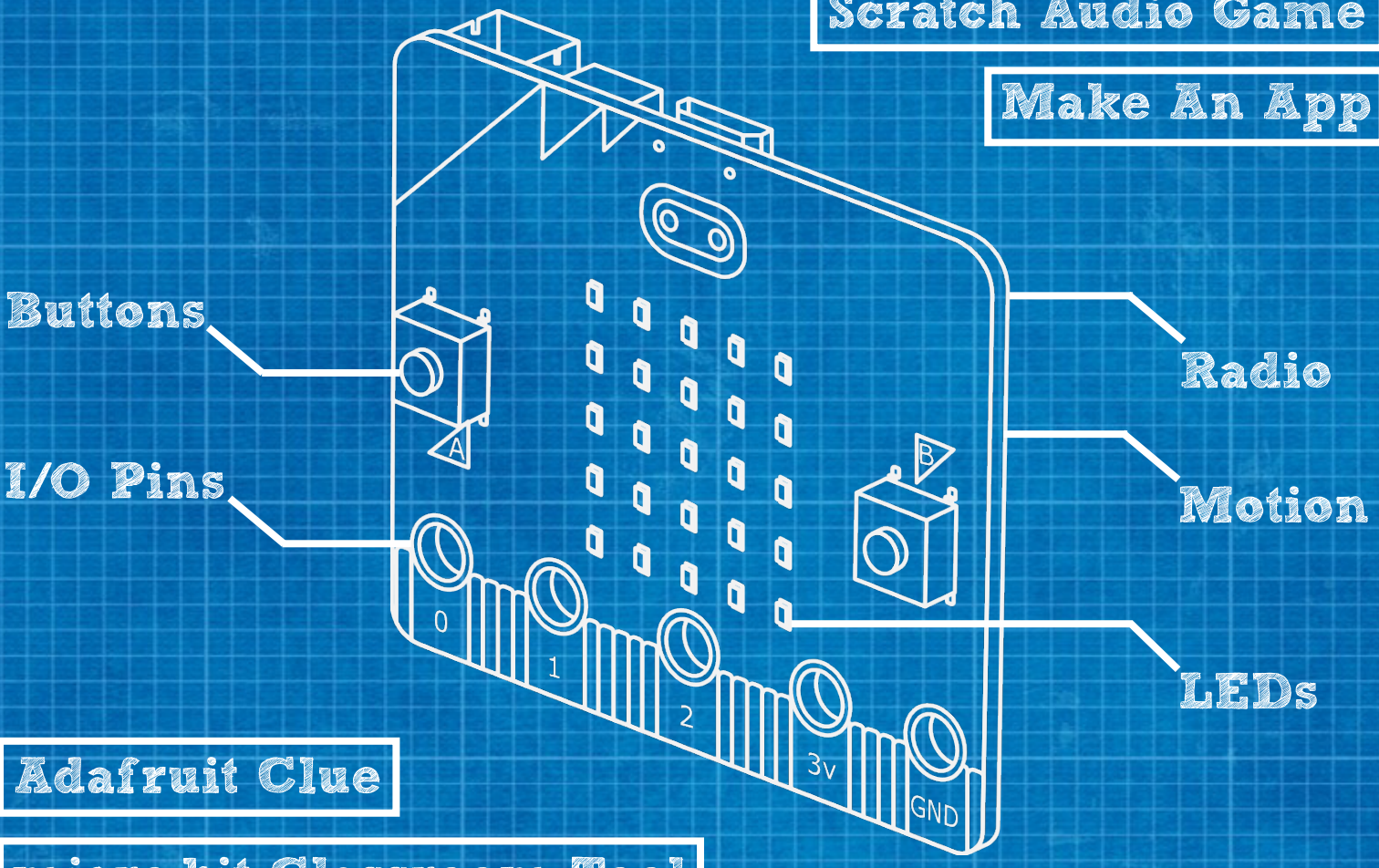
micro:mag

Issue 7
March 2020

5 AWESOME MICRO:BIT FEATURES

Scratch Audio Game

Make An App



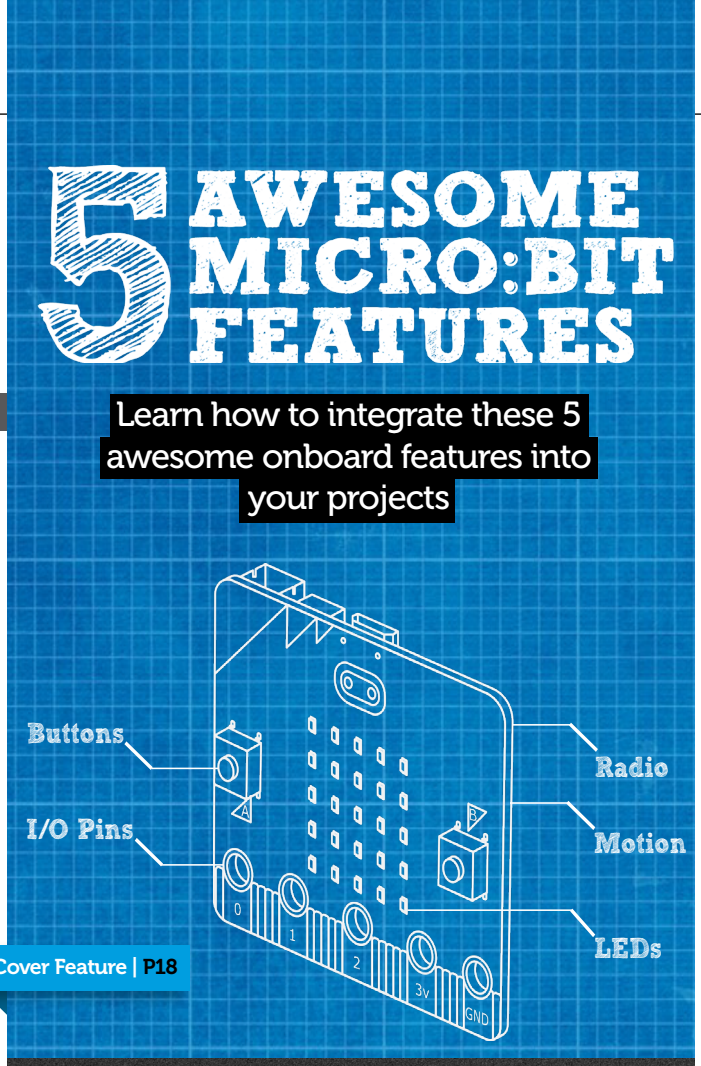
Adafruit Clue

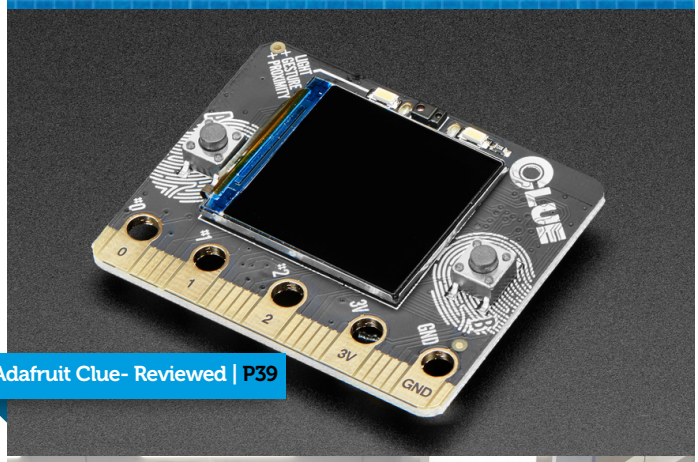
micro:bit Classroom Tool

THE UNOFFICIAL MICRO:BIT COMMUNITY MAGAZINE

micromag.cc

Contents





Hello World

Joshua Lowe
Senior Editor

Welcome to Issue 7 of micro:mag! We hope you've all had a great start to 2020.

We have an amazing Issue 7 for you packed with some amazing content written by the micro:bit community. It's important to note that if you have an amazing story, project or tutorial that you want to share with us, we are always accepting contributions via our new contributions page: micromag.cc/contribute.

In this issue, we cover five amazing onboard micro:bit features and show you how to integrate them into your projects using MakeCode, EduBlocks and MicroPython. We hope this serves as a great beginners guide for new users wanting to explore the BBC micro:bit. As for our amazing community articles, we have the Micro:bit Educational Foundation sharing their new website and classroom tool, two new exciting developments. Also, Farnell share with us how they take the micro:bit from a design to a finished product explaining each step along the way (Big thanks to the factory in china for providing images, especially given the current circumstances). In our make section, learn how to make an audio game in Scratch & micro:bit as well as learning how to use the temperature sensor in micro:hit! As always, we hope you enjoy this new issue of micro:mag!

Meet the team

Kerry Kidd
Editor In Chief

Kerry is a freelance programmer/educator who enjoys writing tutorials and tinkering with the micro:bit

Follow me:
@RaspiKidd
raspikidd.com

Joshua Lowe
Senior Editor

Josh is a young coder & creator of the Edublocks tool for micro:bit He has delivered lots of workshops & talks around the world.

Follow me:
@all_about_code
edublocks.org

Luke Castle
Editor + Designer

Luke is a student & event organiser currently studying Computer Science with a passion for Raspberry Pi and micro:bit

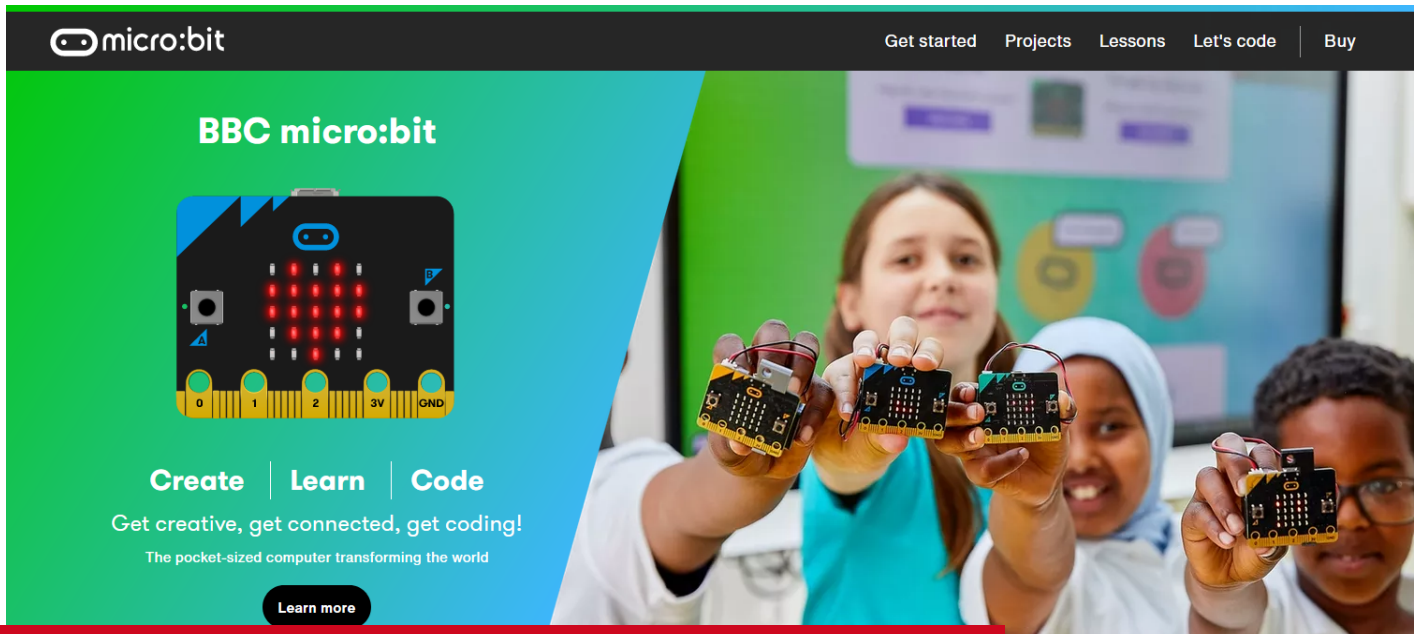
Follow me:
@YorkPiJam
@lukecastle03

Contributors

- Les Pounder
- Micro:bit Foundation
- Jon Haavie
- Natalie Ward
- Sarah Fawcett
- Patrick Benfield
- Marjan Milanov
- David Held
- Sean McManus
- Tony Goodhew
- Zayd Nashed
- Jay
- Kitronik LTD
- Adafruit Industries
- 4tronix

SCRATCH AUDIO GAME

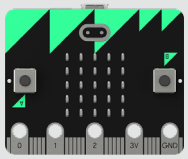
Learn how to integrate the BBC micro:bit and Scratch to make a cool game



BRAND NEW micro:bit Website

A new free educational platform to help teachers teach computing with purpose. By Micro:bit Educational Foundation & Nominet.

About the Author



The Micro:bit Educational Foundation is a not for profit organisation. Our vision is to inspire every child to create their best digital future.

The Foundation was legally established with the support of our founding members in September 2016.

microbit.org

This Month, the Micro:bit Educational Foundation, with funding from its founding partner Nominet, has launched a new free online educational platform including micro:bit classroom, a unique tool for schools, as well as comprehensive educational resources to aid planning, share ideas and save teachers' time. The new platform aims to empower all UK teachers – from computing aficionados to first-time coders – to bring foundational computing concepts, computational thinking skills and aspirations into lessons via the BBC micro:bit.

Four years ago one million BBC micro:bit devices were delivered to every UK secondary school and since then the micro:bit has gone global, with devices embraced by world-leading education systems including Canada, Finland, Denmark, Singapore, Taiwan and Hong Kong. The micro:bit is also being used as a strategic device for educational programmes investing in the acceleration of critical 21st century skills as demonstrated by the Western Balkan countries, and many South American countries including Chile, Uruguay, Colombia and Brazil.

The launch is part of the Foundation's ambitious plans to democratise technology and broaden young people's access to successful digital futures, beginning with computing education using the BBC micro:bit. It is committed to introducing every UK child to the creative possibilities of coding and the applications of technology to their real-world experience, so they become designers and creators, not just consumers of technology.

The benefits

Time-saving features are built into the new platform – the BBC micro:bit basics can be learned by any teacher in just an hour and micro:bit classroom makes it possible to set-up programming lessons in less than 2 minutes.

About micro:bit Classroom

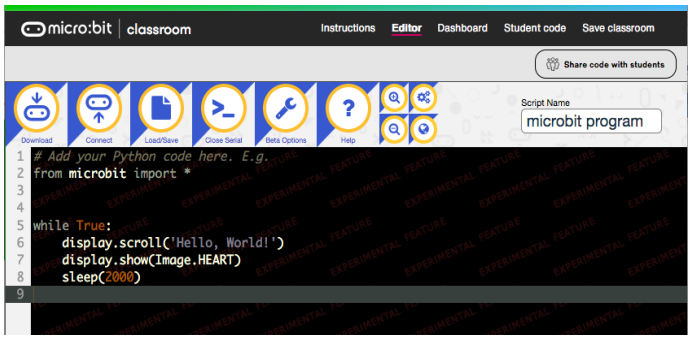
The new micro:bit classroom product allows teachers to view student work live and in one place, as well as stop and resume lessons at any point. Lessons are also easily downloadable to record students work, and there is no need to remember usernames or passwords because no registration is required. No student data is stored outside the school's own secure storage. The tool is uniquely designed to maintain student privacy.

Using the tool is easy. Simply go to classroom.microbit.org to get started. You'll be asked a few bits of key information.

- **The name of your classroom activity**
- **The programming language you'd like to use. For now there is a choice of Makecode & Python but this may expand in the future.**
- **Whether or not you'd like to use Temporary Local Storage.**

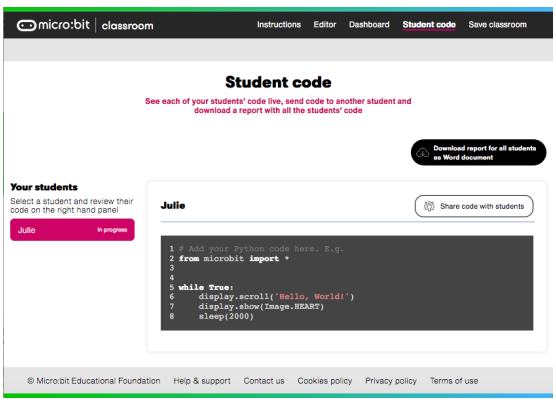
Now you've got your classroom setup. There are a few things you'll be able to do with this new tool.

Share code with your students



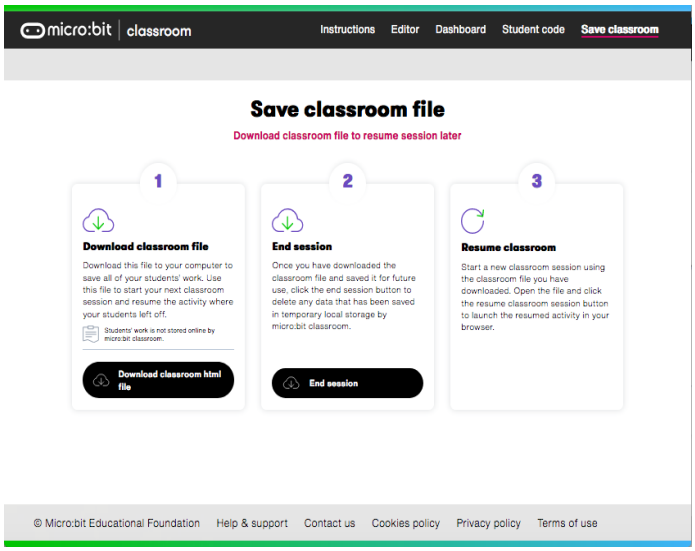
Using your choice of MakeCode or Python editor you can share code with your students at a click of a button from within micro:bit classroom.

View your students' code live



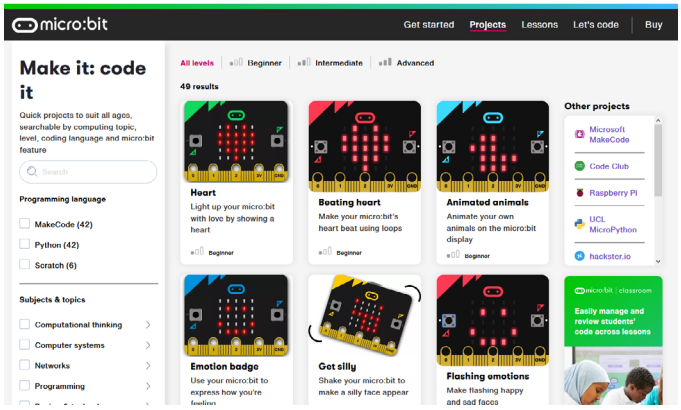
On the student code page you can view all your students' code in one place as they are working and download a report as a Word document.

Resume the lesson at a later date



The save classroom page lets you download your classroom file. Simply open this file in a browser to resume the activity where everyone left off.

About the new projects page



On the new website there is a host of new lesson resources, plans and projects linked to the curriculum gives teachers everything they need to help them deliver an entire term of inspiring content with ease. The majority of these are in the form of the new Make It: Code It projects written by the Foundation's education team. There are currently 49 projects that cover MakeCode, Python and Scratch with easy to follow instructions, downloadable Hex files as well as a button to load each code example directly into micro:bit Classroom.

We are really excited about these new developments in the micro:bit ecosystem and hope that they lower the barrier to entry even further for students and teachers using micro:bit in their classrooms.

Special Thanks

The new website and coding tool were made possible with funding from: NOMINET

SUPER:BIT Takes Off in Norway

60,000 kids and teachers gain access to the micro:bit in Norway as part of the new Super:bit program. By **Jon Haavie**

About the Author

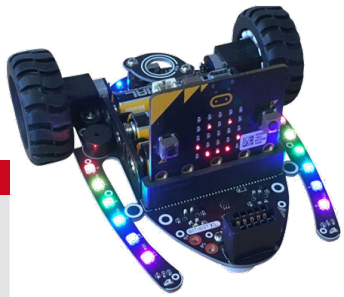
Interested in developing tools for involving people in science and technology.

Project developer @ super:bit, the largest EdTech initative in Norway

"Coding with micro:bit is great, but when you connect it to servos, robots and LEDs to create your own projects, the real fun begins" a teacher told me some years ago. Now, every 12-year old in Norway will have this opportunity. Over the next two years they will take part in the super:bit program, involving 60.000 kids and their teachers, ten science centers, 180 code clubs as well as the national broadcaster NRK.

to be creative with programming and to develop their maker skills and interests. Every class taking part in the program will attend a two hour educational program and the teachers will be offered teachers training. Each school will receive a kit with 20 micro:bits, 10 robots (Bit:bot XL from 4tronix), LEDs, servos and speakers. Simultaneously, 180 code clubs at Lær Kidsa Koding, a Norwegian volunteer movement, will receive the same equipment so that kids can continue after school. In this way the project aims at connecting formal and informal learning. Educational resources are available at web page superbit.no and the national broadcaster NRK is making videos to inspire both teachers,kids and volunteers.

BitBot XL



Left:
Each school and code club will recieve a class kit.

Left: The first challenge is to program the robot to drive on a road in the city. Learning by failing is at the heart of it all.

The super:bit challenge is to program a self-driving robot (Bit:bot XL) to drive around a city. First, kids use simple blocks in MakeCode, programming the robot to move forward 1 meter, turn and return. Later, they get more advanced challenges such as driving on a road, following a black line using the line-following sensor, using the ultrasound sensor to avoid a collision, or build and program traffic lights using LEDs. Once they get started, the aim is that they come up with their own ideas for a project in the city using the equipment they have access to. Can you make street lights that automatically turn on when it gets dark? Can you control the robot with a boom barrier?

How can you make an alarm, technology for a smart house or a light searching solar panel?

super:bit has been funded by the Norwegian Ministry of Education and the DNB Savings Bank Foundation with a total of 7 million EUR. The program is a part of the ongoing reform of the Norwegian curriculum where programming, tinkering and making will be an integrated part of mathematics, science, arts and crafts.

Left: The schools can build their own cardboard city with traffic and street lights, windmills, or whatever they want. Creativity and imagination is an important part of super:bit

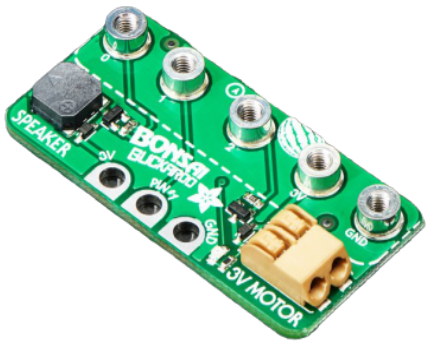
New Accessories SHOWCASE

We take a look at some of the brand new micro:bit accessories that have just hit the shelves!

Kitronik :ZIP Halo HD

The Kitronik Halo HD board for the BBC micro:bit incorporates 60 individually addressable full colour ZIP LEDs. It also breaks out P1 and P2 to a standard 0.1 footprint, it features a MEMS microphone for detection of sound, and a piezo buzzer to play sound. If that weren't enough, it also features an onboard real time clock (RTC) controlled by I2C lines from the microbit.

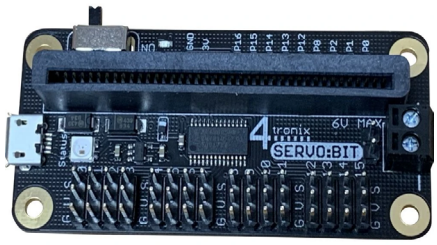
Price: £25.80 **Buy:** kitronik.co.uk/5672



Adafruit Bonsai Buckaroo

This little add-on board for the micro:bit adds sounds, a 3V motor controller and aligator clip connectors for measuring soil resistance with just 5 screws. A great little starter board for getting into plant and climate monitoring with the BBC micro:bit.

Price: $35 **Buy:** go.micromag.cc/bnsb



4Tronix Servo:Bit

This little board, exactly the same form factor as a Raspberry Pi Zero and the 4tronix Drive:Bit, allows you to control and power up to 16 servos at the same time.
The servo power connections can be made through the USB connector or the 2-pin screw terminal OR by using the 2-pin male header

Price: £9 **Buy:** go.micromag.cc/servb

micro:bit | classroom

The new micro:bit classroom tool makes managing coding sessions so much easier.

Send starter code to a whole group, view and capture their work and save the whole session as a single file to resume at a later date. Pupils do not have to worry about naming or saving their work.

micro:bit classroom works with MakeCode or Python. There are no student logins or passwords, just a simple PIN. **It's completely free and always will be**.

Try it today at **classroom.microbit.org**

micro:bit

microbit.org

MAKING OF MICRO:BIT

The Journey of a BBC micro:bit from Design through to Manufacture.

By **Natalie Ward, Farnell**

About the Author

Natalie Ward is Manufacturing Manager for Farnell, responsible for the manufacture of the BBC micro:bit, Raspberry Pi and other single board computers.

Web: farnell.com

As a reader of micromag, you will know the BBC micro:bit well - a pocket-sized programmable device that can be coded and controlled to bring ideas, games, and projects to life. But do you know how it is made?

The BBC micro:bit is manufactured by Farnell, an electronics distributor and part of Avnet. But distribution is not all that Farnell does. Farnell is also behind the manufacture of the Raspberry Pi and a range of associated accessories – so the BBC micro:bit is in good hands. Since becoming the exclusive licenced manufacturer of the BBC micro:bit in 2016, Farnell has manufactured over 4.5 million boards, which have been distributed across the globe.

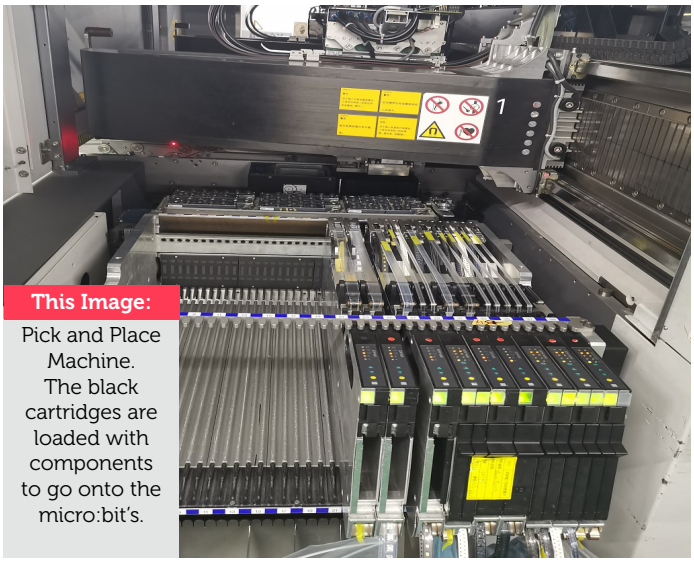
The small device begins its design journey at the micro:bit Foundation. The micro:bit Foundation is the not-for-profit organisation which was founded in 2016 to inspire every child to create their best digital future, using the BBC micro:bit. Once the design is tested and approved, Farnell step in to bring the design to life, working with its experienced manufacturing partner, Embest, based in China.

There are 6 steps to the manufacturing process for any electronic component – and the BBC micro:bit is no different:

- SMT (Surface Mount Technology)
- Inspection
- Reflow
- X-ray
- Final Inspection/Testing
- Kitting

SMT

SMT is the stage where the required components are laid out on the bare board – this would include the chips that provide the processing power, LEDs and other components such as the compass. All components are listed within the BOM – the Bill of Materials – which is like a shopping list of all the components required. There are two stages to the process. First of all the parameters are set. This is done by referring to the design file to identify the co-ordinates where each component should be positioned, then the components are placed onto the PCB (printed circuit board) in the right positions.



This Image:
Pick and Place Machine. The black cartridges are loaded with components to go onto the micro:bit's.

Inspection

Once SMT is complete, the boards are manually inspected against the design files. Every single board is inspected – for part orientation (which way the parts have been added) and if any parts are missing or incorrect. There is also a 100% visual inspection against the customer's design.

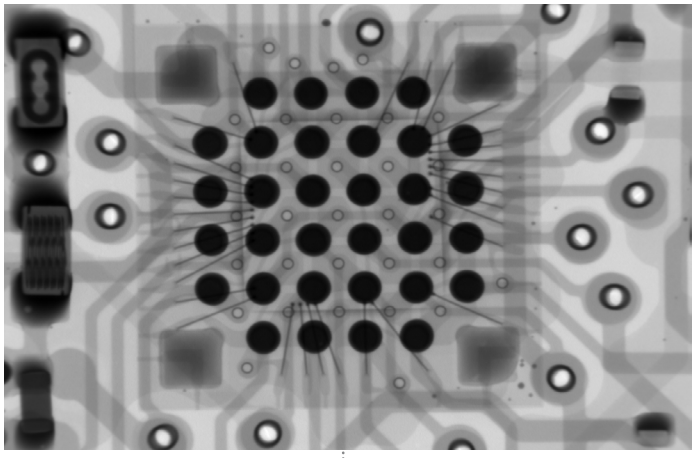
Reflow

Once it is determined that all the components are in the right positions, solder paste is applied to attach the components. This is called the reflow stage.

Left:
One of the Reflow ovens at the Embest Factory, China

X-Ray

Next up, every BBC micro:bit is x-rayed. This process will show whether any solders are touching – a "solder bridge", which would result in a short circuit of the board when used.



Left:
A PCB under the X-Ray machine showing where an IC would go

Final Inspection/Testing

Once all this is complete, each board is inspected again and subjected to various functionality testing to ensure the boards are working as they should, and look as they should.

Kitting and Shipping

Once the boards are approved as ready to go, the finished boards are packed into the boxes, either in singles, or in the range of packs that are available to buy – from the Go pack and Club Packs to the Bulk pack which includes 300 micro:bit for use in the classroom. The BBC micro:bit are then shipped to Farnell warehouses across the globe – in the UK, USA, China and Singapore, ready to send to customers.

That's it! Now you know what goes into making a BBC micro:bit, from design to manufacturing all the way to a classroom near you.

Left:
A micro:bit "club" pack which is manfucatured by Farnell. Pic: Fair Chance Learning

MICRO:BIT ENABLED CIRCUIT BLOCKS

Tinkering, Making and Hacking for Younger Learners. By **Patrick Benfield**

About the Author

Patrick is the founding director of two educational Makerspaces and Co-Founder of co.lab // Community Makers, a non-profit fab lab in Austin, Texas.

Twitter: @McLemoreAve

In the i.lab for Design + Making, the constructionist learning lab at the Magellan International School in Austin, Texas, the interconnected ideas of agency and empowerment are key principles that inform our approach to maker-centred learning.

Whether students are working with materials and hand tools or programming robots, the most powerful artefact of learning is the "maker mindset" being developed by each learner. With each setback, every iteration, and successful outcome, students begin to realize that they have the capacity to change the world around them.

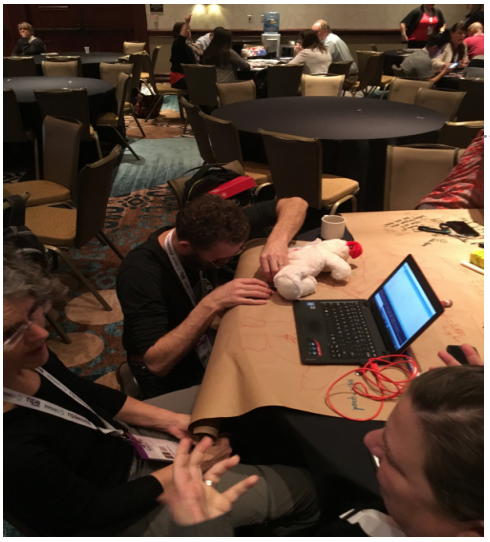
For our older primary and middle school students in particular, this sense of maker empowerment deepens when they delve into programming and electronics. As they gain experience with these concepts, they begin to see themselves as not just consumers of technology, but creators and inventors. However, for our youngest students, it can be challenging to interact with this technology in meaningful ways.

While there are many educational products aimed at this age group that claim to offer "kid-friendly" entry into electronics, programming, and robotics, they are typically cost-prohibitive and usually "black box" or otherwise obscure how the technology actually functions. In an effort to make these concepts more clear and accessible, with minimal barriers to seeing how things work, Patrick has created a set of modular electronics blocks designed for our youngest learners but still valuable for our oldest.

HackFun

The seed for this project was planted while at the South by Southwest education conference (SXSW Edu) during a maker educator workshop. At one table, where attendees had just finished a toy take apart activity, Patrick noticed one of the participants, Jeff Branson from SparkFun Electronics, doing something Patrick hadn't seen in person: taking control of a newly deconstructed toy with an Arduino-based board. Patrick was amazed not only

by the cool factor of what he was doing but also by the sheer confidence Jeff displayed throughout this process; here was the maker mindset in action. This "toy hacking" demonstration had all the elements that Patrick knew would be a hit with students: it was novel, empowering, and had just the right amount of mischievousness that encouraged looking at everyday objects as an invitation for exploration, rather than just a closed system. While this was a powerful idea to bring back to campus, Patrick's main concern was figuring out a way to make it accessible for learners of all ages.

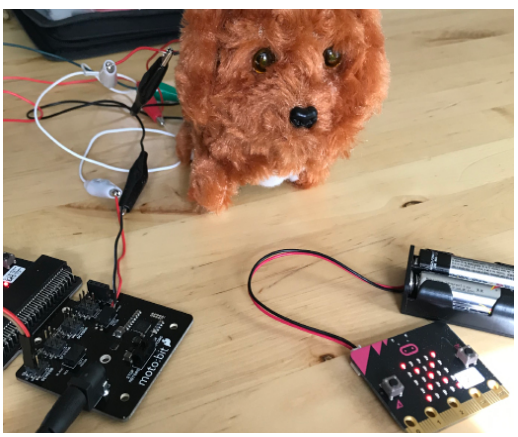


Left: Jeff Branson creating "HackBear" at SXSW edu

The Missing Link

The only missing element then was a way to bridge the gap between the fundamentals and advanced concepts. This presented Patrick with both an opportunity to create his own system to accomplish this and a way to contribute to the ideas from the maker community that this project was founded upon.

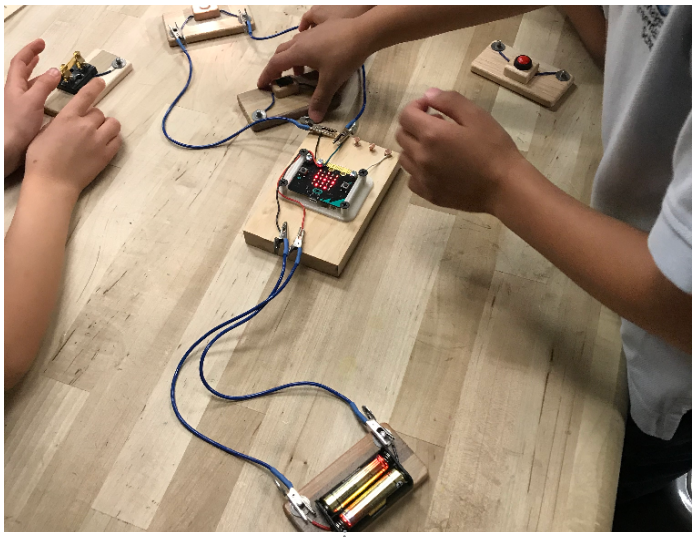
Patrick had a hunch that micro:bit could fulfil this role, but at that point, wasn't sure if it had the capability Patrick was looking for. To test this theory, Patrick wanted to see if he could recreate Jeff's toy hacking idea using one. With only a standard board, the initial attempts were unsuccessful but then Patrick discovered that the micro:bit, when connected to a board with an edge connector, provided access to pins that are normally not available. Shortly afterwards, Patrick's "hackpup" was born and so was the proof of concept.



Left: HackPup is born!

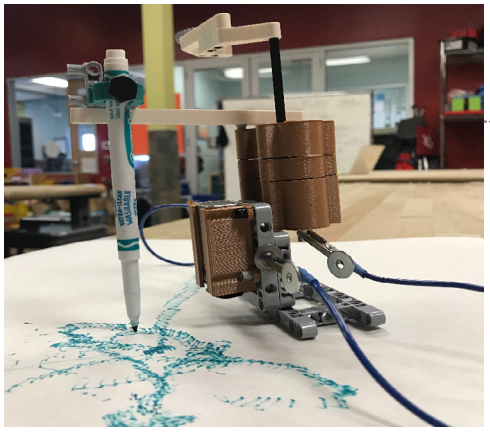
Micro:bit Enabled Blocks

Over the next several months, Patrick created a series of rough prototypes consisting of a micro:bit mounted on a wooden block and mixed them in with the "classic" circuit boards they had on-hand and began testing them with kindergarten and other early primary classes



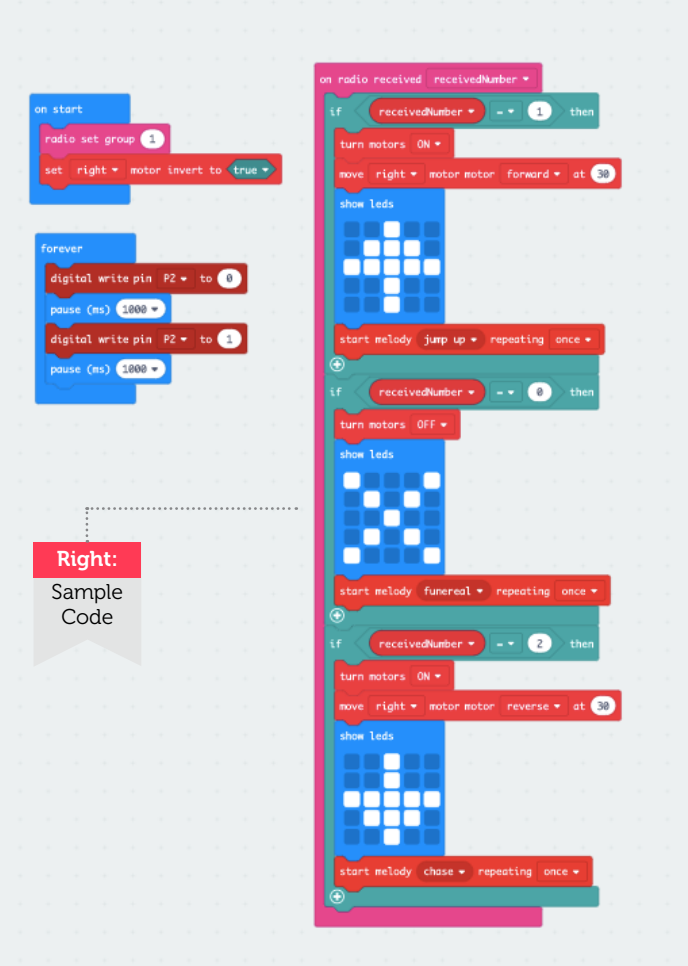
Above: Testing the prototype micro:bit blocks

As these sessions progressed, Patrick slowly began increasing the complexity so that students could take more control of the various components. This started with having a computer connected to a micro:bit block so students could see and tinker with the code for the LED and DC motor blocks. Soon after, they began to experiment with different combinations of inputs, outputs, and coding, much to their delight.



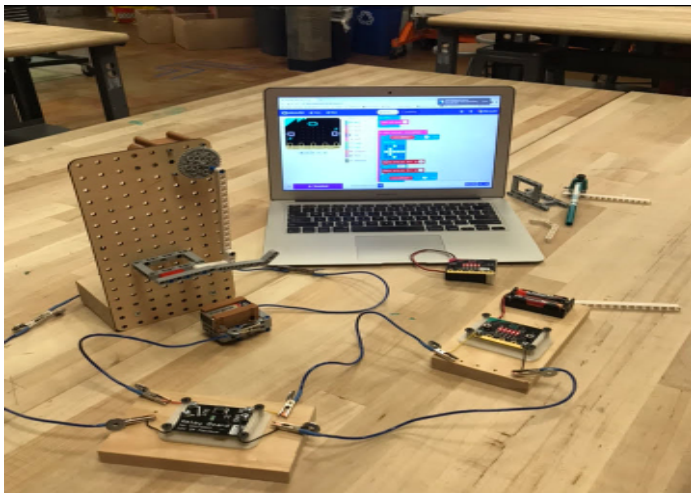
Left: Scribble Machine Example

Gradually, once students gained experience and confidence to take this setup even further, Patrick added another element into the mix, this time another micro:bit with the radio module enabled. Now they were able to remotely control objects within the systems they were building, including kinetic elements like linkages and scribbling machines.



Right:
Sample Code

In much the same way as Jeff's HackBear, this became an invitation to not only tinker with the physical components of the system, but also the computational aspects. This notion referred to as "computational tinkering" by the Exploratorium, combines the open-endedness of traditional tinkering with the powerful ideas offered by creative computing.

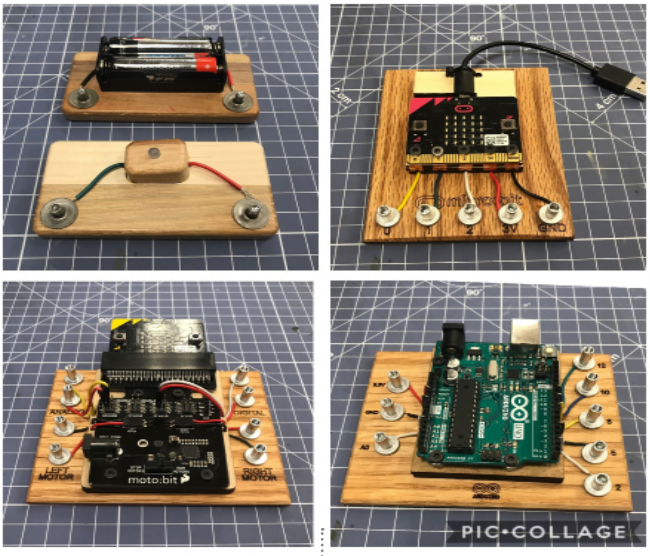


Above:
Ready for computational tinkering

What's Next?

With preliminary testing complete, Patrick has begun to create higher fidelity versions of the blocks with a form factor based ones we have here at Magellan that was purchased from the Children's Innovation Project (CIP). Besides being durable, the alligator clips that are sold with the CIP blocks have small washers soldered onto them, making it even easier for younger students to work with. Additionally, I've added several new blocks to the set, including servos and neo pixel rings.

As mentioned previously, the initial goal was to introduce micro:bits connected to carrier boards as a way to bridge the divide between the fundamentals of electronics and physical computing. However, as this project has moved forward and evolved over time, it has become apparent that there are still many more ideas that can be explored before making that transition.



Above:
The progression from least to the most complex

Along these lines, the next components that Patrick is "blockifying" are a variety of resistors that can be wired in series or parallel, LEDs that can be quickly replaced when students inadvertently burn them out (while learning about resistance), and a voltage meter (like this one from Chibitronics. Ideally, the physicality and ease of setup will offer students a clearer understanding of the phenomena for concepts like Ohm's Law. Besides the hardware elements, the plan is to also incorporate other avenues for programming, including Scratch 3.0 and MicroPython.

In the meantime, if this project sounds like something you'd like to try out for yourself, remix, or reimagine, the designs and fabrication instructions will be available soon on Instructables and Thingiverse.

Find out more at:

go.micromag.cc/mbcircuitblocks

WILL MICRO:BIT Revolutionise Education?

Marjan Milanov tells us how micro:bit will revolutionise our education

About the Author

Marjan is an ESL teacher who is an educational technology enthusiast, blogger, author of several PD programmes, teacher trainer and a British Council's master trainer in Serbia within the "21st Century Schools" programme.

Twitter: @paradoksija

Web: improveenglish.weebly.com

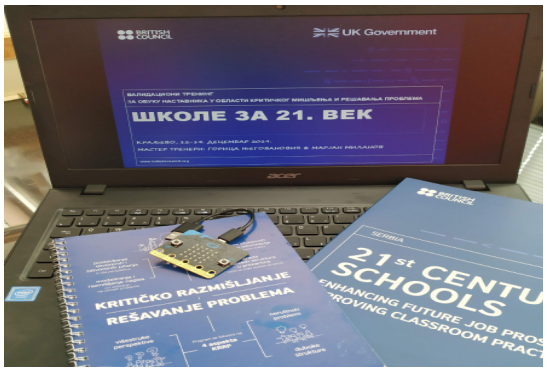
The British Council

The British Council is the UK's international organisation for cultural relations. They've created a really useful micro:bit for teachers course which has 12 modules covering the micro:bit and coding it with Makecode.

Visit the course site:

go.micromag.cc/bcmb

" What has micro:bit to do with critical thinking and problem-solving?" is the question Marjan often gets when delivering PD sessions within the "21st Century Schools" programme. The programme has been designed by the British Council and supported by the UK government. The pilot phase was conducted in 2017 and 2018 on a sample of chosen schools in the Western Balkans, and after excellent feedback and results from schools, teachers and students, the programme was launched in March 2019. More than 18000 teachers from 4000 schools in Serbia, Montenegro, Bosnia, Macedonia, Albania and Kosovo will be trained until May 2021. Elementary schools in the region will be given almost 110000 micro:bit devices to work with and implement in their teaching.



Local Ministries of Education have embraced and supported the programme

But, is it only about training teachers to code, training them to teach school children aged 10 to 15 to code? Actually, no! micro:bit is only one part of the three-day PD programme. The idea is to support teachers in developing some of the 21st-century core skills with their students. Primarily critical thinking and problem-solving combined with digital literacy. Some of the teachers have heard of micro:bit, have used Arduino and coded in Scratch, so they expect to hear and learn more about micro:bit. Others meet micro:bit for the first time and are a bit "scared" of new technologies, let alone coding! So, how does the programme connect the two? What has micro:bit to do with critical thinking and problem-solving? I would say nothing or everything! It's up to you to choose and decide!

Critical thinking and problem-solving are skills that can be developed and practised entirely without micro:bit or digital technologies. Making the distinction between fact and opinion, looking for evidence, having in mind different perspectives, asking high-order questions, looking for deep structures... having functional knowledge and solving real-life, non-routine problems...

But having all that technology at hand and not using it – Marjan thinks it is a luxury we do not have!!!

Depriving our children of using digital resources and tools - is a luxury we do not have!!!

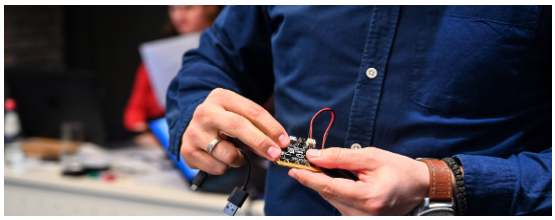
On the other hand, teaching children to code for the sake of coding is one viable option. But not the one we want! This should be only the first step toward something much more important - teaching them how to code or use technology in order to solve problems, in order to find solutions, in order to develop their creativity, to communicate and cooperate better.

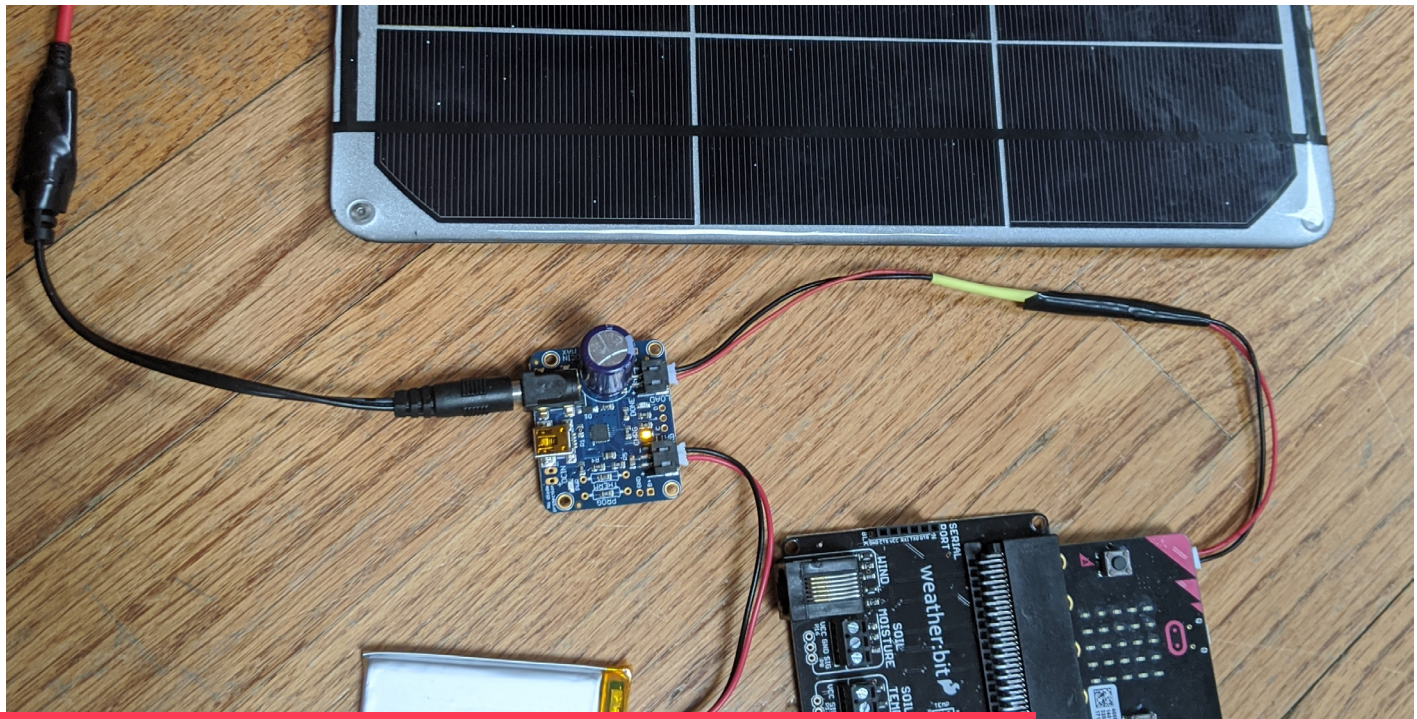
If we properly understand that, we will allow our children to play with the micro:bit while thinking they are playing and having fun, and actually we will be directing them towards developing their essential skills that are of such importance for life-long learning, for boosting their employment prospects, for preparing them to compete in the global labour market one day.

If we allow for the micro:bit to really enter not only our IT classrooms but also our Science, English, Maths or Geography classrooms, I firmly believe that the question is not whether it "will" transform out teaching practice and thus revolutionise our education. The question that remains is "How?".

And the answer is entirely in our (teachers') hands!

The children are ready! Are we?





IoT micro:bit WEATHER STATION

David Held tells us about his IOT weather station

About the Author

David Held is a retired educator and electro sound artist living in the Hudson Valley area of New York

@dheld

makingwithheld.com

When David first read about the announcement of the micro:bit before its release, He knew it was going to be HUGE in schools and the DIY/Maker Community. David had been teaching middle school students with Arduinos, but never thought the coding environment was suitable for the age group. As soon as the micro:bit was available for outside purchase, David scooped up as many as he could. The students in his classes loved working with them.

Since then, David has kept trying to push the envelope of what the micro:bit can do. As the micro:bit hardware environment grew, David wanted to try and build an Internet of Things project using the strengths of the micro:bit. David decided to build a weather station that would upload the weather data to a website.

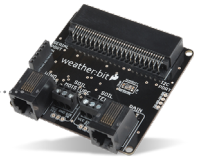
The micro:bit IoT weather station uses two micro:bits, one to take the weather measurements and transmit the data over the micro:bit's radio to the second micro:bit. The second micro:bit receives the radio data and uploads the data to his ThingSpeak (free) website. David used a fairly large solar panel and lithium battery so it would last several days of cloudy weather. You could go smaller on both the battery and panel.

Hardware Used

Weather Data micro:bit:

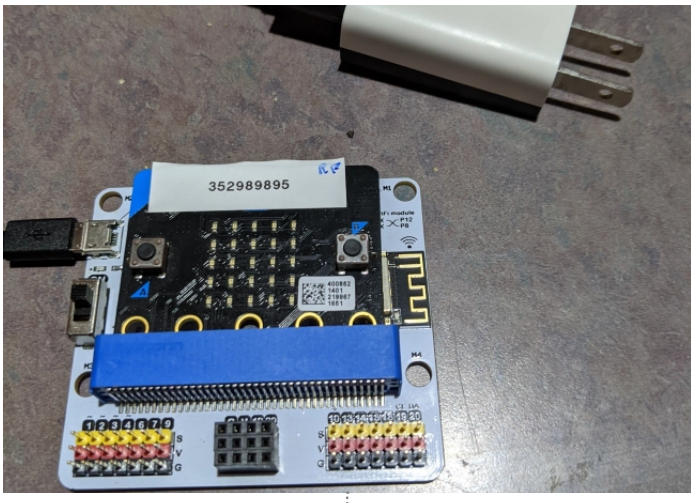
- micro:bit with Sparkfun Weather:bit
- Sparkfun Weather Meter (optional - for measuring wind and rain data)
- Adafruit USB / DC / Solar Lithium Ion/ Polymer charger
- Adafruit Medium 6V 2W Solar panel
- Adafruit Lithium Ion Polymer Battery - 3.7v 2500mAh

Right:
Sparkfun Weather:Bit Heart of the project.



IoT micro:bit:

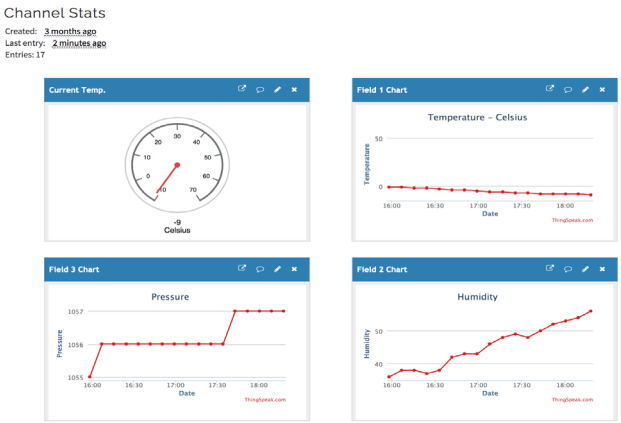
- micro:bit
- Elecfreaks IoT:Bit



Above:
The IoT micro:bit with the Elecfreaks board

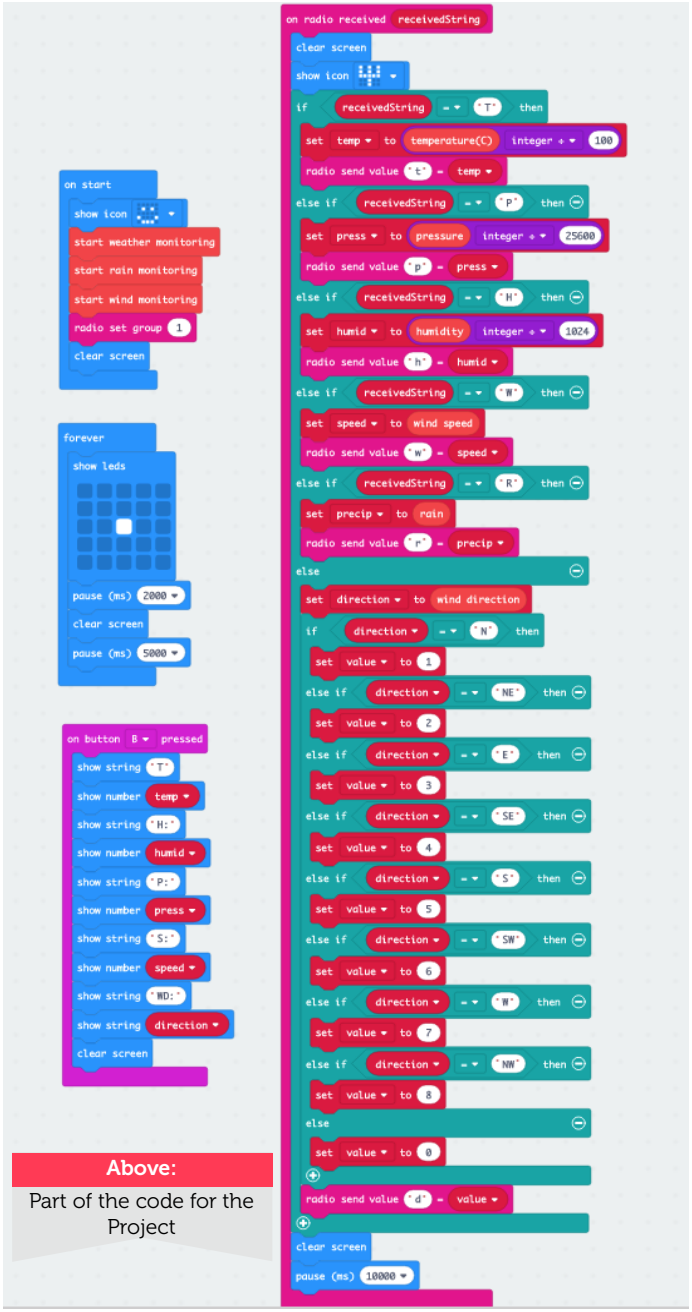
Software Overview

The software is written in the Makecode environment, as for beginners it is the most accessible. The software on the IoT micro:bit is structured, so that about every ten minutes, the micro:bit sends a request to the Weather micro:bit to send it the weather data. The IoT micro:bit then uploads the data to the Thingspeak website and waits another 10 minutes.



Weather micro:bit Software

The On Start blocks start the weather monitoring. There is an Extension created by Sparkfun and must be imported. Next, it sets the radio group number. The Forever blocks only display a single flashing LED to indicate that the micro:bit is running. About every 10 minutes the IoT micro:bit sends over the radio a request for the weather data. All of this is handled in the On Radio Received blocks. The micro:bit uses the "Radio Send Value" to send the variable name and data to the IoT micro:bit.



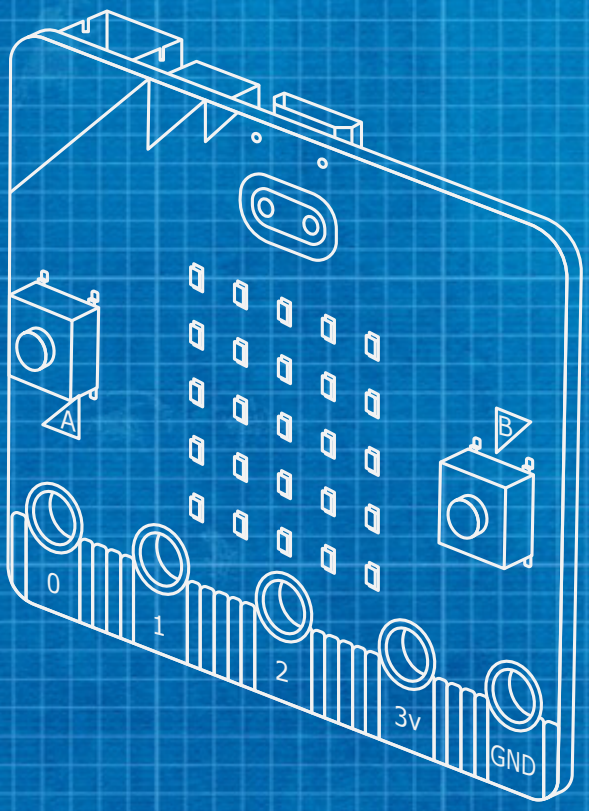
Above:
Part of the code for the Project

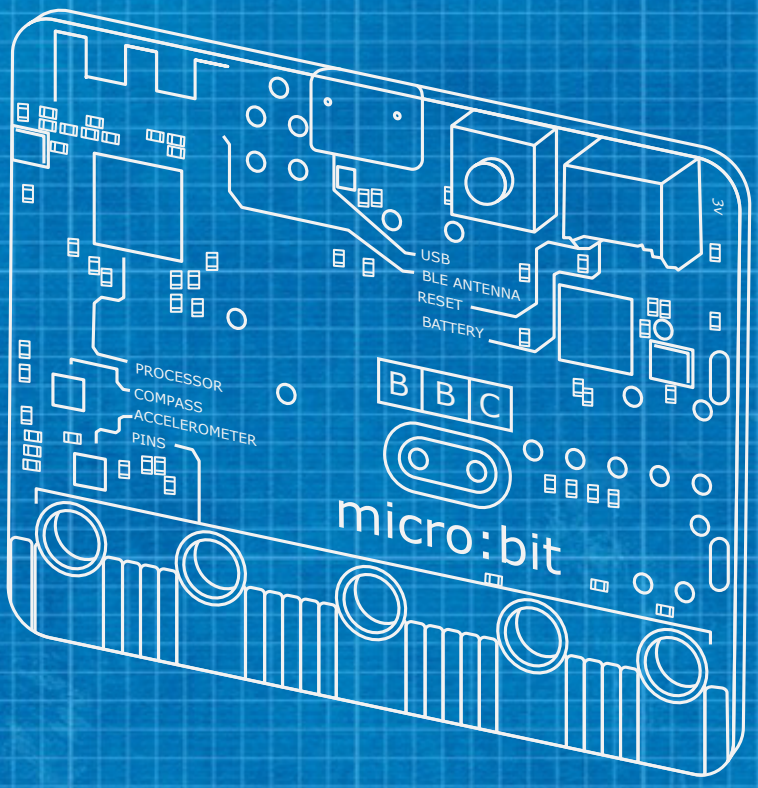
IoT micro:bit

The On Start blocks sets the radio group number and connects to the wireless network. This is an Extension created by ElecFreaks and must be imported. The Forever blocks display a single flashing LED to indicate that the micro:bit is running. About every 10 minutes it sends over the radio a request for the weather data via the On Received blocks. The Weather micro:bit then sends the data by the Radio Sent block to send the variable name and value to the IoT micro:bit. There is a 10-second delay between sends so there is no buffer overflow in the transmission.

The full software is available on my website makingwithheld.com.

5 AWESOME MICRO:BIT FEATURES





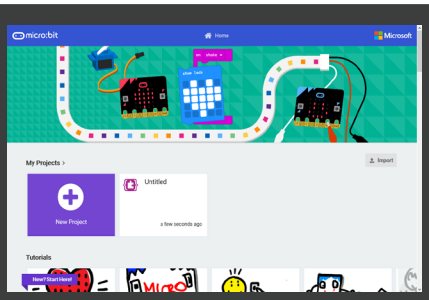
If you're reading this magazine, you probably already know that the BBC micro:bit is AWESOME. However, it's easy to get caught up in using accessories such as robots, speakers and LED lights and forget about some of the amazing onboard features the micro:bit has. That's why this issue we have compiled five of our favourite features found on the BBC micro:bit. and put them into this cover feature with code examples in three of the major code editors for the micro:bit (MakeCode, EduBlocks and Python) which should cater for all levels of coders.

This cover feature was written by Kerry Kidd

Getting Started

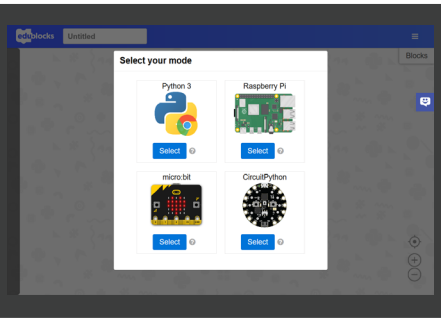
MakeCode

1. Open up a web browser of your choice
2. Go to makecode.microbit.org
3. Click on the purple “New Project” button
4. You’re ready to get coding!



EduBlocks

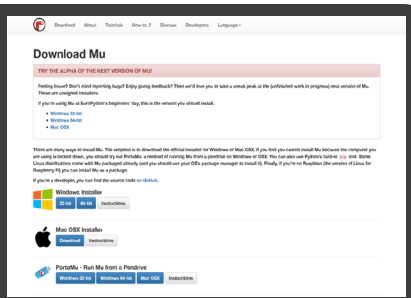
1. Open up a web browser of your choice
2. Go to app.edublocks.org
3. Click on the blue select button under micro:bit
4. You’re ready to get coding!



Python

We recommend using the Mu editor when coding Python

1. If not already installed, in your web browser go to codewith.mu
2. Click the relevant download link for your computer
3. Run the installer
4. Load up Mu and select the micro:bit mode when prompted

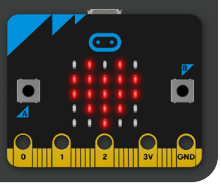


5x5 LED Matrix

About the LED matrix

On the front of your micro:bit, there are 25 red LEDs which you can use to scroll text and display images. A popular project to make with the micro:bit is a scrolling name badge and by using Makecode, EduBlocks or Python, it's simple to take advantage of these LEDs

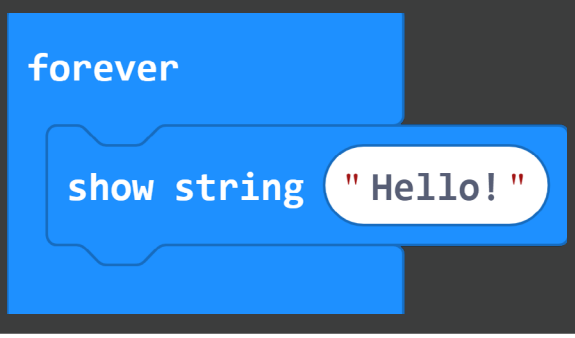
In this section we'll cover how to display text and then take it a bit further and display an image from the inbuilt image bank.



Display Text

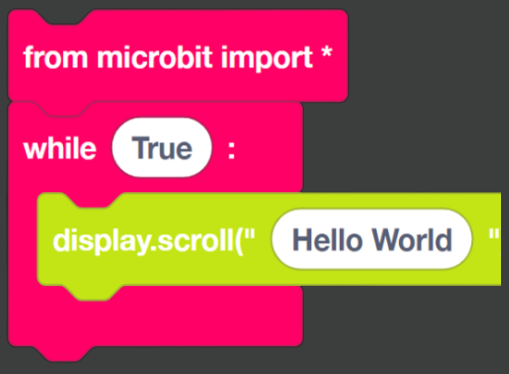
MakeCode

1. Open the MakeCode editor in your favourite browser: makecode.microbit.org
2. Delete the on start block by clicking and dragging it to the left odf the screen and dropping it on the recycle bin.
3. Click on Basic. Click and drag a show string "Hello!" block to the code area and attach it within the forever code block.
4. Click on Download to download the code to your computer.
5. Plug your micro:bit into your computer using a microUSB cable.
6. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



EduBlocks

1. Open the EduBlocks editor in your favourite browser: app.edublocks.org
2. Click on micro:bit.
3. Click on Basic. Click and drag a from microbit import * and drop it within the coding area.
4. Click on Basic. Click and drag a while True: block to the code area and attach it under from microbit import *.
5. Click on Display. Click and drag a display. scroll("Hello World") to the coding area and attach it within the while True: block.
6. Click on Download Hex, this will download the code to your computer.
7. Plug your micro:bit into your computer using a microUSB cable.
8. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



Python

1. Open your preferred method for programming Python.
2. Type from microbit import * this will import the micro:bit library to use with Python and press enter.
3. ype while True: this will create a while True loop meaning that the code inside it runs forever press enter.
4. Type display.scroll("Hello World") This will display Hello World on the micro:bit LED matrix forever.
5. Plug your micro:bit into your computer using a microUSB cable.
6. Plug your micro:bit in & click on Flash, this will download the code to your computer (if this gives you an error click flash again and it should work).

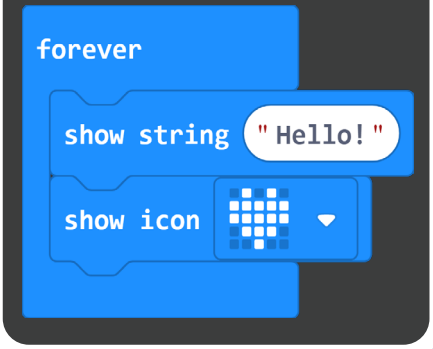
```
from microbit import *

while True:
    display.scroll("Hi")
```

Display Images

MakeCode

1. Click on Basic. Click and drag a show icon block to the code area and attach it under the show string "Hello! code block.
2. Click on Download to download the code to your computer.
3. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



EduBlocks

1. Click on Display. Click and drag a display.show(Image.HAPPY) block to the code area and attach it under the display.scroll("Hello World") code block.
2. Click on Basic. Click and drag a sleep(1000) block to the code area and attach it under display.show(Image. HAPPY)
3. Click on Download Hex to download the code to your computer and drag the file onto the MICROBIT drive.

Python

1. Type display.show(Image.HAPPY) and press enter. This will display a picture of a happy face.
2. Type sleep (1000). This will pause your program for 1 second. If this isn't included your program will skip over the happy face.
3. Click on Flash to download the code to your micro;bit.

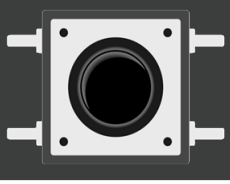
```
from microbit import *
while True:
    display.scroll("Hi")
    display.show(Image.Happy)
    sleep(1000)
```

A+B Buttons

About the Buttons

On the front of your micro:bit, there are 2 tactile buttons that you can use in your code to trigger an event on your micro:bit. You can respond to the press of Button A or Button B or respond to them both being pressed at the same time.

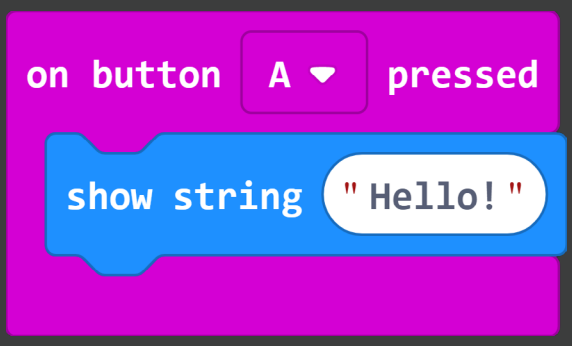
In this section we'll cover how to use the buttons to scroll some text and then create our very own emoji badge to display our mood.



Event Trigger

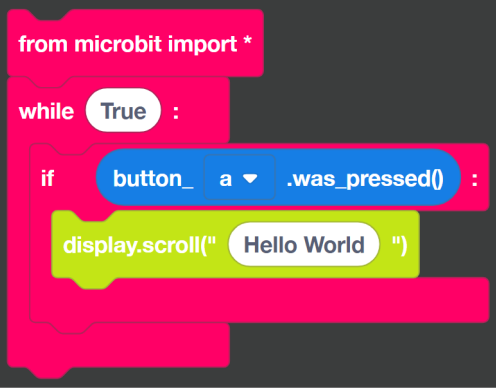
MakeCode

1. Open the MakeCode editor in your favourite browser: makecode.microbit.org
2. Delete the on start block & forever by clicking and dragging ithem to the left of the screen and dropping it on the recycle bin.
3. Click on Input. Click and drag a on button A pressed block. Then in Basic drag a show string block. Click on Download to download the code to your computer.
4. Plug your micro:bit into your computer using a microUSB cable.
5. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



EduBlocks

1. Open the EduBlocks editor in the browser and select micro:bit.
2. Click on Basic. Click and drag a from microbit import * and drop it within the coding area.
3. Click on Basic. Click and drag a while True: block to the code area and attach it under from microbit import *.
4. Click on Basic. Click and drag an if block. to the coding area and attach it within the while True: block. The inside of the input of the if block, drag a button was pressed block
5. Click on Display. Click and drag a display. scroll("Hello world") block inside the if block
6. Click on Download Hex, this will download the code to your computer.
7. Plug your micro:bit into your computer using a microUSB cable.
8. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



Python

1. Open your preferred method for programming Python.
2. Type from microbit import * this will import the micro:bit library to use with Python and press enter.
3. Type while True: this will create a while True loop meaning that the code inside it runs forever press enter.
4. Type if button_a.was_pressed(): . This will check for a button press.
5. Type display.scroll("Hello World") This will display Hello World on the micro:bit LED matrix forever.
6. Plug your micro:bit into your computer using a microUSB cable. and press flash to upload your code onto you micro:bit.

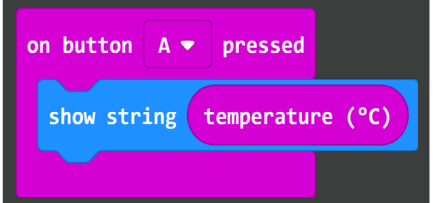
```
from microbit import *

while True:
    if button_a.was_pressed():
        display.scroll("Hi")
```

Read the temperature on a press

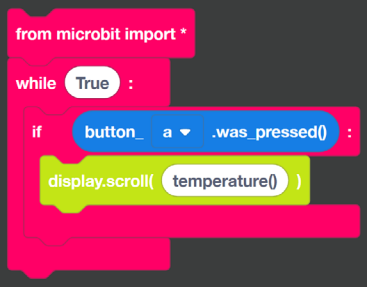
MakeCode

1. Click on Input and drag a temperature block into the "Hello" input of the show string block in your existing code.
2. Click on Download to download the code to your computer.
3. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



EduBlocks

1. Drag the display.scroll("Hello World") block into the bin.
2. Replace with a display.scroll(0) block.
3. Inside of the new block we have just dragged in, type in "temperature()" to read the temperature of the micro:bit's CPU.
4. Click on Download Hex to download the code to your computer and drag the file onto the MICROBIT drive.



Python

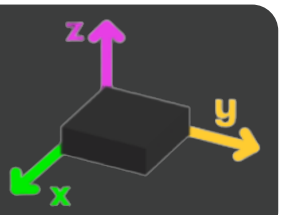
1. In your existing program from the section on the last page, replace "Hello World" with temperature(). Don't forget to remove quotation marks!!
2. Click on Flash to download the code to your micro;bit.

```
from microbit import *
while True:
    if button_a.was_pressed():
        display.scroll(temperature())
```

Motion

About the Accelerometer (Motion Sensor)

Your BBC micro:bit has a combined accelerometer and compass. We will be focusing on the accelerometer. You've most likely used one of these before, ever played a mobile game where you have to tilt the device to interact with the game? An accelerometer reads the tilt value and the game acts accordingly. We will be making a simple spirit level using the micro:bit's accelerometer, follow along!



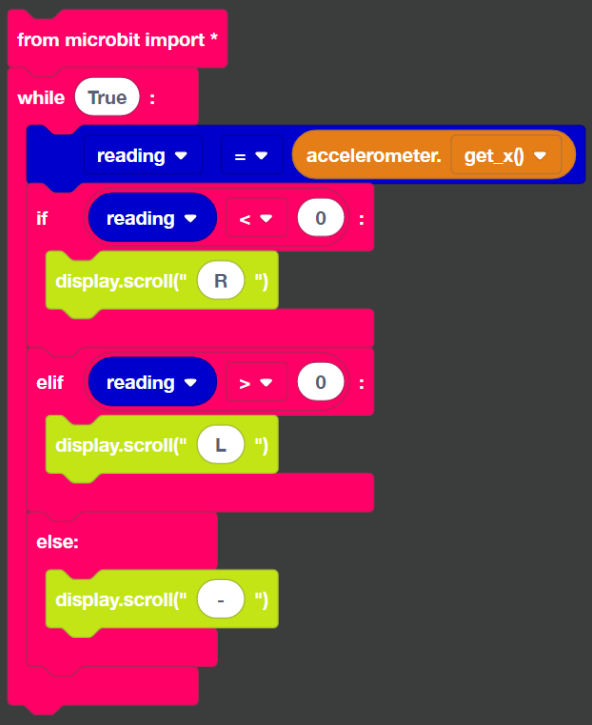
Simple Spirit Level

1. Click and drag the on start block to the left and drop it in the bin
2. Click on Variables Click on Make a Variable... Type reading and click on Ok
3. Click on Variables Click on set reading to 0 and drag it to the coding area and attach it within the forever block
4. Click on Input Click and drag acceleration (mg) x to the coding area and attach it within the 0 of the set reading to block
5. Click on Logic click and drag a if true then else block to the coding area and attach it under set reading to acceleration (mg) x block
6. Click on Logic Click and drag a 0 < 0 block to the coding area and attach it within the true of the if then block
7. Click on Variables click and drag reading to the coding area and attach it within the first 0 of the if then block
8. Click on Basic Click and drag a show string "Hello" block to the coding area and attach it under if reading < 0 then Click where it says Hello and type R Click on the small + below else
9. Click on Logic Click and drag a 0 < 0 block to the code area and attach it within the blank space of the else if block
10. Click on Variables click and drag a reading block to the coding area and attach it within the first 0 of the else if block
11. Click on Basic Click and drag a show string "Hello" block to the coding area and attach it under else if reading > 0 then Click on **Hello and type L
12. Click on Basic Click and drag a show string "Hello" block to the code area and click on Hello and type -

MakeCode

EduBlocks

1. Open EduBlocks within your favourite browser and click on micro:bit
2. Click on Basic Click on **from microbit import *** and drop it within the coding area
3. Click on Basic Click on while True: and attach it under **from microbit import ***
4. Click on Variables Click on Create variable... Type reading and click on OK
5. Click on Variables Click and drag a reading = 0 block to the code area and attach it within while True
6. Click on Accelerometer Click and drag a accelerometer.get_x() block to the code area and attach it within the 0 of the reading = 0 block
7. Click on Basic Click and drag an if True: block to the coding area and attach it under reading = accelerometer.get_x()
8. Click on Basic Click and drag a 0 == 0 block to the coding area and attach it within True of the if True: block. Click on the little arrrow next to == and click on <
9. Click on Variables Click and drag reading to the code area and attach it within the first 0 of the if block
10. Click on Display Click on display.show(Image.HAPPY) and drag it to the coding area and attach it under if reading < 0: Click where it says Image.HAPPY and type "R"
11. Click on Basic Click and drag a elif True: block to the coding area and attach it under the if reading < 0: block
12. Click on Basic Click and drag a 0 == 0 block to the coding area and attach it within the True of the elif True block. Click on the small arrow next to == and click on >
13. Click on Variables Click and drag a reading block to the code area and attach it within the first 0 of the elif
14. Click on Display Click and drag a display.show(Image.HAPPY) to the code area and attach it within the elif reading > 0 block Click on Image.HAPPY and type "L"
15. Click on Basic Click and drag a else: block to the code area and attach it under the elif reading > 0
16. Click on Display Click and drag a display.show(Image.HAPPY) block to the code area and attach it within the else: block. Click on Image.Happy and type "-"



Python

1. Open up your favourite Python editor for the micro:bit
2. Type from microbit import * This imports the micro:bit library to use within python so we can program with the micro:bit
3. Type while True: This create a loop that will go on forever
4. Type reading = accelerometer.get_x() This create a variable callled reading and sets it to the x axis of the accelerometer
5. Type if reading < 0: This creates a statement to check if the micro:bit is tilted to the left
6. Type display.show("R") If the micro:bit is tilted to the left it displays an "R" on screen to get you to tilt it back to the centre
7. Type elif reading > 0: This checks to to see if the micro:bit is tilted to the right
8. Type display.show("L") if the micro:bit is tilted to the right it displays an "L" on screen to get you to tilt the micro:bit back to the centre
9. Type else: This checks to make sure the micro:bit is centre
10. Type display.show("-") This displays a line on the screen if the micro:bit is sat centered.

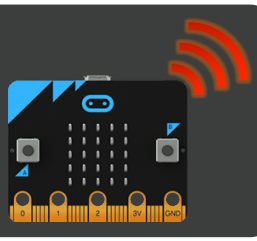
```
from microbit import *

while True:
    reading = accelermoter.get_x()
    if reading < 0:
        display.show("R")
    elif reading > 0:
        display.show("L")
    else:
        display.show("-")
```

Radio

About the Radio

Probably one of our favourite onbaord features of the BBC micro:bit is the Radio functionality. This allows two micro:bits to talk wirelessly to each other and it's really simple to use! On these pages we'll show you how to send and recieve messages.

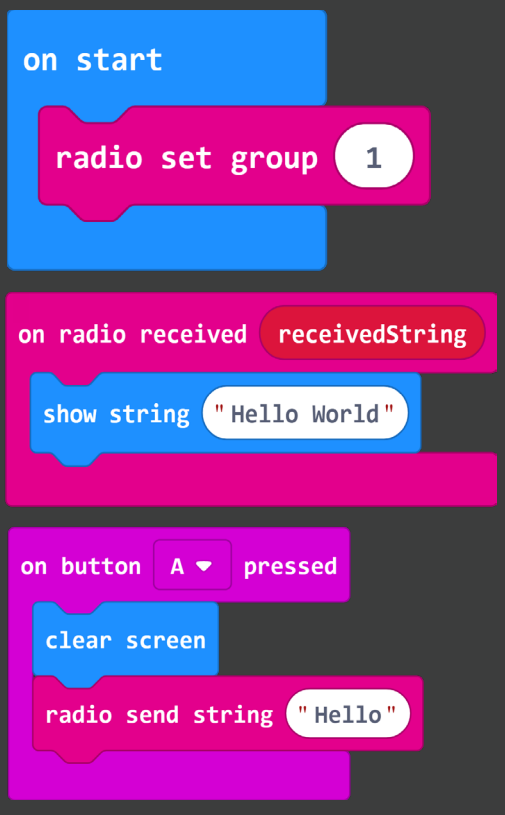


NOTE: You will need to flash the code onto two or more micro:bits!

Send + Recieve Messages

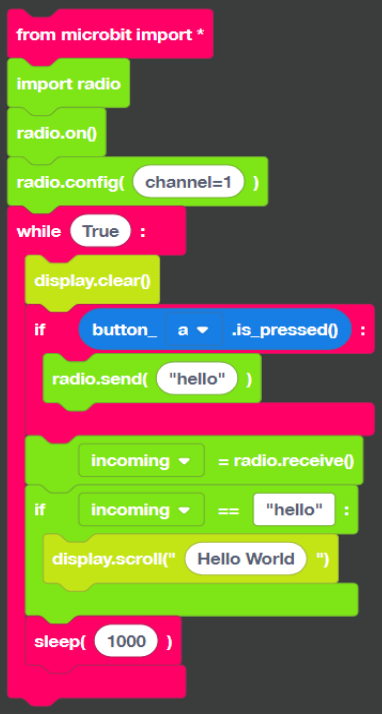
1. Open MakeCode within your favourite browser
2. Click and drag the forever block to the left to delete it as we will not be using that block
3. Click on Radio Click and drag a radio set group 1 to the code area and attach it within the on start block
4. Click on Input Click and drag a on button A pressed block to the code area
5. Click on Basic Click on ...more click and drag a clear screen block to the coding area and attach it within on button A pressed block
6. Click on Radio Click and drag a radio send string "" block to the coding area and attach it under the clear screen block within the on button A pressed block
7. Click within the blank space of the radio send string block and type Hello
8. Click on Radio Click and drag a on radio received receivedString block to the coding area
9. Click on Basic Click and drag a Show String "Hello" block to the coding area and attach it within the on radio received receivedString block.
10. Plug your micro:bit into your computer using a microUSB cable.
11. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.

MakeCode



1. Open EduBlocks within your favourite browser and click on micro:bit
2. Click on Basic Click and drag a **from microbit import *** block to the coding area
3. Click on Radio Click and drag a import radio block to the code area and attach it under the **from microbit import *** block
4. Click on Radio click and drag a radio.on() block to the coding area and attach it under the import radio block
5. Click on Radio Click and drag a radio.config(channel=7) block to the code area and attach it under radio.on()
6. Click on Basic Click and drag a while True: block to the code area and attach it under radio.config(channel=7) block
7. Click on Display Click and drag a display.clear block to the coding area and attach it within the while True block
8. Click on Basic click and drag an if True: block to the code area and attach it under display.clear
9. Click on Buttons Click and drag a button_a.is_pressed() block to the coding area and attach it within the if block where it says True
10. Click on Radio click and drag a radio.send("hello") block to the coding area and attach it within the if button_a.is_pressed() block
11. Click on Radio click and drag a incoming=radio.receive() block to the code area and attach it under the if button_a.is_pressed(): block
12. Click on Radio Click and drag a if incoming == "hello": block to the code area and attach it under the incoming=radio.received() block
13. Click on Display Click and drag a display.scroll("Hello World") block to the coding area and attach it within the if incoming =="hello" block
14. Click on Basic Click and drag a sleep(1000) block to the code area and attach it under the if incoming == "hello": block.

EduBlocks



1. Open your favourite micro:bit Python editor
2. Type from microbit import * this impoorts the micro:bit library for us to communicate with the micro:bit
3. Type import radio this will import the radio libaray for us to communicate with the radio on the micro:bit
4. Type radio.on() This will turn the radio module on so we can interact with it
5. Type radio.config(channel=1) this sets the radio channel to 1. It is important to make sure the micro:bits you are using are on the same channel. These can be numbered from 1-255.
6. Type while True: This will create a loop that will run forever
7. Type dispplay.clear() this clears the display on the micro:bit
8. Type if button_a.is_pressed(): this executes the next instuction if button a on the micro:bit is pressed
9. Type radio.send("hello") This sends the message hello to another micro:bit that is listening on the same channel
10. Type incoming=radio.receive() This sets a variable called incoming to radio.receive
11. Type if incoming == "hello": This executes the next lot of instructions if the incoming message = hello
12. Type display.scroll("Hello World") this will scroll Hello World accross the micro:bit display
13. Type sleep(1000) this will pause the program for 1 second.

Python

```
from microbit import *
import radio
radio.on()
radio.config(channel=1)
while True:
    display.clear()
    if button_a.is_pressed():
        radio.send("hello")
    incoming = radio.receive()
    if incoming == "hello":
        display.scroll("Hello World")
    sleep(1000)
```

I/O Pins

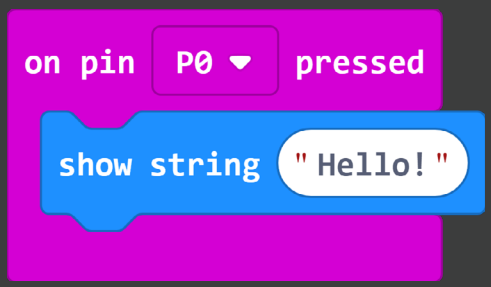
About the I/O Pins

At the bottom of your micro:bit is an array of pins, we call this the edge connector. You can use these pins to connect up components like LEDs and Buzzer or even slot it into an accessory like a robot. What you might not have known though is that there are a few pins on the micro:bit that can be used to detect resistive touch. In this section we'll take a look at how to detect a pin being touched and use it in a program.

Touch a pin

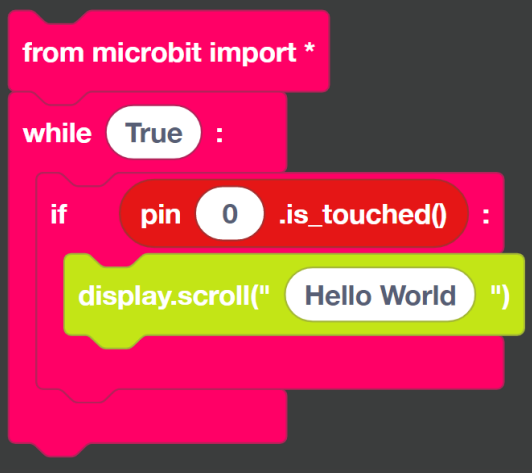
MakeCode

1. Open MakeCode within your favourite browser
2. Click on the on start block and drag it to the left and drop it in the bin. Do the same for the forever block
3. Click on Input Click and drag a on p0 pressed block to the code area and drop it
4. Click on Basic Click and drag a show string "Hello!" block to the code area and attach it within the on p0 pressed block
5. Plug your micro:bit into your computer using a microUSB cable.
6. Navigate to where your code downloaded to, This will normally be the downloads folder on your computer. Copy the file to the MICROBIT drive.



EduBlocks

1. Open EduBlocks within your favourite browser
2. Click on Basic Click and drag a **from microbit import *** block to the code area and drop it
3. Click on Basic Click and drag a while True: block to the code area and attach it under the **from microbit import *** block
4. Click on Basic Click and drag a if True: block to the coding area and attach it within the while True: block
5. Click on Pins Click and drag a pin 0.is_touched() block to the code area and attach it where it says True in the if block
6. Click on Display Click and drag a display.scroll("Hello World") block to the code area and attach it within the if pin 0.is_touched() block

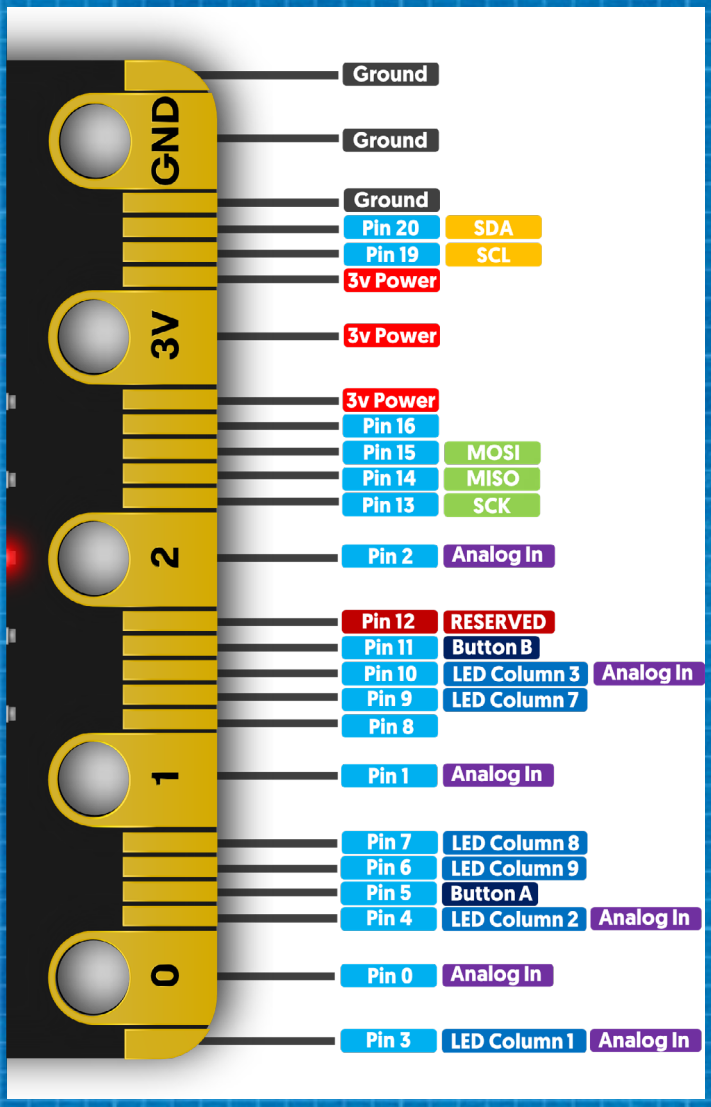


Python

1. Open your preferred method for programming Python.
2. Type from microbit import * this will import the micro:bit library to use with Python and press enter.
3. ype while True: this will create a while True loop meaning that the code inside it runs forever press enter.
4. Type if pin0.is_touched():. This will check if pi 0 is being pressed.
5. Type display.scroll("Hello World") which will scroll a message on the LED matrix.
6. Plug your micro:bit into your computer using a microUSB cable and flash the code.

```
from microbit import *

while True:
    if pin0.is_touched():
        display.scroll("Hello World")
```



Pinout Reference:

Each pin on the edge connector of your micro:bit has it's own individual job.

The image on the left is a pinout reference that shows which pin does what.

There are 25 pins on the micro:bit and the majority of these pins are I/O Pins but some of them are reserved for the LED matrix and some also serve as power and ground.

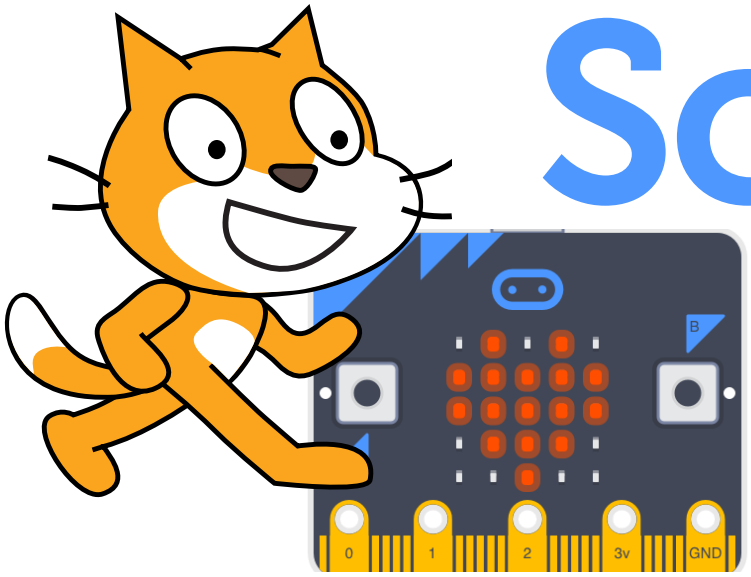
A Pinout chart is really handy when using the pins in your project as it serves as a reference guide so you know exactly which pin to use for each component.

We've create our very own pinout chart that you can download using the link below.

Also, if you buy a printed copy of Issue 7, you'll get one of these A3 posters FREE inside the magazine for you to put up on your wall.

If you'd like to get your own printed copy of the poster though, we'll also be selling them on the store for £1.50 each.

FREE DOWNLOAD:
go.micromag.cc/pin



Scratch Audio Game

This game tests your focus and reaction times by giving you spoken instructions for moving the micro:bit. How many moves can you get right?

BY Sean McManus

About the Author

Sean McManus is the author of Scratch Programming in Easy Steps (now updated for Scratch 3, including a new project for micro:bit) and Cool Scratch Projects in Easy Steps.

Twitter:

@musicandwords

Web:

www.sean.co.uk/scratch

Scratch 3 introduced some great new features, including micro:bit compatibility and Text to Speech. In this project, we'll combine them to make an audio game that tells you how to move your micro:bit and times how long you take. It's a game that tests your ability to focus.

I'm assuming some experience of Scratch here. If you're a newcomer, there's a beginners' introduction on my website at www.sean.co.uk/scratch.

Step 1: Set up the micro:bit

We explained how to set up your micro:bit in micro:mag #2 (see pages 22-24). Those instructions refer to the beta version, but you can use the normal version of Scratch now. If you don't have issue 2 handy, follow the instructions on the Scratch website at https://scratch.mit.edu/microbit.

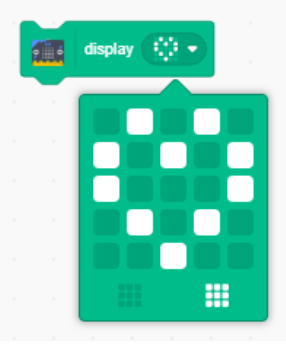
Step 2: Add the sound effects

We'll give each movement a different sound effect. Click the Sounds tab, and then click Choose a Sound in the bottom left. Delete the Meow sound and add six short sound effects. I chose Clown Honk, Siren Whistle, Big Boing, Bite, Bonk, and Pluck. Finally, add the sounds Drum Funky and Crowd Gasp as sounds 7 and 8 on the sprite.

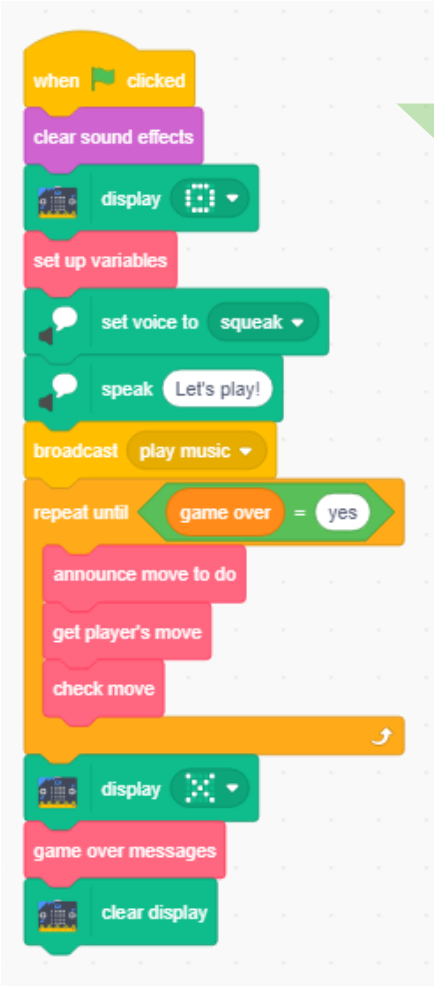
Step 3: Add the first script

We'll make our own blocks to make the project easier to read. Click the Code tab, then click My Blocks in the Blocks Palette, and click Make a Block. Create five new blocks called "set up variables", "announce move to do", "get player's move", "check move", and "game over messages". Use the Add Extension button in the bottom left of the Blocks Palette to add the Text to Speech extension. Click Variables in the Blocks Palette and make a variable called "game over".

Now you're ready to build the first listing. It shows a shape on the micro:bit LEDs using the **display** block, says "Let's play!" and begins the main game loop. You can use the **display** block to show any pattern of LEDs. Click the pattern in the block to redesign it.



Above: The micro:bit display block in scratch with the drawing dialog open



The Code

The block colours and any symbols on them help you find them in the Blocks Palette. Click the matching button to the left of the Blocks Palette to go to the right section.

Step 4: Set up variables

In the Variables, part of the Blocks Palette, use the Make a Variable button to create variables called "average response time", "move done", "move to do", "random direction", "score", and "total time". Make a list called "actions". Now add the second listing to your project. You'll need the **define** block that was created when you made the "set up variables" block in Step 3. It'll be in the Code Area. This script sets up the variables and list. The list is used to read the actions out, so beware of typos.

Step 5: Announce a move

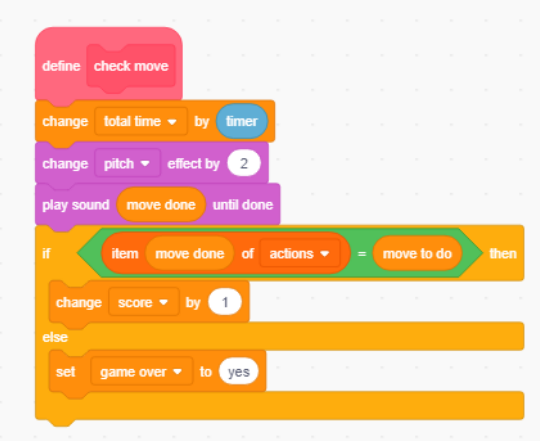
The next listing picks a random move number and then reads its name out. It also resets the timer, so the player's reaction time starts counting after the move is announced. Add this script now. Click the green flag and you should hear a rapid barrage of random instructions. Click the red stop button when you've heard enough.

Step 6: Get the player's move

The script to get the player's move checks for valid game movements until one is made. The move number the player makes is stored in the "move done" variable. It corresponds to the move name in the "actions" list (e.g. move 1 is "left button").

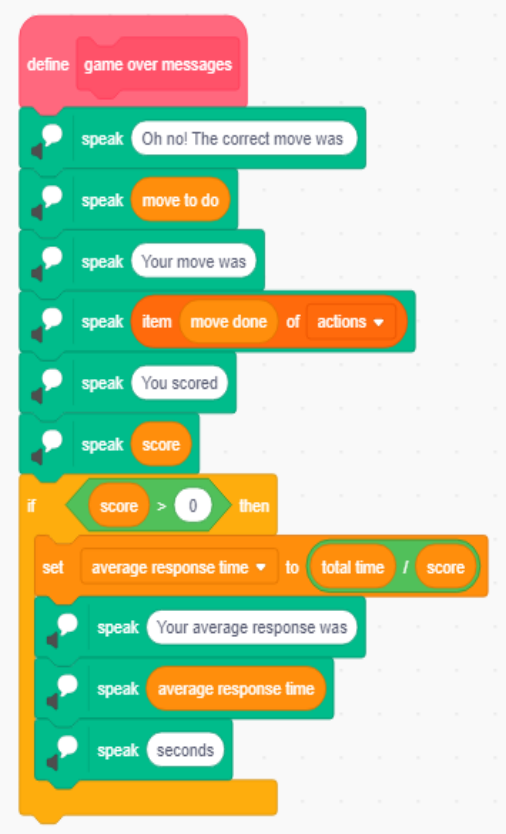
Step 7: Check the player's move

The **play sound until done** block is usually used with a sound name, but you can drop a number variable on it. I'm using it to play one of the first six sounds on the sprite, depending on the player's move, so each move has its own sound. I've used the new **change pitch** block to make the pitch higher each move, which adds tension. Each move, the time taken since the player's turn began is added to the "total time" variable. If the name of the player's move is the same as the name of the move the player needs to do, the score goes up. Otherwise, the "game over" variable is set to "yes" to exit the main game loop. Add this listing, and you can test the whole game flow now



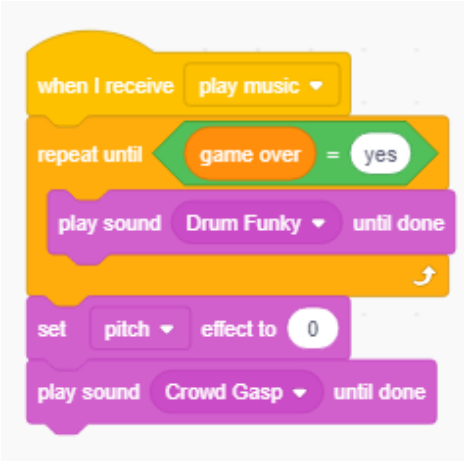
Step 8: Game over sequence

The "game over messages" script uses a number of **speak** blocks to tell the player how they did.



Step 9: Add background sound effects

The final script plays music during the game and plays the "crowd gasp" effect when the game ends. The background music is also affected by the changing pitch, so it seems to become more frantic as the game goes on.



Step 10: Refining the game

Now the game is ready to play! I set the minimum tilt angle at 30 degrees to make sure the tilt detected is intentional. When playing, you need to tilt until the move registers, then set the micro:bit flat again ready for the next move.

There are lots of things you can do to build on this game. You can change the mechanics so that players have to see how many they can get right in 30 seconds, or challenge them to memorise a sequence of moves to replay. You can add a script that displays a random arrow each move to try to confuse the player. What about rounding the average response time to two decimal places, so it's easier to understand when read out? Hint: you can treat numbers as strings and use blocks like **length of apple** and **letter 1 of apple** on them. There's a version of this game including the random arrows and the rounded result on my website.

Above:
I added a title screen to the game

You can access the project at:

go.micromag.cc/scratchaudiogame

micro:bit

Temperature with MicroPython

By Les Pounder

About the author

Les is a maker and trainer who has worked with the Raspberry Pi Foundation and the BBC to deliver computing training

@biglesp

bigl.es

You will need...

- BBC micro:bit
- USB Cable
- Mu (codewith.mu)

Quick Tip

When putting code inside of a loop in Python, we indent the code. Mu automatically indents when it detects you are doing this.

Did you know that the micro:bit has a built-in temperature sensor? It is part of the processor and using MicroPython we can unlock the power of the sensor and use it as an input in a project. In this project we will show a smiley face when it is warm, but a sad face when it gets too cold. For this project you will need to download and install Mu from codewith.mu.

Step 1 - Importing libraries

```
from microbit import *
```

To use the features of the micro:bit with MicroPython we need to import the microbit library. Using * means we can import the library and use the contents without starting every command with microbit.

Step 2 - Time Travel

```
import utime
```

To control the speed at which our code will run we need to import the utime library. This library has a function which can pause the running code.

Step 3 - Looping forever

```
while True:
```

To keep our code running while the micro:bit is powered up, a while True loop is used. This works in the same way as the forever block used in Makecode.

Step 4 - Temperature Check

```
    if temperature() > 20:
```

To keep our code running while the micro:bit is powered up, a while True loop is used. This works in the same way as the forever block used in Makecode.

Step 5 - Showing an image

```
        display.show(Image.Happy)
```

To show an image on the micro:bit's LED matrix we use the display.show() function and to show a happy face if the temperature is greater than 20C.

Step 6 - Pause for thought

```
        utime.sleep(1)
```

Using the sleep function from the utime library we instruct the micro:bit to wait for one second. This will show the face for one second.

Step 7 - What about cold?

```
    else:
```

If the temperature is not greater than 20C, we need to do something else, and to do that we use an else condition to catch any other scenario

Step 8 - Cold == Sad

```
        display.show(Image.Sad)
```

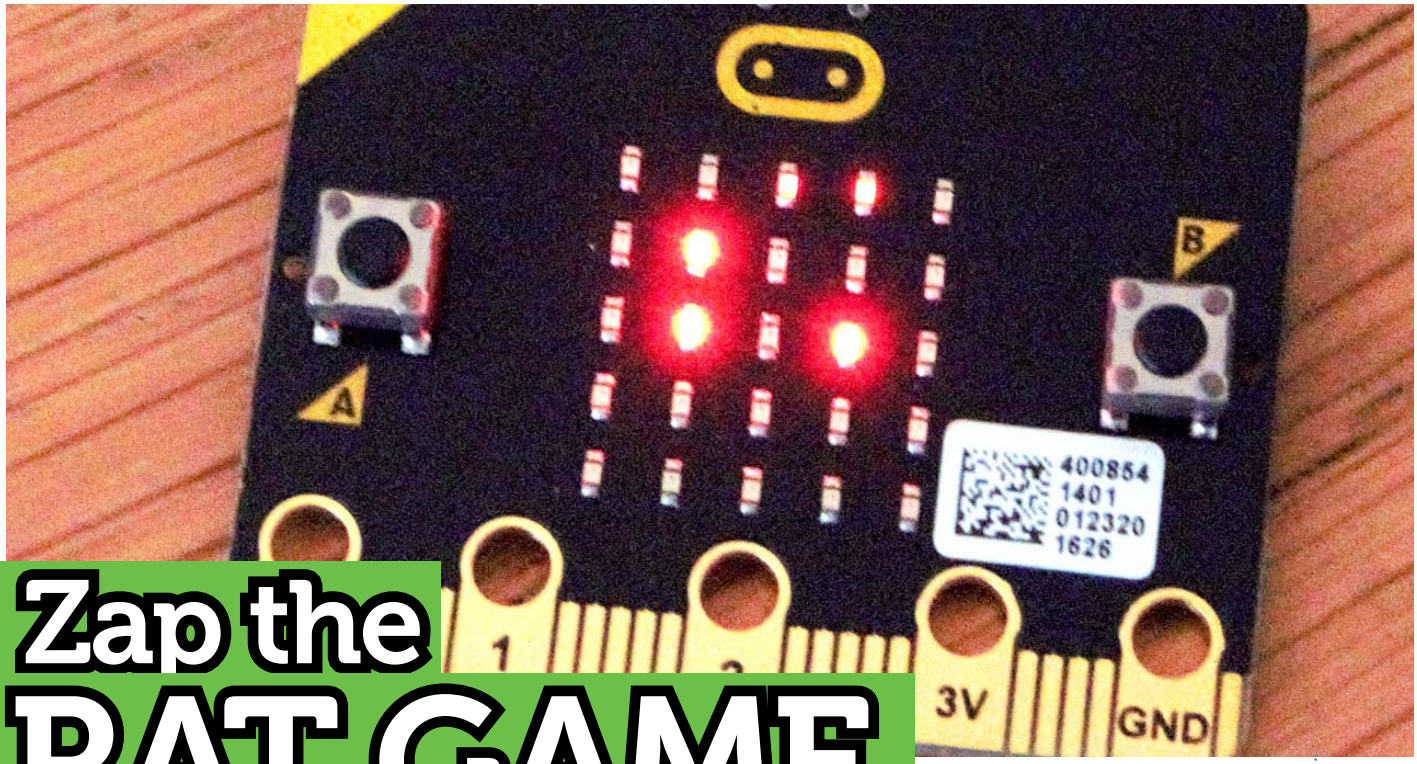
When the micro:bit is cold, we will show an unhappy face using the display.show() function. This can be changed to whatever you wish.

Step 9 - Another short pause

```
        utime.sleep(1)
```

To ensure that we can see the sad image on the micro:bit we need to add a short pause.

This will be the last line of code for the project, and now save the code to your computer as temperature control.py and then when ready click on Flash to send the code to the micro:bit. When finished, you should see an unhappy face. Touch the processor on the back of the micro:bit and the warmth of your finger will trigger the smiley face to appear.



Zap the RAT GAME

Above:
Zap the Rat in Action

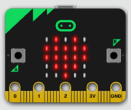
BY Tony Goodhew

Control pixels directly in Python and add a procedure to improve the game

About the Author

Tony is a retired Computing teacher who loves coding, travel and photography.

You will need...


micro:bit


USB Cable


Mu Editor

Is it possible to make a competitive game with just 25 LEDs and a single button?

A 'rat' runs around the edge of the board and as it passes between the lit LED and button B the player must momentarily press button B - WHILE THE LED IS LIT. If you manage to ZAP the rat a score LED lights up on the left-hand side of the display and the next rat runs faster. Each missed ZAP counts against you.

You have to Zap 3 rats to finish the game. Your score is then displayed. Zero is perfect!

The layout of the game

```
+ + + + +
+ S     +
A + S   M Z B
+ S     +
+ + + + +
```

+ = rat track
S = Score – rats zapped so far
Z = Zap position
M = Zap position marker LED
Rat moves around the track anti-clockwise

Entering the code

Tony suggests that you use the Mu editor to enter the script and then modify it. You can find the editor at https://codewith.mu/ . Tony uses this editor as it supports REPL (read–eval–print loop) and print statements, makes transferring your program to a micro:bit much simpler and it is easy to switch between modes (for micro:bit, Python 3, Adafruit CircuitPython and Pygame Zero).

To flash your script to the micro:bit you just click on the 'Flash' icon at the top of the editor window – no more drag and drop with 'hex' files!

```
from microbit import *
def grid(n): # Shows score of misses - Max = 25!
    display.clear()
    for y in range(5):
        for x in range(5):
            if n > 0:
                display.set_pixel(x, y, 7)
                n = n -1
def zero(): # Shows zero score
    display.clear()
    for zx in [1, 2, 3]: # Columns
        for zy in [1, 2, 3]: # Rows
```

```
            display.set_pixel(zx, zy, 9) # Pixel on
    display.set_pixel(2, 2, 0) # Clear center pixel

# Circuit LED coordinates - round the edge
LEDx = (0,0,0,0,0,1,2,3,4,4,4,4,4,3,2,1)
LEDy = (0,1,2,3,4,4,4,4,4,3,2,1,0,0,0,0)

while True:
    display.scroll("Zap the Rat")
    sleep(10)
    score = 0 # Rats zapped
    opps = 0 # Zap opportunities
    p = 0 # Position of rat on circuit (0-15)
    delay = 150 # initial delay between rat moves
    old_sw = 0 # Last value of switch B
    running = True # Loop control variable
    display.clear()
    display.set_pixel(3,2,9) # Show zap position
     while running:
        display.set_pixel(LEDx[p], LEDy[p], 5) # Show rat
        if p == 10: opps = opps + 1 # Incr opps = in (4,2)
        sleep(delay - score * 20) # wait - gets faster
        sw = button_b.is_pressed() # Read switch B
        pixel = display.get_pixel(4,2) # Rat in zap position?
        display.set_pixel(LEDx[p],LEDy[p], 0) # Hide rat
        if old_sw == 0 and sw == True and pixel == 5: # Hit?
            score = score + 1 # Increments hits
            display.set_pixel(1,score,9) # Show hits
            if score == 3: # Game over?
                running = False # Stop looping
            p = -1 # Position of next rat
        old_sw = sw # Score switch value
        p = p + 1 # Update rat position
        if p > 15: # Rat pointer out of range?
            p = 0 # reset to start position
    display.clear() # Game over - display opportunities
    opps = opps - 3 # Remove successful opportunities
    display.scroll("SCORE")
        if opps > 25: # Reduce missed opps to 25 max
        opps = 25
    if opps == 0:
        zero() # Zero missed opps - Well Done!
    else:
        grid(opps)
    sleep (5000)
    display.clear() # Tidy display
```

Controlling Pixels

This program illustrates the control of individual pixels in the LED display using (x, y) coordinates and brightness value.

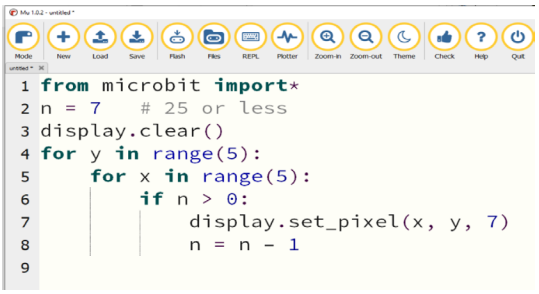
```
display.set_pixel(x, y, b)
```

where:

x is the column (0 to 4 – left to right)
y is the row (0 to 4 – top to bottom)
b is the brightness (0 to 9 with 0 = OFF and 9 is brightest)

```
# Light middle column, bottom pixel with medium brightness
display.set_pixel(2, 4, 5)
display.clear()
```

Try running this short script with different values for n, which are 25 or less.
This illuminates 7 pixels in the display and is used to show the score



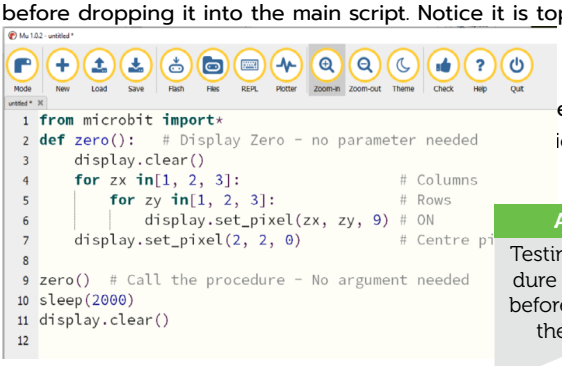
```
1 from microbit import*
2 n = 7    # 25 or less
3 display.clear()
4 for y in range(5):
5     for x in range(5):
6         if n > 0:
7             display.set_pixel(x, y, 7)
8             n = n - 1
9
```

Above: Script in Mu editor to light up pixels

in the game by including it, after testing, in a procedure called grid(n), at the top of the program. It is called by the main program in the third from last line with the parameter(n) replaced by the argument(opps). See:
go.micromag.cc/computer_parameters

Using Procedures

Writing and testing separately a short section of code before including it in a larger script as a procedure is a useful way of developing a larger project. This is how I tested the 'zero' procedure, before dropping it into the main script. Notice it is topped with



```
1 from microbit import*
2 def zero():   # Display Zero - no parameter needed
3     display.clear()
4     for zx in[1, 2, 3]:                  # Columns
5         for zy in[1, 2, 3]:              # Rows
6             display.set_pixel(zx, zy, 9) # ON
7     display.set_pixel(2, 2, 0)           # Centre pi
8
9 zero()  # Call the procedure - No argument needed
10 sleep(2000)
11 display.clear()
12
```

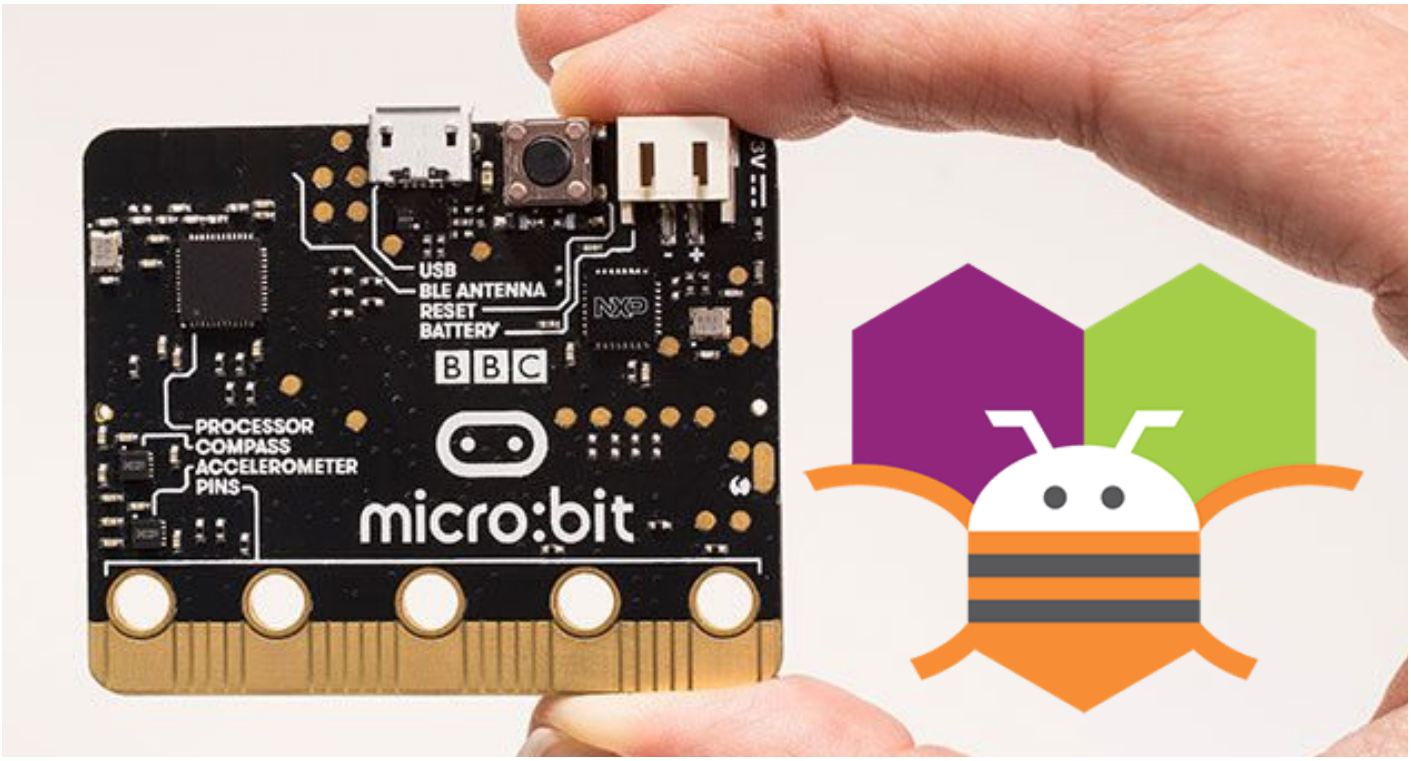
ed a few ions.

Above: Testing a procedure on its own before adding to the project

Extensions

Modify the basic game:

1. Make the 'rat' move faster (or slower) at the start.
2. Use button B – for left-handed players.
3. Make the rats run in the opposite direction.
4. Move the re-start position nearer the Zap position on each restart.
5. Make the 'rat' on the second and third runs move even faster.
6. Make an impressive '0' score display instead of zero(). Start with a single centre pixel, then a 3 x 3 square then a 5 x 5 square with the square growing and shrinking 4 times. (Develop as a procedure and then 'drop it' into the script once it works. def nought() with no need to pass it a value



MIT APP INVETOR + BBC micro:bit

BY Zayd Nashed

Learning how to connect and control a BBC micro:bit through MIT App Inventor

About the Author

Zayd is 11 years old.who loves to code and make projects with micro:bit.

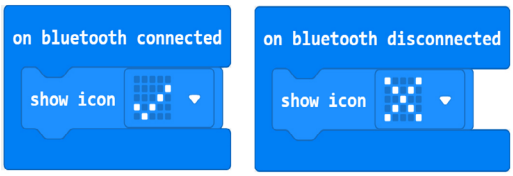
Zayd won the micro: bit Global Challenge for the MiddleEast region in 2018.

YouTube: youtube.com/channel/UCw6v84iDrpESkUmnQ6IpgiA

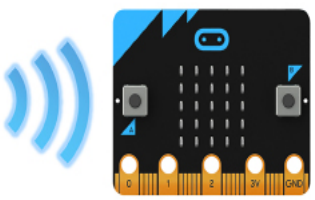
First of all, you will need to install the micro:bit app and the MIT App to your phone, so you can pair the micro:bit with the phone.

Next, open the micro:bit make code from https://makecode.microbit.org/#editor

Then, add the Bluetooth blocks extension and use on Bluetooth connect and disconnect blocks as below.



Then, open the micro:bit app and press connection then pair a new micro:bit after that it will guide you how to continue and what buttons to press on the micro:bit to enable Bluetooth mode.

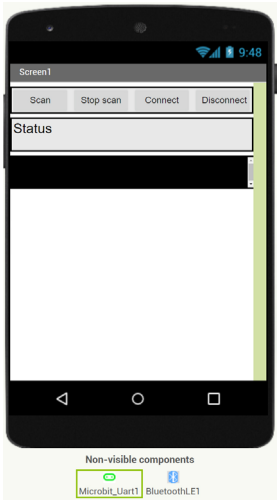


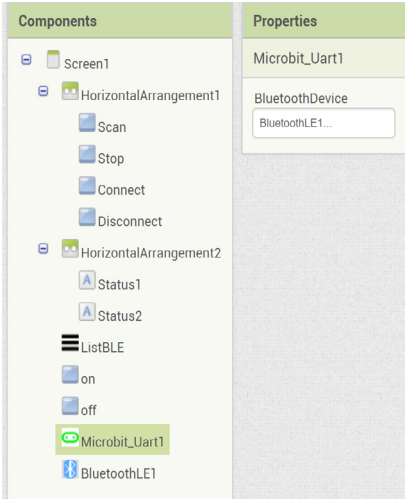
Note: Make sure Bluetooth is switched on on your phone

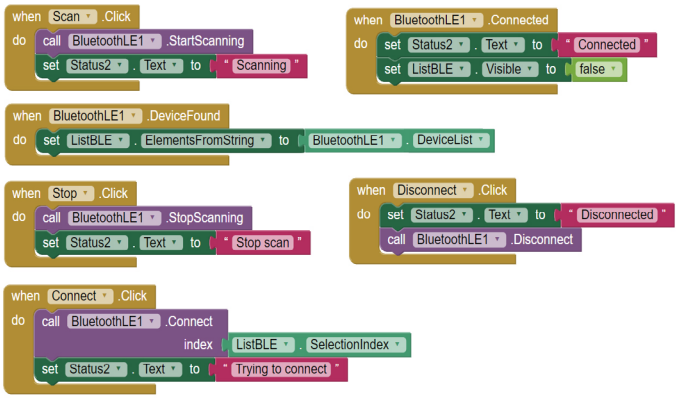
After that, open the MIT app inventor make code from appinventor.mit.edu/
First, you need to set up some buttons to scan and connect to micro:bit over Bluetooth.

- Drag a Horizontal Arrangement from the Layout drawer in the Palette and add 4 Buttons to it.
- Rename the buttons: Button Scan, Button Stop Scan, Button Connect, and Button Disconnect.
- Change their text to "Scan", "Stop Scan", "Connect", and Disconnect "from properties.
- Drag one more Horizontal Arrangement (HorizontalArrangement2)
- add a Label to HorizontalArrangement2, and rename it to LabelStatus1 and change its text to"Status ",
- add another Label to HorizontalArrangement2, rename it LabelStatus2 and remove the text
- Drag in a ListView below to show every single device it scans and rename it to List BLE

In addition, you need to add the micro:bit and the Bluetooth blocks extension to the MIT app.
Download the Bluetooth extension from:
http://iot.appinventor.mit.edu/assets/edu.mit.appinventor.ble.aix
Download the micro:bit extension from:
http://iot.appinventor.mit.edu/assets/resources/com.bbc.microbit.profile.aix
Don't forget after uploading the micro:bit's blocks extension drag the micro:bit UART and Bluetooth to the screen, and from properties click on Bluetooth devices and choose Bluetooth then switch to blocks and start to code the app.





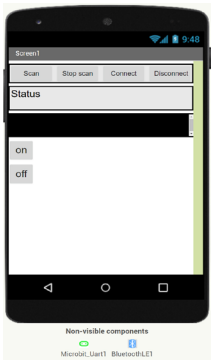


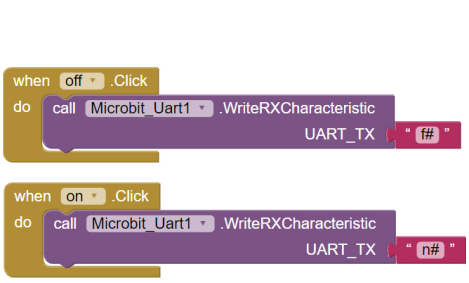
Finally, if you are done, download the code on MakeCode to the micro:bit, and from the MIT click on connect and then AI companion it will give you a code.

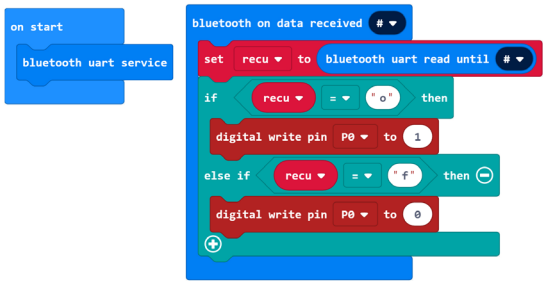
From your phone open the MIT app and then scan the barcode or write it down so it will open the applet on the way you designed it. Go ahead and test it!.

Here is a simple example of how to control LED connected to the micro:bit by using the MIT App.
Just add two more buttons to the MIT App code to switch the LED on and off.

In the MIT App's code, you need to use micro:bit blocks extension when the button is pressed send a letter with a sign like (#) after the letter, and it is compulsory to have that sign.







In the micro:bit's code you need to wire the LED, use the Bluetooth blocks extension and take the UART service to the on start and code it when it receives a letter with the # turn the LED on and the other way around.

Note: you can use voice recognition service from MIT App to control pins on the micro:bit after adding speech recognizer from Palette. Example: https://www.youtube.com/watch?v=Y6l3pxC-T9E
So, there you have your own smartphone application that can control your micro:bit project.

Simple Robotics Kit

Get started with robotics + micro:bit with this DIY cardboard robotics kit from Kitronik. By **Kerry Kidd**

AVAILABLE FROM
Kitronik

kitronik.co.uk/5665

What's Included?

- Motor Driver
- Gear Motors
- Wheels + Tyres
- Solderless converstion board
- Cardboard Chassis

Price

$36.95 USD
€33.38 EUR
£29.94 GBP
Approx

The new Simple Robotics Kit from Kitronik aims to be a grab and go DIY micro:bit robotics kit and has everything you need to get started with robotics and the micro:bit.

This is a great beginners robotics kit. We love that it comes with a nicely illustrated instructions book that takes us through building the robot to then creating code for the robot using MakeCode. This is a really nice touch rather compared to you having to download a PDF for the instructions and viewing them on a mobile device. There is just something about having a physical copy to work from that makes the kit feel really premium.

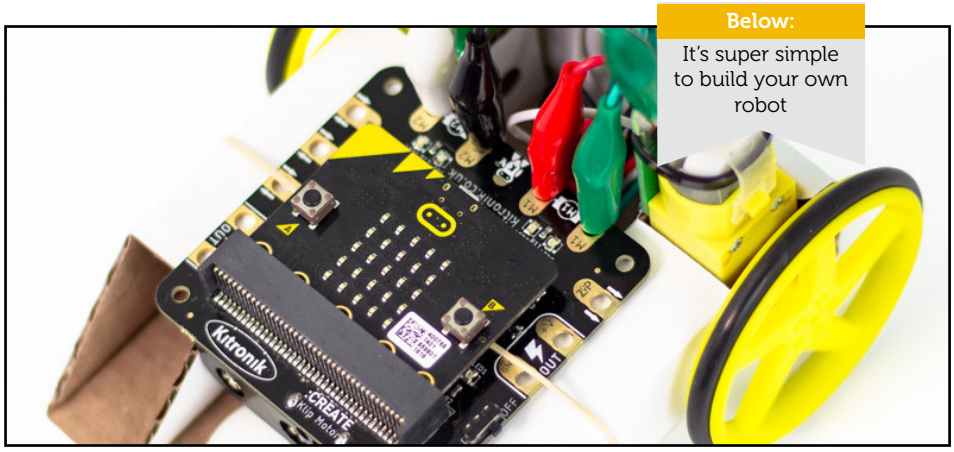
One reason we love this kit so much is it requires no soldering meaning it is great for kids to get started with robotics and also for schools and code clubs who due to risk assessments are forbidden to use soldering irons, something that prevents a lot of schools purchasing DIY robotics kits. One thing we would say is they may need adult help screwing the Solderless conversion kit on onto the motors as we found this could be a bit fiddly and had to push quite hard on the screwdriver.

Kitronik have thought of everything with this kit including putting an extra hole through the carboard between the motors so you can insert a pen and make the robot draw as well as including other pads on the motor controller so you can add some ZIP HEX boards to add some RGB (Red, Green, Blue) lights to your robot. As well as breaking some of the other pins out so you can add other sensors.

Overall this is a great little kit that comes with everything you need to get started all in one handy reusable container. The only downside with this kit is that it's a bit pricey at £30 and this may prevent schools from buying lots of them.

OUR RATING

8/10



Below: It's super simple to build your own robot

Similar Products

Two other products to consider...

01 :MOVE Mini MK2

Another "kit" robot from Kitronik. This one comes in at around £3 cheaper than the robotics kit.

go.micromag.cc/movemini

02 4Tronix MiniBit

Another "budget" robot but this time it's pre assembled so there is no need to build it.

go.micromag.cc/minib

Adafruit Clue

A micro:bit shaped device with an array of fancy features but not the micro:bit killer you may expect it to be. By **Joshua Lowe**

AVAILABLE FROM
Adafruit

adafruit.com/clue

What's Included?

- Adafruit Clue Board

Price (When released)

$39.95 USD
€35.66 EUR
£31.64 GBP

Approx

The BBC micro:bit is known for it's ease of use and out of the box experience (made even easier now with micro:bit classroom, check that out on page 4!) however a few have criticised it for not having fancy features like an LCD screen, larger memory to store our programs and an RGB neopixel. However, the BBC micro:bit doesn't need those features, it serves it purpose well and gets the job done. So this is where the Adafruit Clue comes in, a micro:bit shaped board which has all these fancy features (for a premium on top of the micro:bit price of course) and is even compatiable with micro:bit add-ons, which is the reason we are reviewing one. Let's take a look at some of the features.

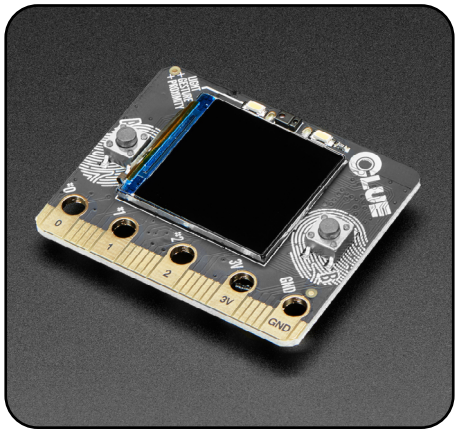
Onboard you will find:

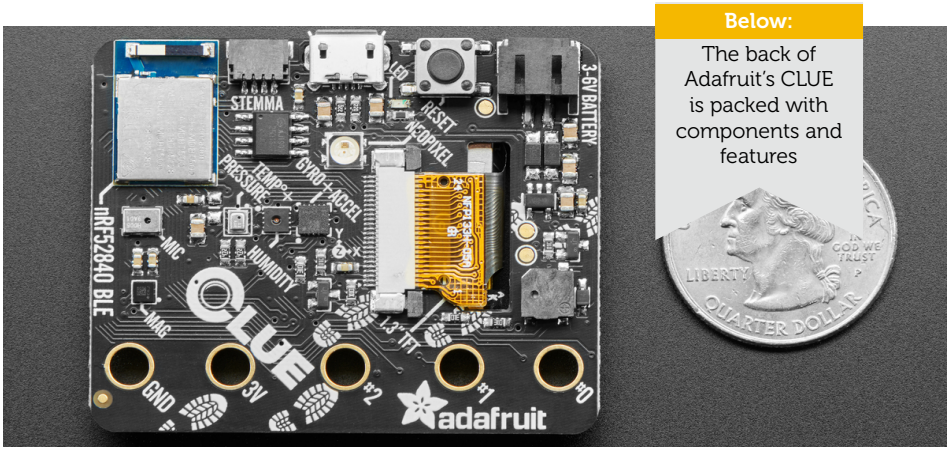
- Nordic nRF52840
- 1.3" IPS TFT Display
- Two A/B Buttons
- Accelerometer & Magentometer
- Proximity, Light, Color, and Gesture Sensor
- Microphone + Speaker
- RGB Neopixel
- 2MB internal flash

You can probably tell that's a lot of features packed into a small space. One of our favourite things is that the Adafruit Clue works with existing micro:bit add-ons due to Adafruit keeping the edge-connector pinout. We managed to get the Clue working with the 4tronix bit:bot XL and cube:bit using MicroPython without any issues and the experience was seamless. The Clue is currently in Alpha and currently doesn't support block based coding via MakeCode but Adafruit say this is on the roadmap. As for the price, we feel whilst it's more expensive than the micro:bit it's fair for what you get. Especially with the TFT Display, this brings a whole new visual element to your projects and is super simple to use, it can even display GIFs! After playing around with this board, it's awesome to note that it's not a micro:bit replacement and more a good "next step". We're excited to see where it goes!

OUR RATING

10/10

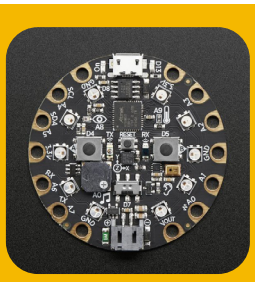




Below: The back of Adafruit's CLUE is packed with components and features

Similar Products

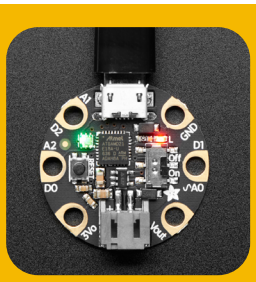
Some of Adafruit's other microcontrollers



01 Circuit Playground Express

A fun board that is perfect for wearable projects and beginners

go.micromag.cc/cpx



02 GEMMA M0

A super small low cost mircontroller that runs CircuitPython

go.micromag.cc/gemmam0

:KLEF Piano

Unlock your inner composer with this eye-catching musical add-on board from Kitronik. By **Kerry Kidd**.

AVAILABLE FROM
Kitronik
kitronik.co.uk/5631

What's Included?
- :KLEF Piano Board

Price
$24.60 USD
€22.27 EUR
£19.20 GBP
Approx

Unlock your inner composer with the beautifully designed :KLEF piano from Kitronik.

:KLEF is a very nice but very noisy add-on for your micro:bit. :KLEF is made up of 15 capacitive keys, an amplifier and speaker. 13 of these keys create a full octave of a piano and the other 2 are used as up and down buttons to change between different octaves.

The :KLEF can be programmed using MakeCode and MicroPython of which we're happy to say that Kitronik have provided libraries/extensions for boths of these which we are really happy to see as MicroPython libraries for products aren't common!

The :KLEF is very easy to attach to the micro:bit with the edge connector on the board. The capactive keys are responsive and it's easy to program in different octaves using the dedicated octave up/down keys on the left of the board.

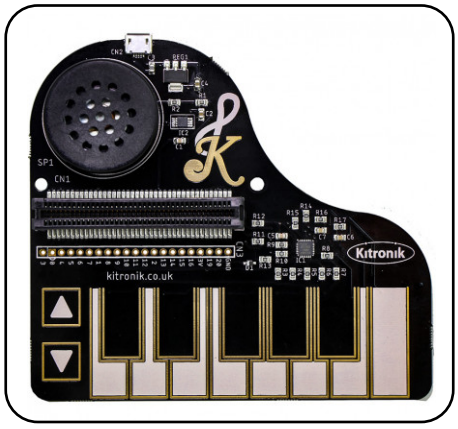
Coming in at £19, we feel that this board is worth the money as it provides something really fun and different. We really like the idea of the laser cut piano stand that Kitronik also includes on their website.

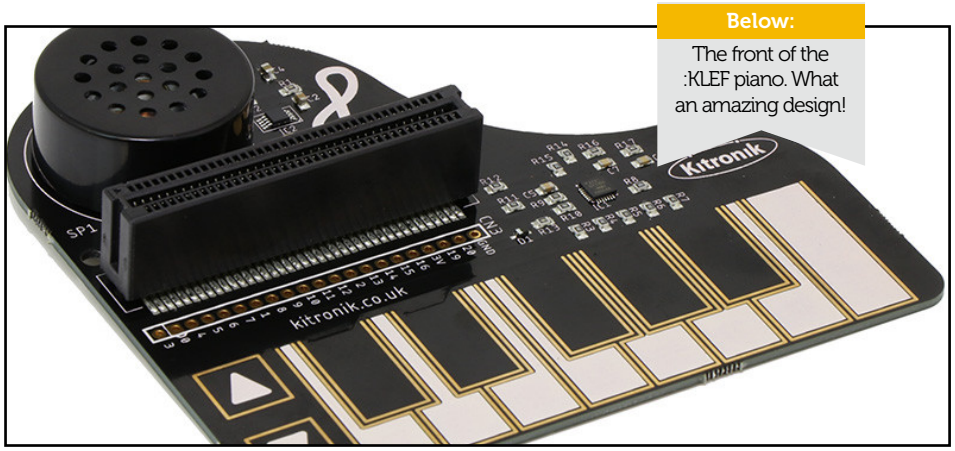
We had a lot of fun playing with the :KLEF piano and making lots of noise than playing anything useful, but if you know how to play the piano you may have lots of fun working out how to play proper songs and changing the octaves up and down.

The only criticism we have of the :KLEF piano is not having a headphone jack or a way to control the volume, which means if you are using these within a classroom setting it will get very noisy very quickly. Overall this is a fantastic looking board and a lot of fun to play with.

OUR RATING

8/10





Below: The front of the :KLEF piano. What an amazing design!

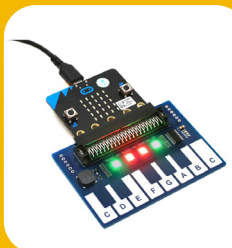
Similar Products

Two other products to consider...

01 Yahboom Piano

A similar sort of idea as the :KLEF piano but with a smaller speaker. This one comes in at $12.99

go.micromag.cc/yhpiano



02 Waveshare Piano

The waveshare piano is much less attractive than the :KLEF but features some RGB LEDs

go.micromag.cc/wspiano

Drive:Bit

Create your own robot or simply drive some motors with this Raspberry Pi Zero sized motor driver board. By **Joshua Lowe**

AVAILABLE FROM
4Tronix

go.micromag.cc/drivebit

What's Included?

- Drive:Bit Board

Price (When released)

$11.12 USD
€10.04 EUR
£9 GBP

Approx

Robotics is a really fun thing to do with the BBC micro:bit. However, the micro:bit can't drive these motors by itself so enter the 4Tronix drive:bit, probably my favourite motor driver board. Here's why.

Coming in at only £9, not only is it one of the cheapest micro;bit motor driver boards, it's probably the simplest and smallest but don't let that fool you, in fact, this is part of the reason why the Drive:Bit is do great, it does it's job for a small amount of money and it's really easy to use.

On the board you'll find some screw terminals for the power, two motor terminals, some servo pins, an RGB LED and some broken out pins for connecting other components.

To get started with this board all you have to do is slot in the BBC micro:bit, connect a battery pack (sold separately) and some motors and you've yourself a fully working setup.

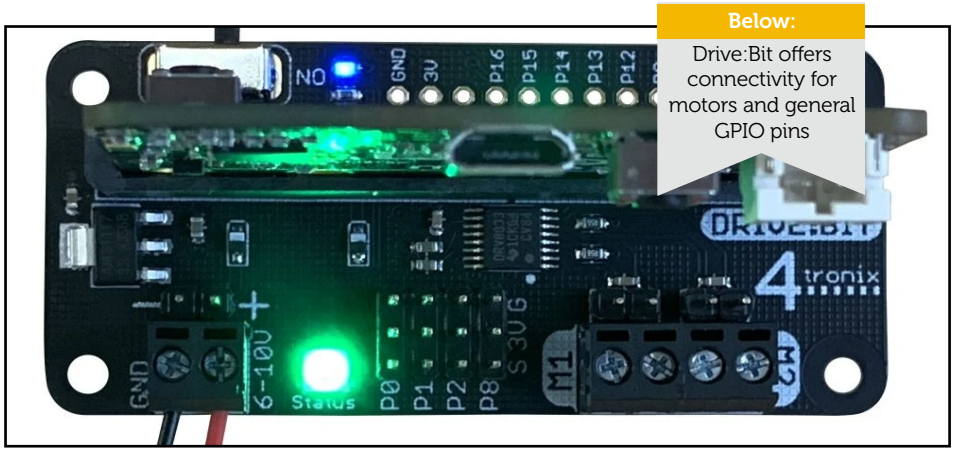
By using the extensive MakeCode library which is a fork of the extremely popular "Bit:Bot" library, it's really easy to get something up and running. The blocks are simple to use and are well laid out so you know exactly what you are doing. I got the kids at Code Club to have a go at using the Drive:Bit and they were able to have motors spinning within minutes.

The included blog post of the 4Tronix website covers everything you need to get started with the board including how to connect up the board, program it in MakeCode and MicroPython.

Overall, I am super impressed with this little board and think it is the go-to option for a compact motor driver board for the micro:bit.

OUR RATING

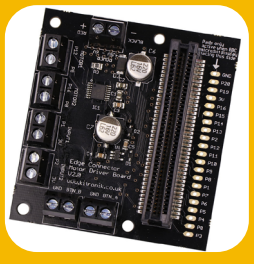
10/10



Below: Drive:Bit offers connectivity for motors and general GPIO pins

Similar Products

Two other products to consider...



01 Kitronik Motor Driver

One of the most popular motor driver boards on the market however it is much bigger than the drive:bit

go.micromag.cc/kitmd

02 Klip Motor Driver

Another motor driver from Kitronik that is based around a croc clip system for connecting motors

go.micromag.cc/klipmd

RETROSPECTIVE OF micro:bit LIVE

We didn't have any post-micro:bit live coverage in Issue 6 so here's a retrospective from a young persons point of view. By **Jay**

Hello, my name is Jay and I am eight years old. I am a Year 4 student and my passion is to conduct robotics and coding workshops all over the United Kingdom. Every first Monday of the month I conduct a robotics and coding workshops at UCLAN Preston.

In this article I will be talking about my experience at micro:bit Live 2019, an International festival which was held at the Media City in Manchester.

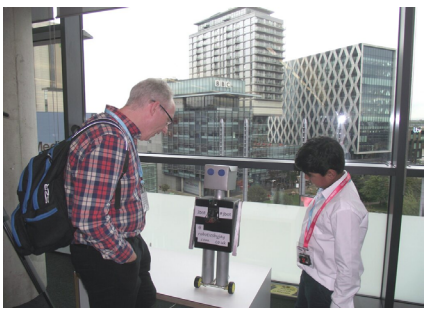
I was quite happy that I was the youngest delegate to be invited to deliver a talk at the international festival. At Media City there were a lot of buildings scattered around but we went into the main BBC building and we went on to the 5th floor.

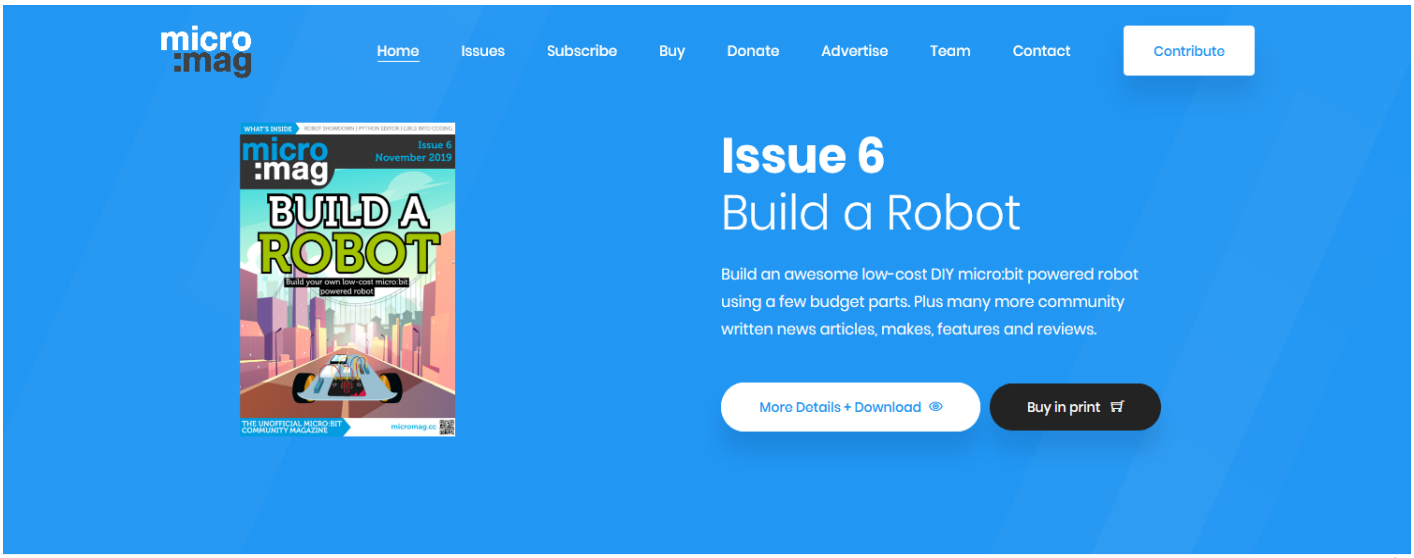
I showed my robot to delegates from all around the world for example China, India, Indonesia, Italy, Malaysia, Canada, Norway and many other countries. The robot I made is called J-bot which was made using a BBC micro:bit.

We listened to a few other talks from other delegates from all over the world. Lots of people took pictures of me and my robot and had a tremendous number of likes on my photos.

On Saturday I gave a lightning talk on the micro:bit at 3:00 pm which was making me a tiny bit nervous but nevertheless I found it to be a great experience and this speech in my opinion helped me build my confidence. I also attended an evening party where micro:bit bots made some interesting music. Thanks to Captain Credible for your amazing project!

Overall, I had an amazing experience. I love the micro:bit and have deliverd many workshops all over the country. Microbit is truly a versatile microprocessor board to learn coding and physical computing for children. I have used to the micro:bit to make many robotic projects as well.





Issue Catalogue

Team Update

ABOVE: Our brand new website that's now live!

Welcome to the first "Final :bit" where we aim to cover stuff like updates about the magazine or a special guest piece that doesn't fit into one of the other categories. For this issue, we'd like to update you on all things micro:mag.

New Website

You may have noticed that our website, micromag.cc, looks a little different than it did a few months ago. We have been planning a complete redesign for a while now as the old website using Wordpress was not fit for purpose and made it difficult to download the magazine due to it not scaling well on mobile devices. After looking into many options we decided to go the static website route. We ended up with something that we can change to suit our needs easily and fits in with our colour scheme and branding and we think it looks great, we hope you agree!

Print + Free Poster

The print issues have been a huge success so we're doing it yet again for Issue 7, however, this time we're including something special. Inside every print copy is a Glossy A3 micro:bit Pinout Reference Poster for you to put up on your wall. It has vibrant colours and a section that shows you how to use the pins in both MakeCode and MicroPython. If you want to buy a poster and not a magazine or want another poster with your magazine, we'll be selling them separately for £1.50 excluding delivery. We hope to do more offers like this in the future so stay tuned and thank you to all of those who are supporting us by buying print copies as this helps fund what we do and make the magazine better for all.

Call for contributions

Since October of last year, we have been laying the ground work for a "Getting Started with micro:bit" user guide book that we want to release in 2020 and we want your help! If you'd like to write an article focused on someone starting out with the micro:bit, we'd like you to get in touch with us via email. You can reach out to us at hello@micromag.cc. We look forward to hearing about your ideas.

Next Issue

Our Next Issue's cover feature will be a beginners guide to creating your own basic add-on board showing the steps all the way from design to having a working PCB. Obviously, this is subject to change due to the current issues with COVID-19 (Coronavirus) and how easy it will be for us to get PCBSs from China.

micromag.cc

£5.99