from side to side). **Photo 3** shows the display.

The N bearing being shared at 0 and 360 degree means the range of voltage detection in the code is slightly different from the other headings.

If the micro:bit display moves in the opposite way to the rotator then simply reverse the 0 and 3V connections to the rotator pot. If your rotator 'stop' position is in a different place, you can of course change the code to suit your particular setup, or even just rotate the position of the micro:bit display!

Project X: interesting things

There are all sorts of exciting things the micro:bit can do and I recommend you read the online information [4] and go through the simple tutorials (it's worth making sure you have the latest version and I will try and keep the link on my web page up to date [5]). The board has an accelerometer and a magnetometer and there is even an on-board thermometer that you can access. The code in **Figure 5** shows you how simple it is to access these functions – it scrolls the micro:bit temperature across the screen every second.

Other bits(!)

I will continue to develop projects with the micro:bit but I am also happy to post your ideas and code so we can build an archive for everyone to use – so if you are inspired to experiment with the micro:bit, do send me your code :-)

I would like to thank Alan, G4GNX and Phil, G4UDU from the Worthing club for their help during the preparation of this article.

Websearch

[1] BBC micro:bit – https://www.microbit.co.uk/
[2] https://thepihut.com/collections/microbit/products/micro-bit
[3] a good beginners book: *micro:bit basics* by Tony Loton, 2016. ISBN 9781537331010
[4] https://media.readthedocs.org/pdf/microbit-micropython/latest/microbit-micropython.pdf
[5] G1EXG's BBC micro:bit page – www.creative-science.org.uk/bbcmicrobit.html
[6] Python Mu editor for the PC – http://codewith.mu/
[7] My Youtube channel, www.youtube.com/watch?v=Wml9QreKvm0