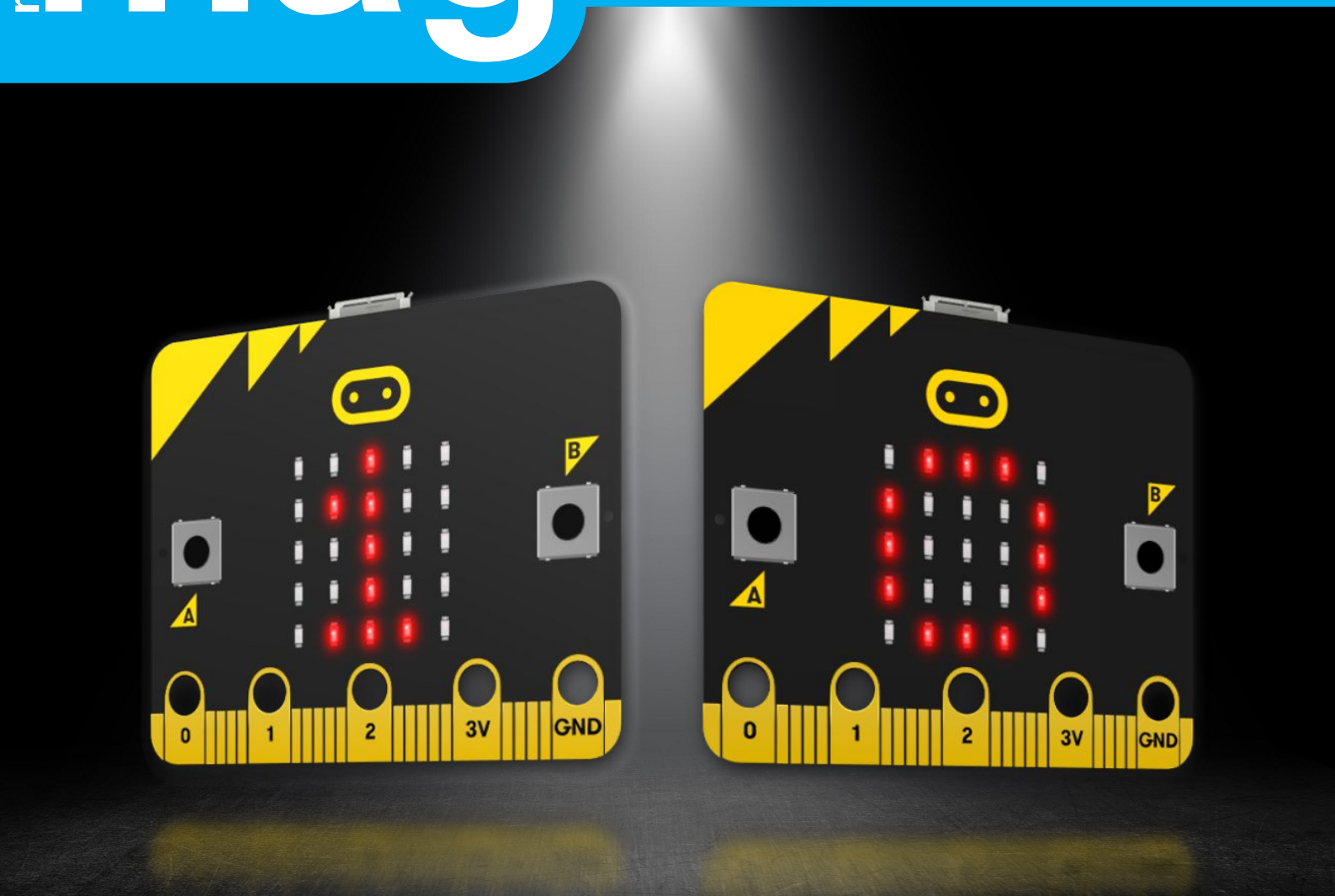
The unofficial micro:bit community magazine

micro:mag

Issue 3
January 2019

micromag.cc
@micro_mag



AMAZING micro:bit Projects

We showcase some of our most loved micro:bit projects

In This **Issue**

Visit us at BETT 2019
ExCel London, 23-26 January

We're starting 2019 with our biggest and best BETT yet.

Join us from 23 - 26 January at ExCel London, in our new spot in the new 21 Century Skills area.

BETT is a chance to participate in an exciting time in education and technology - whether you're a Teacher, an Educator, a Student, or a fan of all-things-micro:bit.

We're running a full schedule of fun and interactive talks and workshops each day, with all kinds of innovators from the micro:bit community. There will be plenty of great opportunities to get hands-on with micro:bits and get coding in minutes.

BETT is ticked but free to visit. Register for your tickets now: bettshow.com

micro:bit

Welcome to Issue 3

Hi

Welcome to issue 3 of micro:mag! We have another fun packed issue for you today!

We have been overwhelmed once again by the number of contributions that we have received from the community. A huge thank you to every one of you, without you creating micro:mag would not be possible!

As well as all the usual community contributions we have a feature article showcasing 10 amazing micro:bit projects by people within the community.

Additionally, we have the micro:bit news section, tutorials, feature articles, the next installment of meeting the micro:bit Foundation as well as a competition for you to win a zbit:speaker add-on for the micro:bit (more details on page 6).

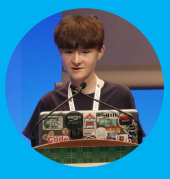
In the next issue, we will be celebrating our 1st Birthday! We cannot believe that we came up with micro:mag nearly a year ago and the support we have received from the community has been outstanding. Thank you so much for your continued patronage.

So, without further ado: grab a seat and a brew and get stuck in to issue 3!

Kerry Kidd
Editor in Chief

Meet the team

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



Joshua Lowe is a young coder, creator of the Edublocks tool for micro:bit and has done lots of workshops around the world.

Archie Roques makes lots of different things, from circuit boards to tables. Some of them even work!

James Bastone runs a local Makerspace and joined the team because he genuinely enjoys proofreading articles.

Team

Editor in Chief
Kerry Kidd

Editorial team
Joshua Lowe
Archie Roques
James Bastone

Contributors

Martin Woolley
Jacqueline Russell
Sam Watson
Rebecca Keough
Alan Yorinks
Carlos Pereira Atencio
Diego Fonstad
Daniel Perks
Pradeeka Seneviratne
Tanya Fish
Eun Jung (EJ) Park
Michael Rimicans
Jackie Pease
Justaboutfine
Chris Penn
Mr ZBit
Les Pounder
Jonny Austin
Matt Hillsdon
Mark Williams
Ross Lowe
Joe Finney
Sam Kent
Kitronik

Contact us:

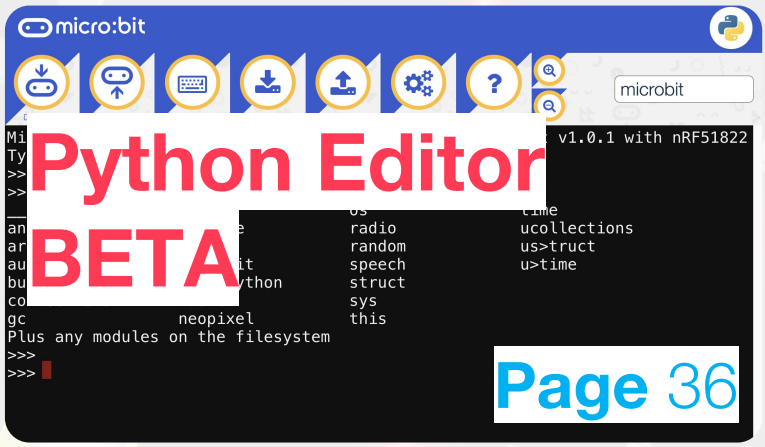
Website: micromag.cc

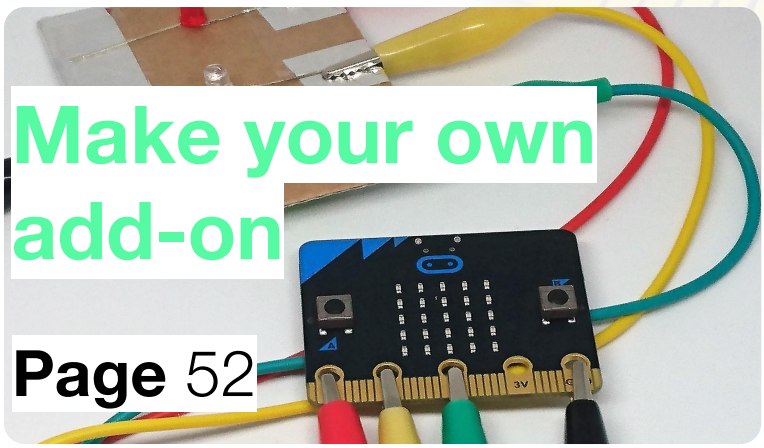
Email: hello@micromag.cc

Twitter: @micro_mag

micro:mag

Issue 3: Contents





:news

Cover Feature

:feature

:make

More Articles

:review

micro:mag

Competition

Win a zbit:speaker!

With zbit:connect, we are giving our readers the chance to win a zbit:speaker add-on board! Judged by the micro:mag team.

How to win

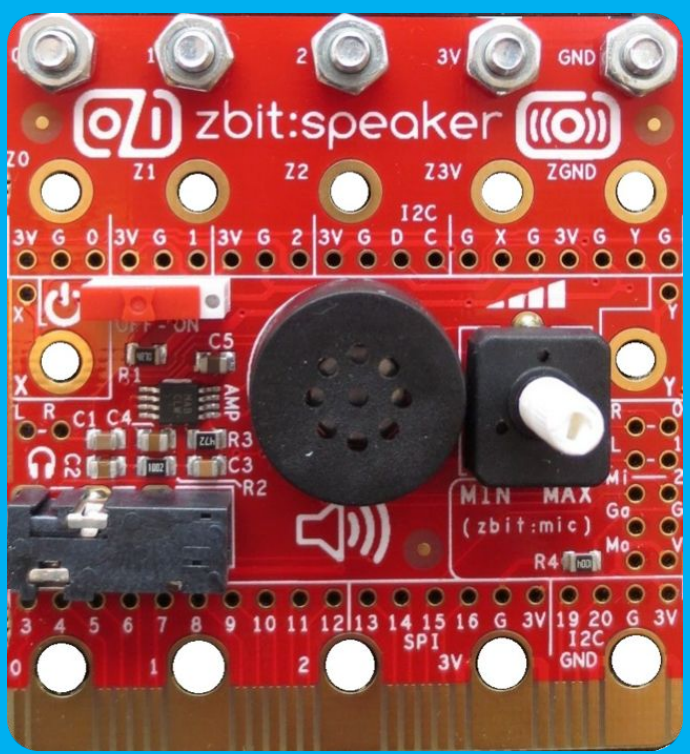
We've got 2 boards up for grabs with 2 categories (UK Only):

1. **Help us tell the world about micro:mag!**

Use the speech module in MicroPython to get your micro:bit to talk about micro:mag. Post a video on twitter using #micromagspeech by February 9th 2019 to enter.

2. **Make a cool music project!**

Use the music module in MakeCode or MicroPython to build a micro:bit project. This has to be a new project and not one that you've done in the past. Post a video on twitter using the #micromagmusic by February 9th 2019 to enter.



Rules + T&C's

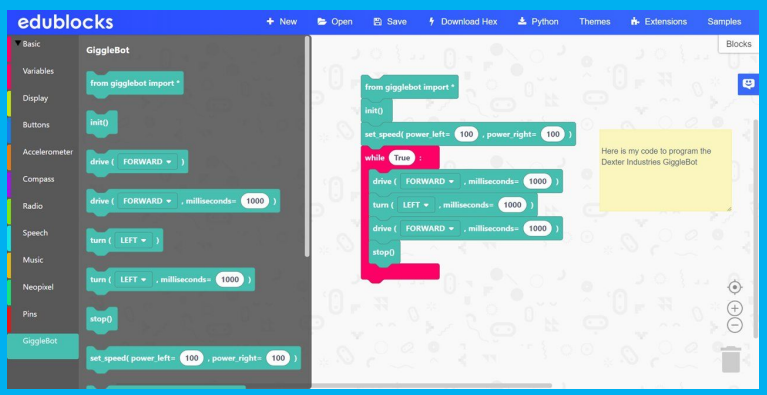
As with any competition, it is important to read the Rules + T&C's which can be found here:

micromag.cc/comp-rules

EduBlocks micro:bit BETA

A refresh of the EduBlocks for micro:bit editor is here in BETA for you to try!

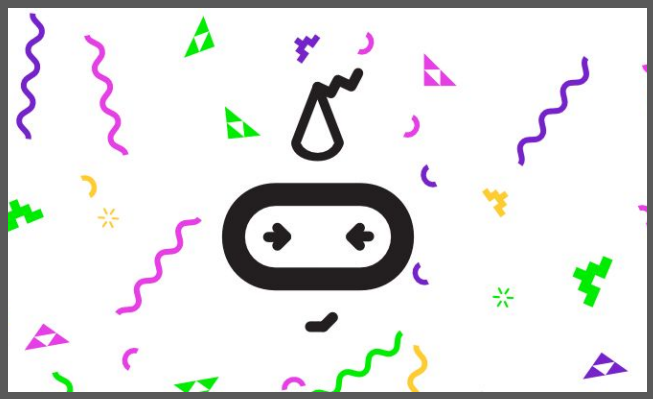
The popular tool that helps people make the transition from block based programming to Python has just announced that an updated version is now available in BETA for people to try out before it goes fully live. This also gives people the chance to update their resources if they need to do so. The main update in this new version is the switch to the Scratch 3 block style which we is in the brand new MakeCode update that has just gone live. Try it out today!



beta.microbit.edublocks.org

Global Challenge Winners!

If you cast your mind back to Issue 2, we advertised the micro:bit Global Challenge. After hundreds of entries and rigorous judging, the Micro:bit Educational Foundation have announced the winners of the challenge. With winners from Europe, Asia, Africa, North America, Middle East & Latin America, there are a whole host of amazing projects that contribute towards the Global Goals and all of them are definitely worthy winners. Check out the winners at the link.



go.micromag.cc/gcwin

UK Scratch Conf 2019!

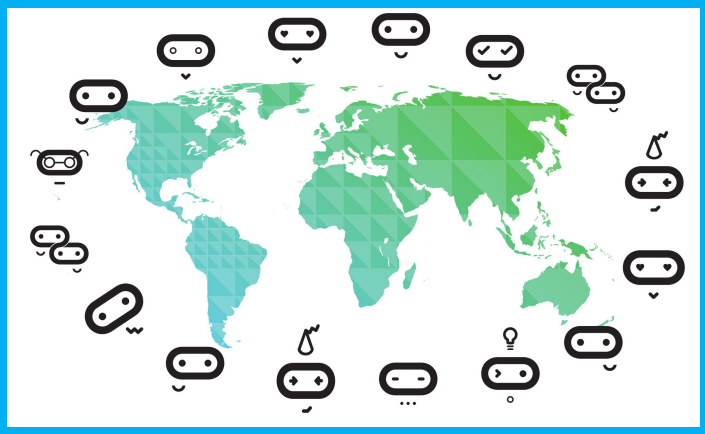
The Raspberry Pi foundation have announced a UK Scratch Conference!

Whilst not direct micro;bit news, we thought this was worth a share. The Raspberry Pi foundation have announced that the Scratch Conference is coming to the UK for the first time! The event will be held at Churchill college in Cambridge on the 23rd-25th August 2019. They have not released tickets yet for this event, however, a call for "content creators" has been launched. So, if you'd like to do a talk, run a workshop, show off a project or hold a discussion session to do with Scratch you can do so now. We would love to see lots of Scratch powered micro:bit projects here too! Learn more using the link.

go.micromag.cc/s-conf

Looking for a job? Micro:bit are Hiring!

The awesome Micro:bit Educational Foundation are hiring for developers! There are currently 2 roles open, Senior Front-end Developer - Freelance (9 -12 months position) & Fullstack Developer - 12 month fixed term position. The foundation are a great bunch of people and if you are looking for developer work and are passionate about getting kids coding then these are perfect for you. Plus, the whole foundation are remote, so apply now!

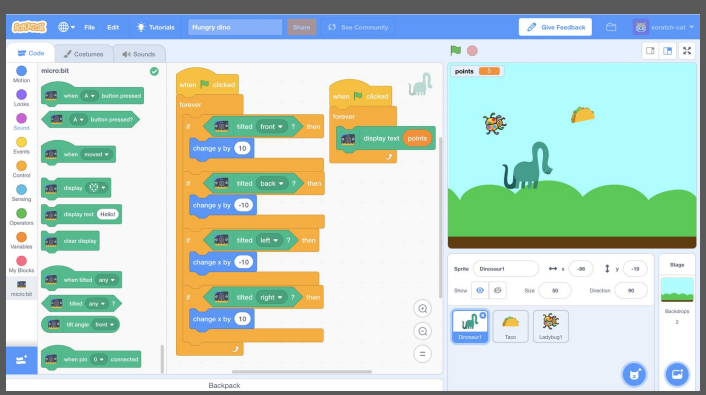


go.micromag.cc/workatmb

Scratch 3.0 is now Live!

Scratch 2 has been retired and replaced with Scratch 3.0

This month, we said bye bye to the flash version of Scratch, version 2 and said hello to the new and improved Scratch 3.0. You may remember we had a whole cover feature that had lots of information and tutorials about Scratch 3.0 in Issue 2. The Micro:bit Education Foundation have produced some amazing coding cards for you to get started with Scratch 3.0 and the micro:bit. There are also some handy links to other resources too, check them out with the link.



go.micromag.cc/mbscratch

Announcing BittyWeb from Bitty Software

Martin Woolley

I love to code and make magic things happen with Bluetooth wireless communications and was lucky enough to be involved in creating the BBC micro:bit as the team's Bluetooth specialist.

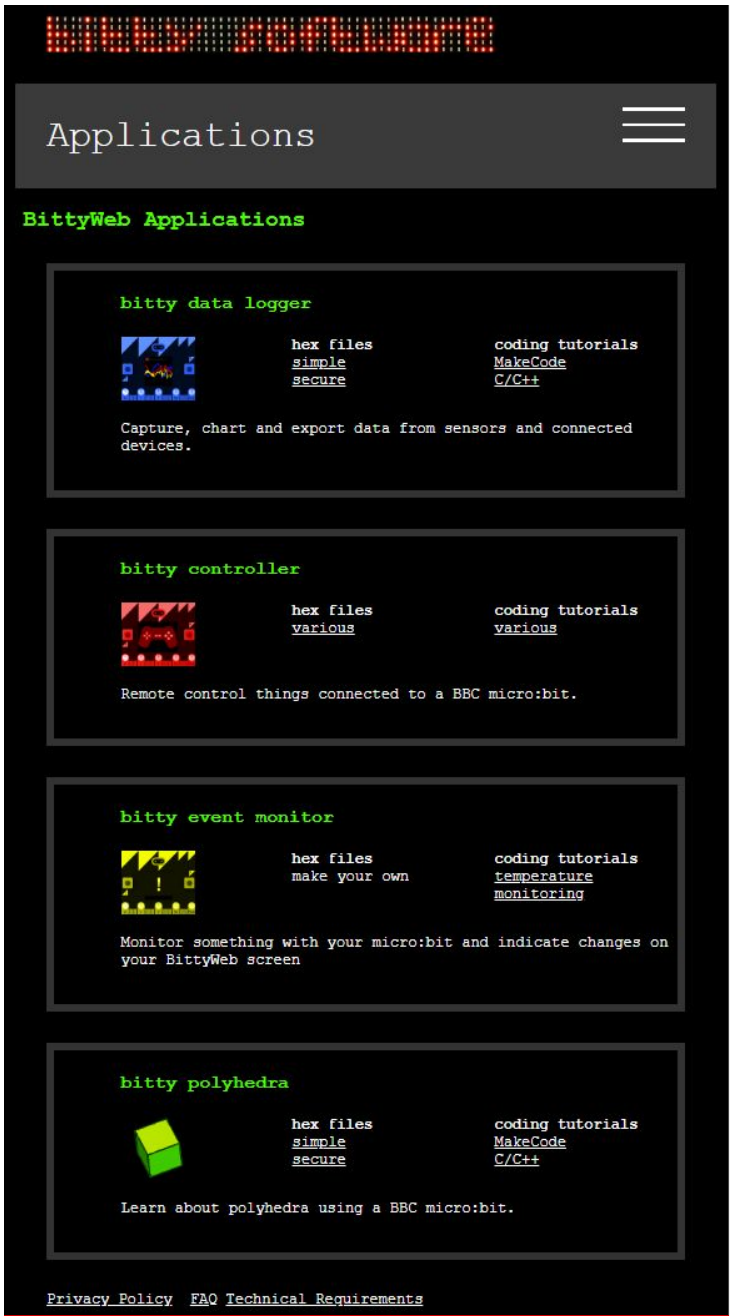
@bittysoftware

bittysoftware.com

Create the most inclusive and immersive STEM lessons for the whole class with BittyWeb

The Bitty Software mobile applications for the micro:bit have proved popular with teachers and STEM ambassadors, allowing invisible physical phenomena to be visualised and analysed using Bitty Data Logger, robots to be programmed and remote controlled using Bitty Controller and coding exercises to be undertaken. The nature of mobile applications and the policy many schools have on the use of smartphones in class has meant that these applications were only really available to the adults in the room, typically teachers. That has all changed with the introduction of BittyWeb.

BittyWeb makes the most popular Bitty Software applications available on PCs running Windows 10, Linux, Mac OS, Chrome OS and Android devices, in all cases from within a web browser. Yes, all you need is the Google Chrome web browser running on a PC, laptop, tablet or smartphone which has support for Bluetooth 4.0 or later and you are all set to run some of the most immersive and inclusive science, technology and coding classes ever!



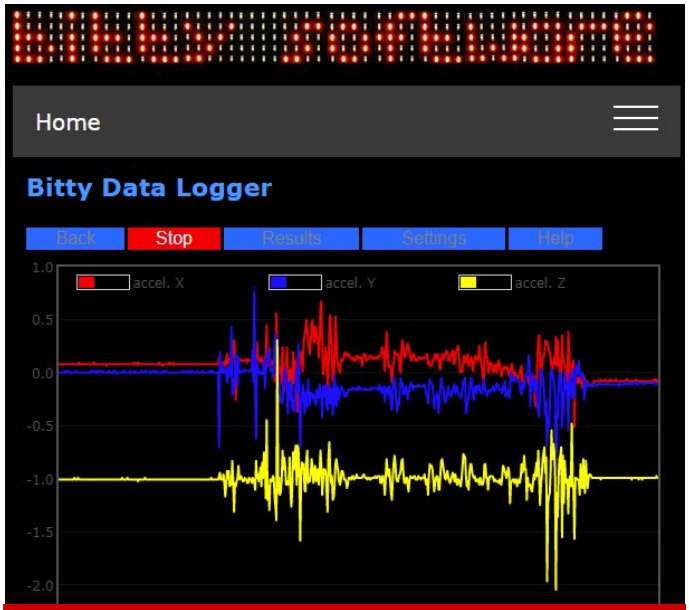
The BittyWeb applications

BittyWeb from Bitty Software :news

To use BittyWeb, organisations need to register for an account and subscribe to use the system for a certain duration (e.g. 12 weeks) and with support for a given maximum number of concurrent users (e.g. 30 for a whole class). Registering will result in an organisation account being created and a user account for the member of staff who will act as the system administrator for their organisation. That user can then log in and create as many standard user accounts as they are likely to need, typically one for each member of the class. Standard user accounts are completely anonymous, by the way.

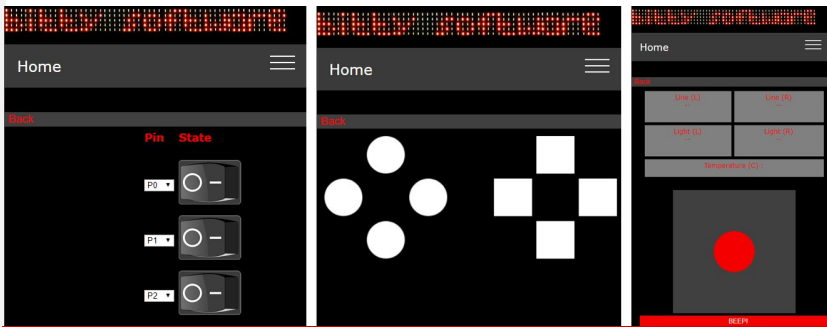
BittyWeb currently offers 4 micro:bit applications including the popular Bitty Data Logger and Bitty Controller applications and two new applications, Bitty Polyhedra and Bitty Event Monitor.

Bitty Data Logger allows data from the micro:bit's internal sensors, including its accelerometer, magnetometer and temperature sensor to be captured and charted in real time. You can also capture data from things connected to the micro:bit edge connector like the C02 sensor that was featured in issue 2 of micro:mag. Data once captured, can be exported for use in other applications like spreadsheets.



Bitty Data Logger

Bitty Controller allows you to use your web browser to control micro:bit connected machines. A variety of user interfaces are available, so you should find something suitable for your type of machine. A classic game controller is perfect for driving vehicles but there is also an analogue controller which has a trackball you can push in any direction. This gives fine control of movement in any direction and at a speed which depends on how far from the centre you push the trackball. Finally, a collection of on/off switches allow you to switch on and off whatever you have connected to the micro:bit.

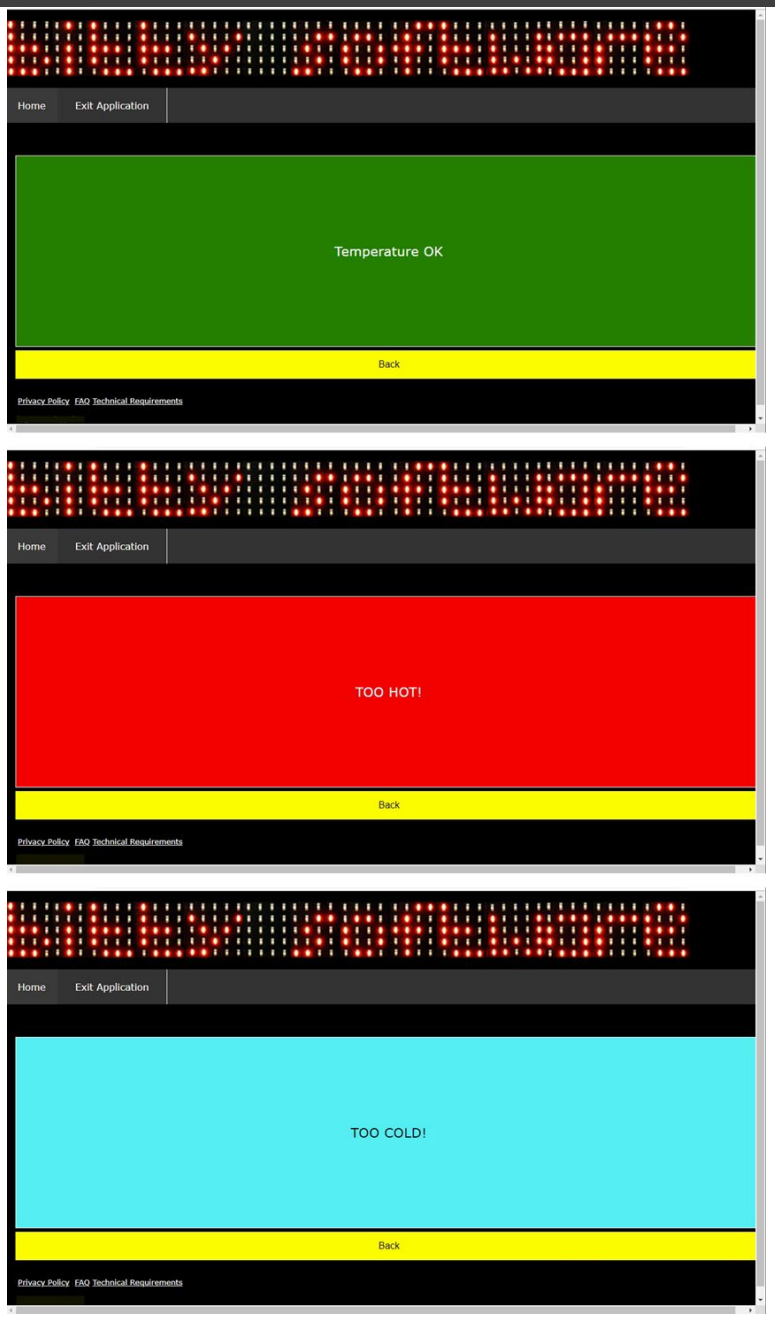


Bitty Controller

Bitty Polyhedra allows 3D shapes like tetrahedrons, octahedron, cubes and prisms to be reviewed and rotated in three dimensions, controlled by gestures

made with a micro:bit. You get to see the selected shape from all angles and really get to know it. Bitty Polyhedra will help teach 3D geometry.

Monitor anything you like using your micro:bit and report significant events to Bitty Event Monitor and it will indicate whatever is going on with the colour and text that you choose. Suitable for a huge range of classroom and maker projects, you could monitor temperature bands, atmospheric pressure ranges or motion, for example. If you can measure it on a micro:bit using the right code, you can report it to Bitty Event Monitor for it to indicate visually on your screen.



Further BittyWeb applications are planned though and we are always open to ideas.

Perfect for class and group projects using the micro:bit, BittyWeb is available now. Sign Up for an account today!

MakeCode for the micro:bit Update

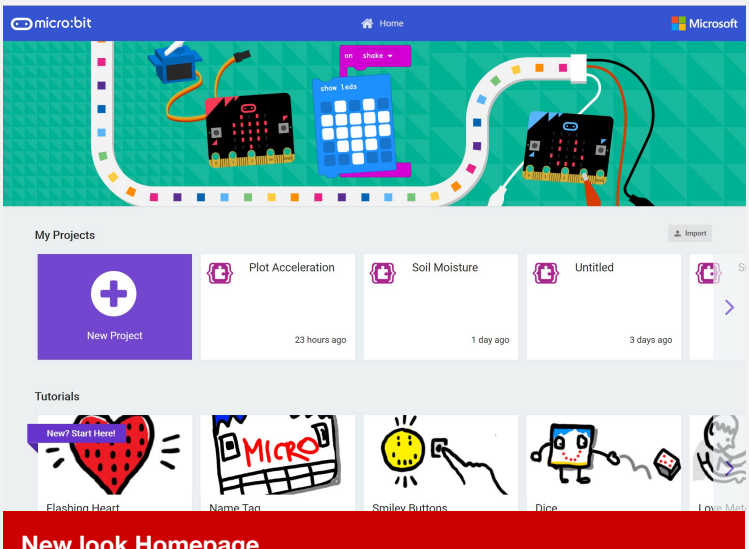
Uncovering some new features of MakeCode

Many of you have probably noticed a recent change in the MakeCode micro:bit code editor. The core principle behind this latest update was to include only incremental changes and improvements that would not be disruptive and would not break any existing programs.

In this article, we'll cover some of the newest features you'll see in this update:

Home Page

We've tried to make the getting-started experience more intuitive by creating a set of step-by-step tutorials and making the example projects more discoverable. We moved all the content that was in the Projects menu before into this new homepage experience.

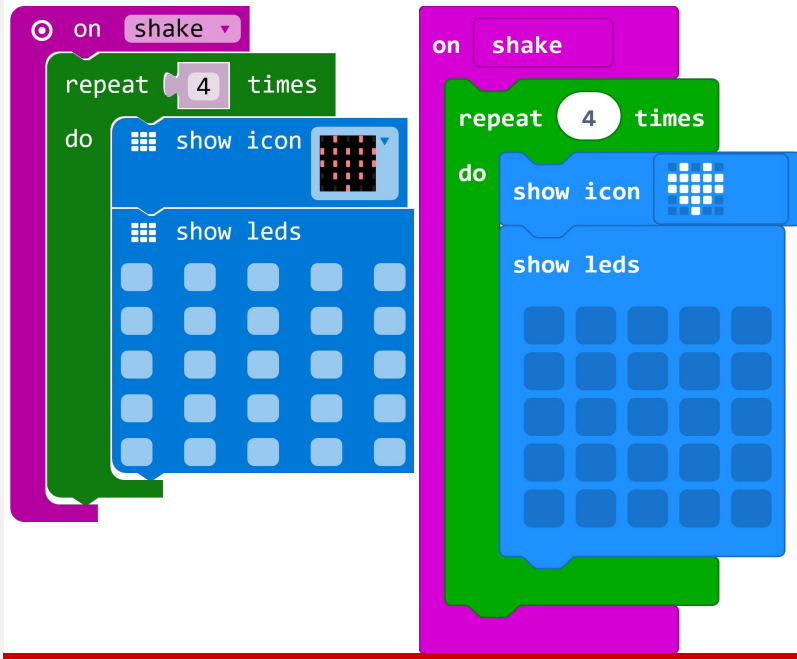


New look Homepage

Block UI update

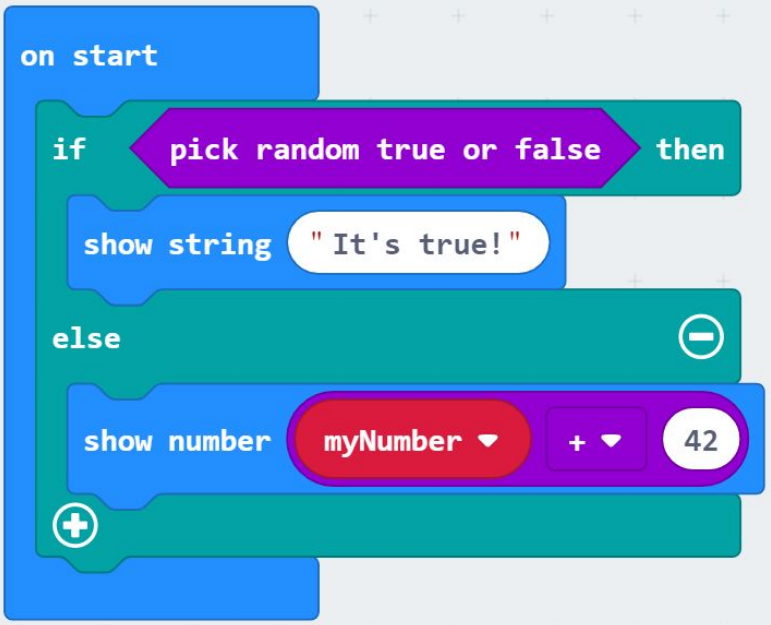
You may notice that our blocks look slightly different. To align with our other editors, we've upgraded from Blockly to the new Scratch Blocks UI (which is a combination of Blockly & Scratch). There are some nice improvements with this new rendering, notably:

- Bigger blocks which make it easier for users with touchscreen devices to drag and drop with their finger, and more efficient use of block space.



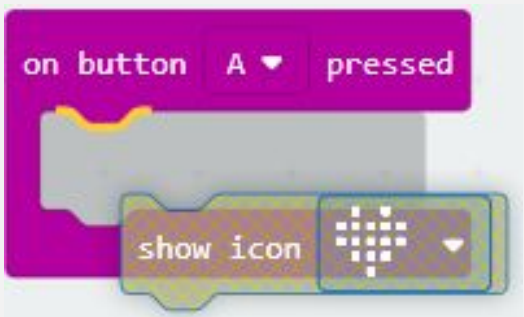
Old block Style New block Style

- Different shapes for different data types – specifically boolean values (true/false) are hexagon shapes, and numbers and text are round .



Showing the new data type block shapes

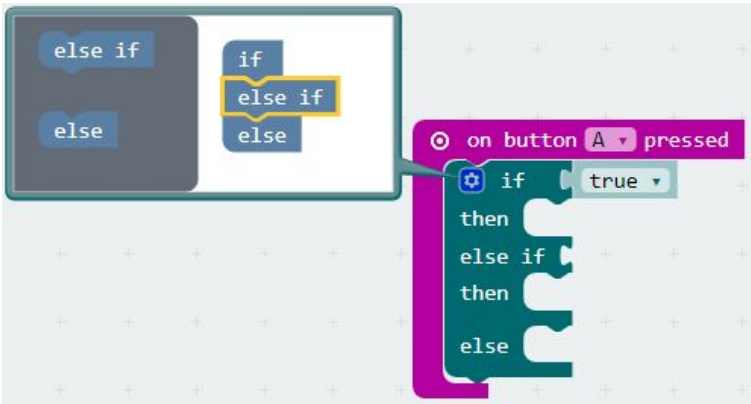
- A better indication of where blocks fit together, and better snapping into place



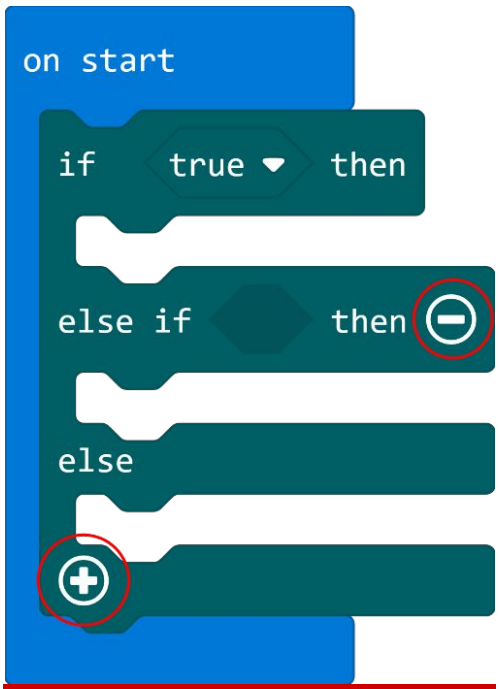
Showing how the blocks connect together

Cogwheel RIP

We got out the shovels, said a few words, and then had a little party when the cogwheel died. Most likely you used the cogwheel in your **If Then Else** blocks to add additional clauses. It was an awkward interface that most people didn't know how to use. We've replaced this functionality with more intuitive plus (+) or minus (-) icons on the blocks to add or delete clauses.



Old style blocks with cogwheel

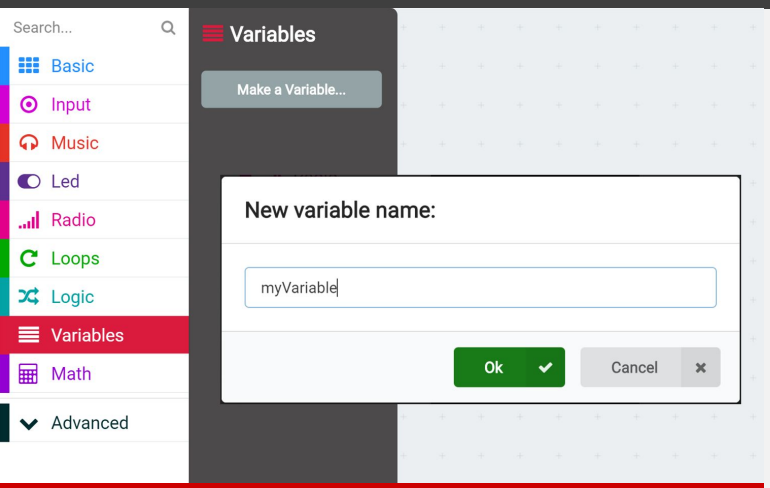


New style blocks with the plus (+) and minus (-)

Make a Variable

Based on feedback, we tried to make the process of creating variables clearer. Now, you will explicitly click on the "Make A Variable" button in the Toolbox to name and create a new variable and the associated variable blocks.

micro:bit MakeCode Update :news



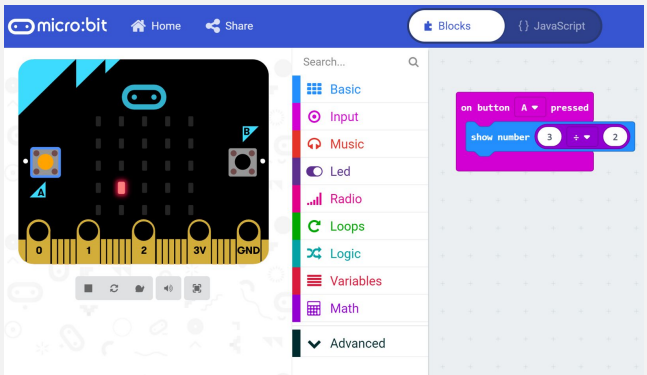
Showing how to create a variable

Radio blocks

We made some slight changes to the Radio API's to make them simpler. Most people won't notice, but if you are an advanced Radio user, you may see that we've reorganized the way we receive radio packets – same functionality is there, but made more explicit through single-use blocks.

Floating point

This is probably the biggest change we've made for this update. All you Math Teachers out there, prepare to rejoice... we now support floating-point arithmetic! This was a big request – as many of you know, we only had support for integers before. But now 3 ÷ 2 really equals 1.5 (instead of 1)!



Showing floating points numbers working

A big **THANK YOU!** to all the folks who have helped with testing, logging bugs, and filing requests – your feedback has been critical in helping us stabilize and deliver a highly performant, reliable platform.

Happy Making and Coding!

The MakeCode Team

Microsoft MakeCode
Hands-on computing education

Scouts Digital Maker Day at Bradley Wood Activity Centre

Sam Watson

Sam lives in Huddersfield and enjoys coding in his spare time. Sam takes great interest in the micro:bit and raspberry pi and goes to the Huddersfield Library Code Club.

The Raspberry Pi Foundation partnered up with Scouts to create a Digital Maker Badge. I was lucky enough to go and help one of the leaders run part of the event.

I recently went along to a Digital Maker day at the Jubilee Centre at Bradley Woods Activity Centre in Huddersfield. Scouts have been associated with Bradley Woods since 1925! I was asked to help because I go to the Huddersfield Code Club, and also because I am a Scout. Michael (the volunteer who runs the code club) asked me to help Beavers, Cubs and Scouts who were attending the event use the BBC Microbit and Raspberry Pi.

The morning

I arrived at Bradley Woods Jubilee Centre and found Michael, who showed me around the different rooms and also the main hall where people had brought various technology along to show the Scouts.

Using micro:bit & pi-tops

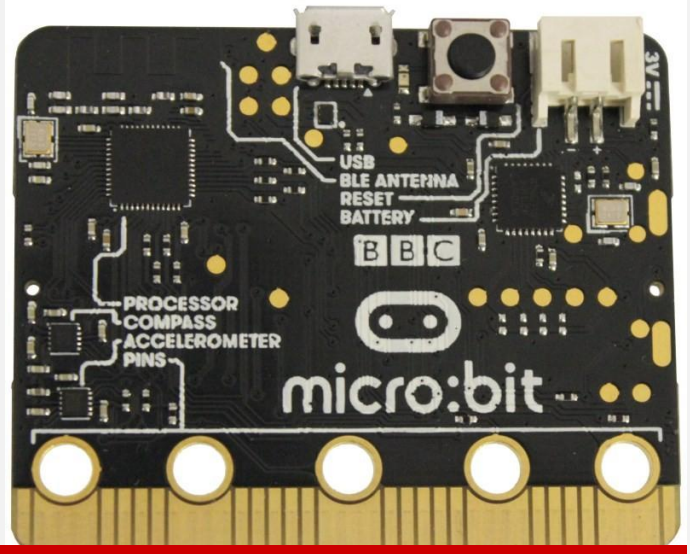
The Jubilee Centre at Bradley Wood where the Digital Maker day took place.

First, a group of Scouts arrived and Michael (the leader) explained what they were going to be doing with the micro:bit and Raspberry Pi. As I specialise with the micro:bit I helped the Scouts that had chosen to use them. The first task was to create an interactive badge. If you would like to try this for yourself then you can find the link to the instructions for the code here. Then some Scouts moved on to a project called 'Rate your mates', which can be found here. Others experimented further with their interactive badge. Meanwhile, another group were using Thonny Python to hack Minecraft Pi edition.

Scouts Digital Maker Day at Bradley Wood Activity Centre

They were creating giant areas of different kinds of blocks including TNT which they would then blow up!

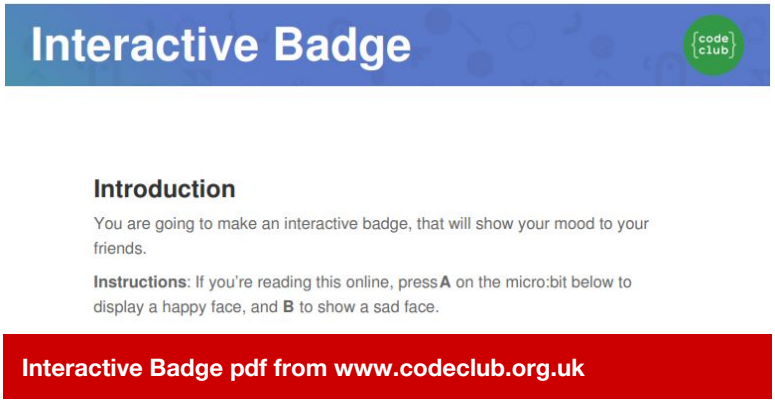
As well as Scouts, Cubs and Beavers using the micro:bits and Pi Tops in the room I was in, downstairs they were using stickman animation, looking at robotics, and 3d printed items. During the morning we had 3 groups of Scouts visit the micro:bit room, I showed them all how to use the blocks and download the code to the micro:bit itself. Most of this group were extremely knowledgeable about coding and once shown were able to complete the task easily. Then we all had a break for lunch.



BBC Microbit used on the Digital Maker Day to create an Interactive Badge, or to do Rate your Mates.

The afternoon

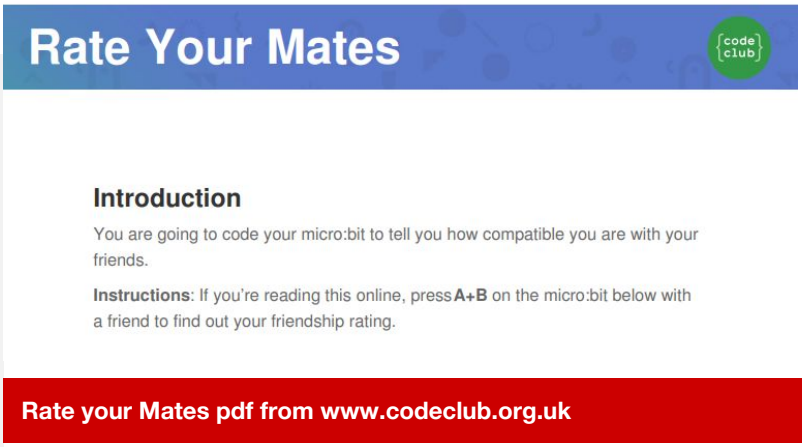
Then in the afternoon, we had some Beavers and Cubs come in too. They needed more help to get going with the micro:bits as they were younger. For the majority it was their first time coding – some had not even heard of coding before! For these children I found out if there was anyone who knew a little about it sat next to or near someone who didn't know about it so that they could help them. As these children were less knowledgeable about coding, they spent longer exploring the Interactive Badge.



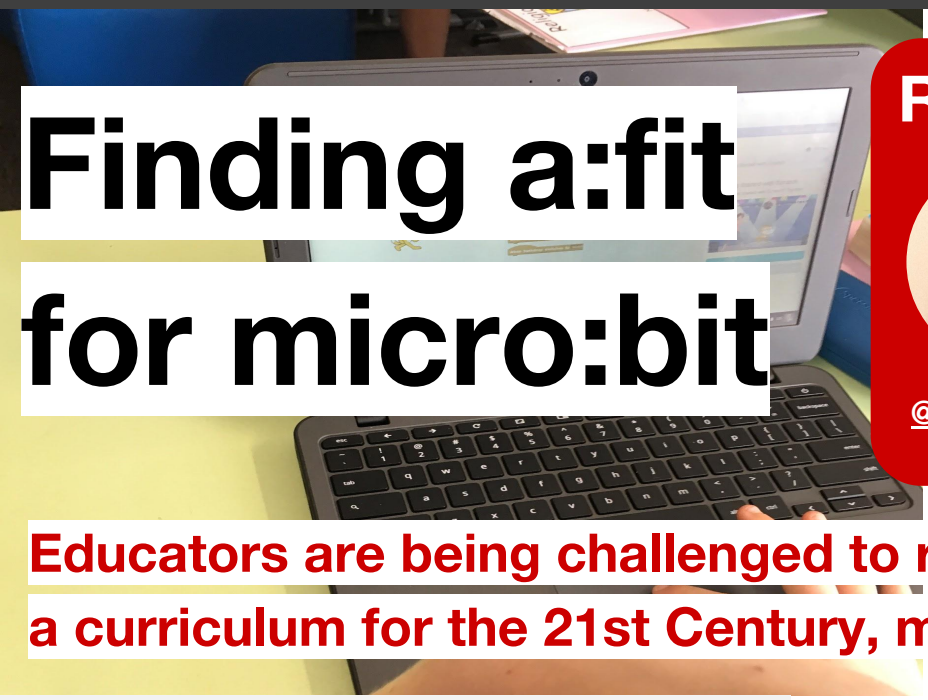
Interactive Badge pdf from www.codeclub.org.uk

Badge Presentation

At the end of the day all the Scouts, Cubs and Beavers went into the hall and were presented with their Digital Maker badge. A Scout Leader thanked everyone for coming and then I was asked to stand up and go to the front of the hall to collect my badge. After that, I spoke to the man who had the 3D printers to find out the best one. Finally, I helped Michael pack away all the equipment that we had been using. I had a really brilliant and amazing time helping others to learn to code, it was a really great day! I hope you enjoy using the Microbit to do either the Interactive Badge or Rate your Mates.



Rate your Mates pdf from www.codeclub.org.uk



Finding a:fit for micro:bit

Rebecca Keough

Rebecca Keough (B.GS/T & MEd G&T) is a Classroom Teacher and Coordinator in NSW Australia. Passionate about inquiry learning, PBL & HPC.

@BeckKeough1

Educators are being challenged to reinvent how they deliver a curriculum for the 21st Century, micro:bit might just fit

For innovation in education to occur here in Australia, it will require the re-augmentation of current practice and the diversification of lessons to include technology in an integrated and authentic manner. Avoiding the stock standard 'silo' approach to teaching and learning is of paramount importance in a time where we work with a crowded curriculum that demands authentic, 21st Century pedagogies.

However, the expectations on Australian Teachers to now teach the new Digital Curriculum as at 2019 has a lot of educators feeling very uneasy and some feeling conflicted as they face a future of working alongside their students and moving away from practices where the teacher is the expert.

Location & Learning

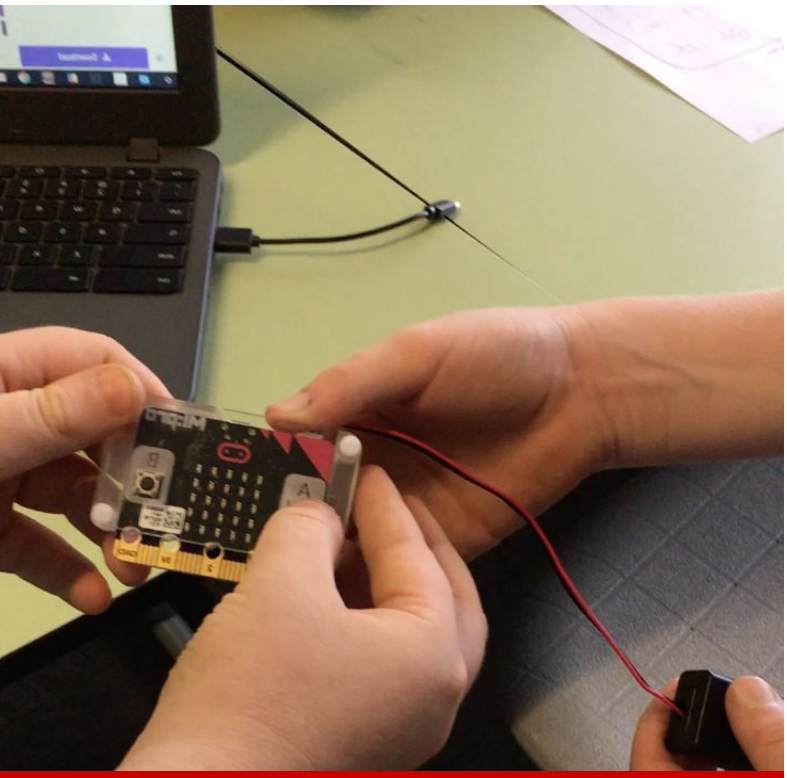
Our school is located in a rural area and we are at the commencement of our journey with utilizing technology at this new level of requirement. This is something that we realise will challenge us, yet will also grow our capacity. As a school located outside the proximity of main training, we have to travel to attend professional development. There are a lot of innovative and inspiring things happening in capital cities or in major centres, alas the logistics and cost around attending these opportunities on a regular basis mean we often miss out.

Through working with the ACARA DTiF (Digital Technologies in Focus) Project in conjunction with Project Curriculum Officer Kim Vernon, the staff at St Mary's Primary School in Moruya have forged ahead to challenge these ideas and begin to explore the multiplicity of ways that the micro:bit and computer programming (both plugged and unplugged) can enrich current practice in classrooms.

Growing & Gathering

The project involved providing staff with mentoring from an ACARA representative who is a digital expert. Kim modelled lessons in

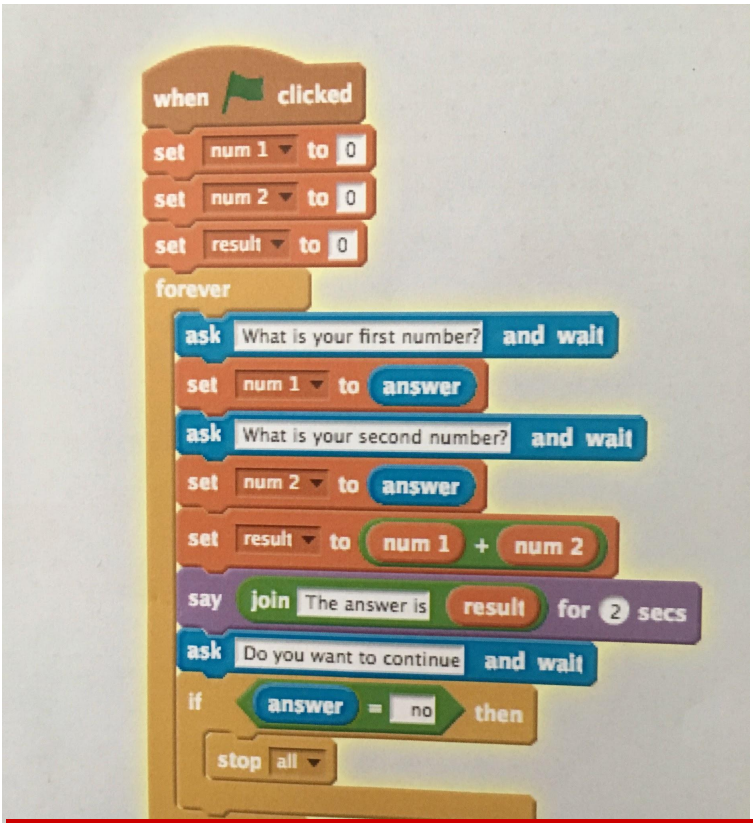
classrooms, where teachers were privy to watching students in Year Four program micro:bits to measure room temperature and light saturation within the scope of a Science lesson.



Students work collaboratively to explore the micro:bit for the first time.

Lessons involved the block programming on MakeCode, whereby the students were introduced to the notion of 'buckets' of information, which they used to create the with 'temperature package'. In the Year Five class students used the micro:bit to create a word game for an English lesson. Students in year 3 were given the challenge to create an addition game, using pre-written code. The simulator on the screen providing opportunity for 'test and see' before they uploaded the program to their micro:bit. Whilst younger students spent time exploring interactive robots to explore coding through a different lense. Students were highly engaged and the teachers were very impressed with the many ways that micro:bit could fit into a variety of learning areas. The lessons were incredibly high paced, with students working quickly through the instructional sequence, which demonstrated 'end goal' learning.

The integration of the micro:bit into Key Learning Areas (KLA) was intentional, with the important message to all involved being that the use of technology of any kind needs to have a purpose and not be disconnected from what occurs in classrooms already.



Witnessing the code on the screen and the ability to manipulate it quickly if it did not work ensured time for reiteration and success

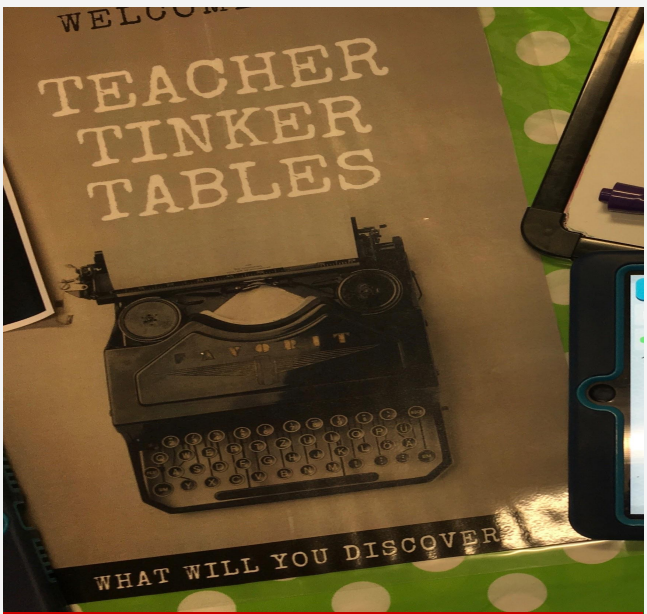
New Ideas & Next Steps

This left one very big question begging; how do we as expert educators emulate this kind of learning in our classroom? Our university degrees do not qualify us, nor do they equip us with the skills to teach these new requirements.

Daunting indeed.

Breaking open discussion between staff, facilitator and mentor through bringing in lessons that had already been planned to engage teachers in discussions around embedding rather than reinventing was key. This was well received and it has certainly provided provocation for staff to begin thinking about 'what next?' as we began to think about the new digital curriculum in a new light.

We often allow students time to engage in pre-learning activities such as sand pitting or tinkering, whereby they are able to look, explore, discuss and discover through play. Why can we not facilitate this for staff? Enter the 'Teacher Tinker Table'.



Engaging teachers in exploring micro:bits was rewarding and challenging.

The micro:bits were programmed to play 'Scissors, Paper, Rock' and left on a table in the staff room with the code for staff to explore. Frustration as they played ... check. Dialogue in the staff room .... check.

An interest to learn more ..... check. So now what?

The next part of this journey will involve working with staff to provide short sharp workshops where they can use the micro:bits in an authentic and integrated manner to give students the programming skills required, whilst walking alongside the staff in the classroom to deliver the kind of lessons they want their students to experience.

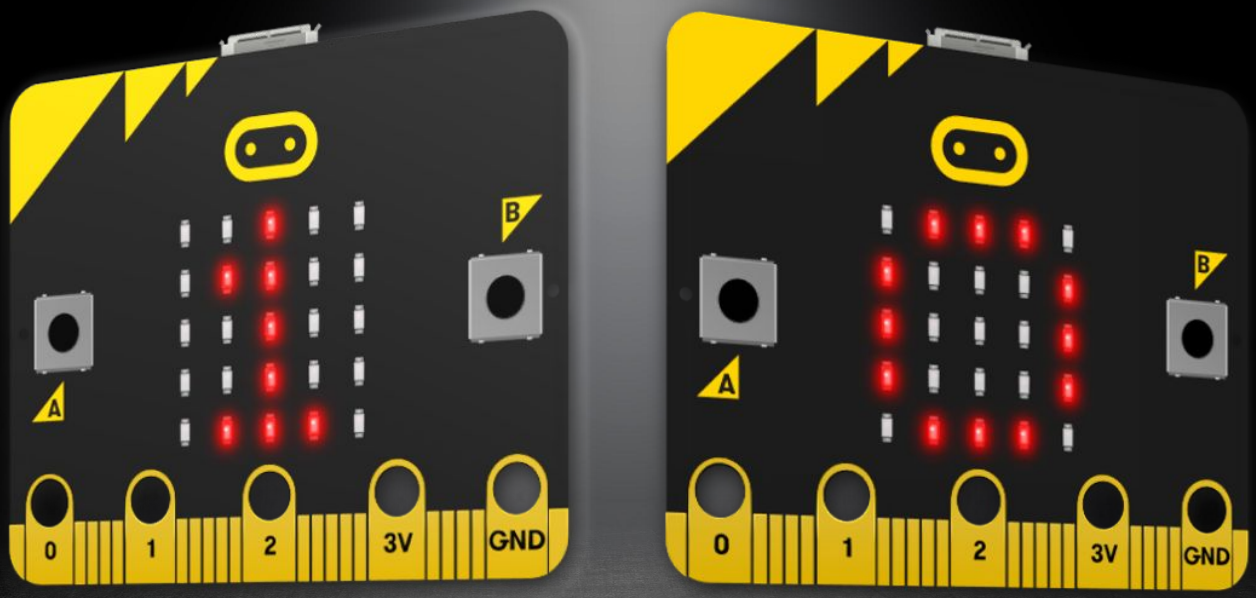
micro:bit we are finding the fit!

micro:mag needs you!

All of our content is written and provided by community members. We're really keen to hear from anyone who would like to contribute to the magazine, whether you're a seasoned writer or just want to have a go.

Get in touch!

micromag.cc/contribute
hello@micromag.cc
@micro_mag

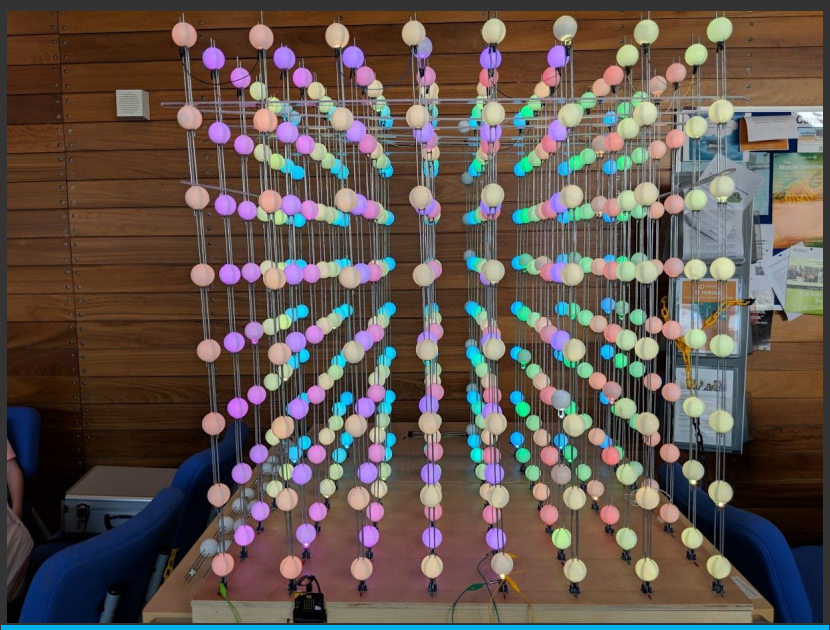


AMAZING micro:bit Projects

This issue, we wanted to showcase some of our favourite micro:bit projects we have recently seen doing the rounds on Twitter and in the community. The amazing projects our community gets up to is really what makes the micro:bit amazing and unlike any other microcontroller out there.

Whether it be in Schools, at home or at a club, the micro:bit is powering tons of cool projects like LED cubes, there is no shortage of cool projects with the micro:bit. So, without further ado, in no particular order, let's get into some of our favourite micro:bit projects!

Cubert



Cubert in action! Photo: Paul Beech

Made by:

Lorraine Underwood

What is it?

Cubert is an amazing 8x8x8 3D LED cube built by maker extraordinaire Lorraine Underwood. This dazzling project features 512 LEDs soldered onto a custom PCB that are encased inside a ping pong ball. Each LED is fully programmable and addressable, so you can give it an RGB value and have each LED turn a custom colour of your choice. Cool, eh? Lorraine says: "It is a metre cubed in size. It breaks down into individual columns that can be transported in the boot of my car." You may have seen Cubert at events like Maker Faire UK or Raspberry Fields last year. Cubert has been built twice with Cubert 2.0 being the one that rectifies the lessons learnt from the first. Building Cubert was a long and tedious project, as you may have seen on Twitter, where Lorraine posted regular updates on building cubert at @LMcUnderwood.

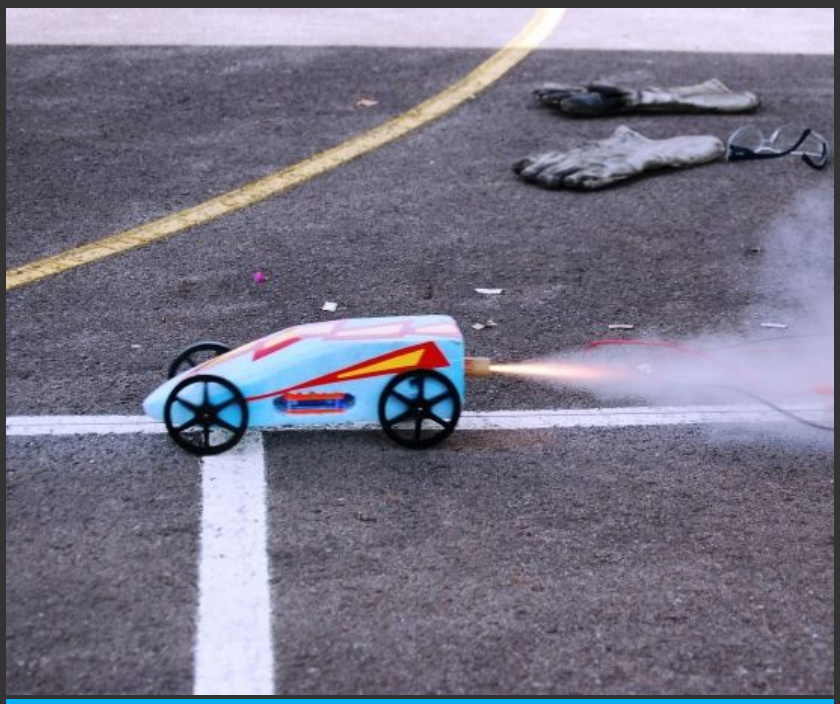
How does it work?

Cubert has 512 Neopixels which are tiny LEDs soldered onto a PCB. In cubert one, Lorraine hand soldered these LEDs onto a custom PCB, this was very temperamental and there were "lots of loose connections". In Cubert 2.0, Maker company Pimoroni donated some redesigned PCBs in which lorraine used Copper Rods to hold the cube together and connect the PCBs. The cube is mainly controlled by MakeCode's NeoPixel extension for the micro:bit, but can also be controlled via a Raspberry Pi or Arduino.

Find out more

Lorraine did a talk at Raspberry Fields here: go.micromag.cc/cuberttalk

Race for the line



Look at them go! Picture: Bloodhound Edu

How does it work?

Students get given a polystyrene block that they can model into a design and shape of their choice. A hole in the centre of the car is cut out to hold a micro:bit in a custom built case. On this micro:bit is the Bitty Data Logger hex file (learn more about Bitty software on page 10) which will connect to a phone via bluetooth. This tracks the accelerometer data from the micro:bit and displays it on the phone via a handy little graph. We think this is a great project to get kids involved with the micro:bit and STEM.

Find out more

Visit the Race for the Line project here: go.micromag.cc/raceforline

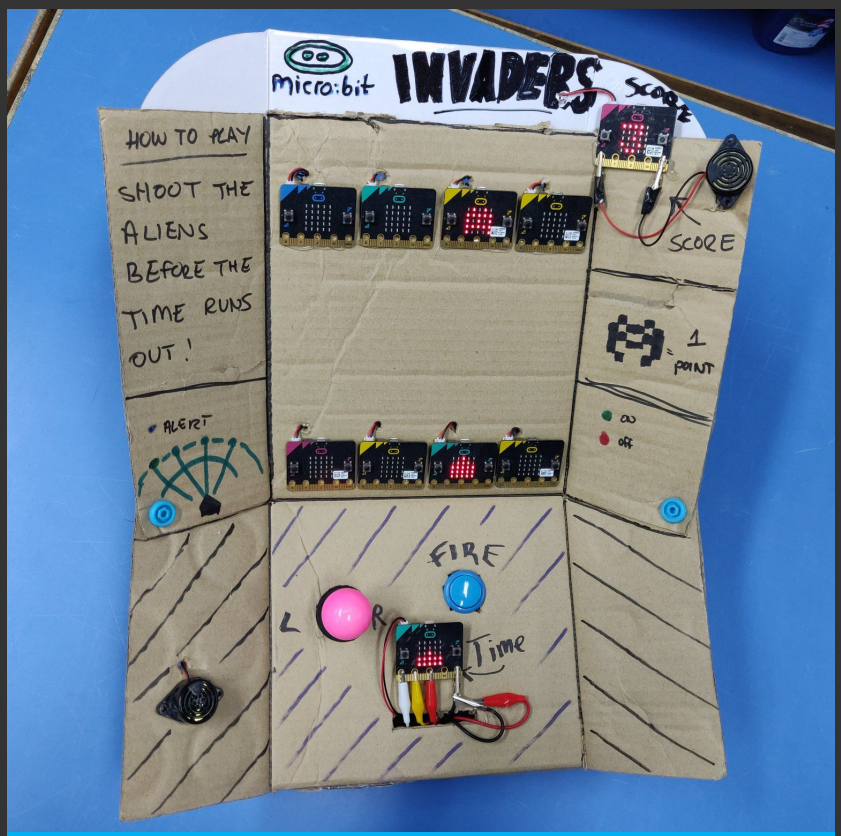
Made by:

Bloodhound Education

What is it?

You may have heard of the Bloodhound land speed record attempt car, well in 2016, Bloodhound Education and partners brought a competition to Secondary schools in which they could build a model out of polystyrene which was cut and designed to how the participants wanted and then raced. Bloodhound say "Inspired by the rules governing the World Land Speed Record attempt that the 1,000 mph BLOODHOUND Supersonic Car is targeting, the model rocket cars must blast along a wire and through a set of timing gates with a BBC micro:bit on board gathering vital data that enables the Teams to modify and improve their designs." Another cool thing about this project is others got creative with testing out their projects, by building timing gates using a micro:bit, serial data and infrared sensors, like this project by Les Pounder : "micro:bit Infrared Timing Gate"

micro:bit Invaders



The project in its full form. Photo: Stu Lowe

Made by:

Stu Lowe

What is it?

micro:bit invaders is an amazing project made by Hong Kong teacher Stu Lowe. Stu is always making amazing things with the micro:bit and we love this cardboard craft project that is powered by the micro:bit. It consists of 11 micro:bits in total which are all programmed in Microsoft MakeCode for micro:bit. This recreates the classic game by having the ghosts appear on the top row of micro:bits and the shooter on the bottom row. It also features a neat little controller box that has a micro:bit acting as a timer as well as controlling the joystick to move the shooter along the bottom and a blue button to fire. Another micro:bit is also placed at the top which counts the current score of the game. We think this is a really impressive mix of technology and crafts!

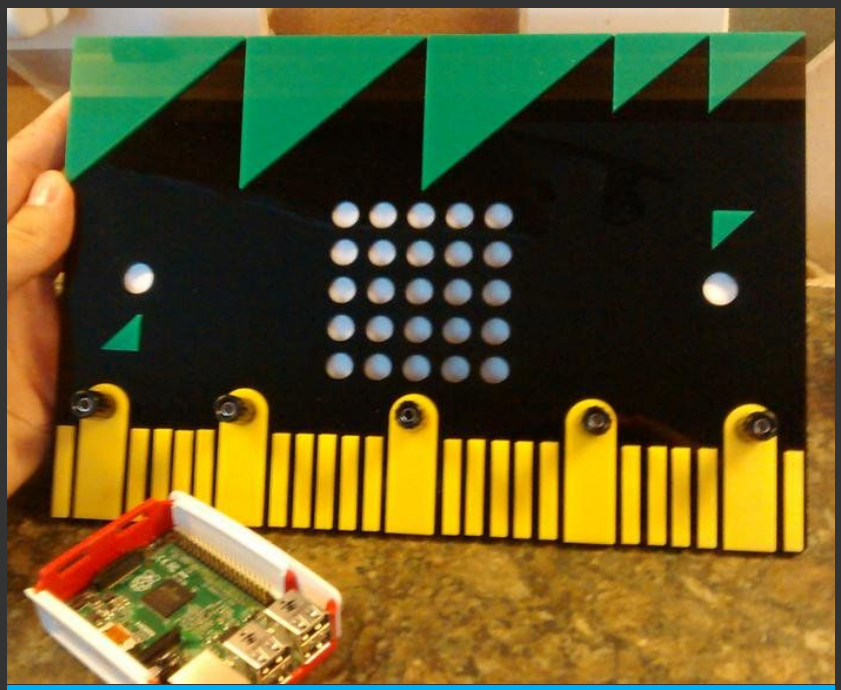
How does it work?

The 11 micro:bit's that make up micro:bit invaders are all coded in MakeCode. All of the micro:bit's are connected up via the radio functionality of the micro:bit. It's split up into 5 different programs, one for the controller, ship, alien, score and another for music. The controller acts as the hub for the other micro:bits and communicates via wireless radio, telling all the micro:bits what to do and when.

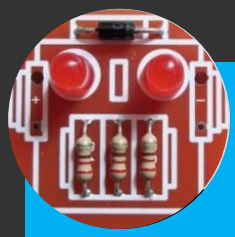
Find out more

Stu recorded a video on Twitter showing the project here: go.micromag.cc/mbinvade

mega:bit



The original mega:bit. Photo: Andrew Gale



Made by:

Andrew Gale

What is it?

The Mega:Bit is a scaled up micro:bit, it features the 5 x 5 LED matrix and buttons, just like the normal micro:bit. Best of all, it is fully functional. The mega:bit was developed for use in the classroom as a prop to show students how a micro:bit works. It is all controlled by a very complicated looking PCB which has 25 big LEDs in which are then encased by an acrylic micro:bit. However, the wiring is quite complicated as not all the LED signals are broken out on the micro:bit's edge connector, so this is something that would be quite difficult to build at home. Fear not though, Andrew is planning on taking the mega:bit to market with an updated version that should be easy for everyone to make themselves! The original plan for the mega:bit was for it to be controlled by a Raspberry Pi as the micro:bit was not available when this project was made, however, David Whale, who had access to one at the time, stepped into help with making this fully micro:bit powered.

How does it work?

The micro:bit is slotted into an edge connector which is connected to a PCB with wiring for the big LEDs and buttons. Some of the LEDs from the micro:bit are directly read from the pins of the edge connector, however, some of the LEDs are not broken out so you have to carefully solder wires directly onto the LEDs which is very fiddly. At the end though, it all comes together as a neat little package for a fully functioning large micro:bit.

Find out more

Andrew has written a post on his blog here: go.micromag.cc/megabitpost

DIY VR with micro:bit

The VR headset. Photo: Colin Ord

Made by:

Colin Ord

What is it?

VR headsets can be very expensive, so why not make your own? Well, Colin Ord did just that! He's taken a Google Cardboard VR headset, a HDMI display and 2 micro:bits to make his own. The main part of the project is the head tracker which uses a micro:bit and it's accelerometer to track the movement of the headset. Instead of using a phone (which is what normally gets used with the Google cardboard), Colin used an LCD which was powered by a Raspberry Pi. He used Blitz 3D to create a 3D scene which can be interacted with when using the headset. The micro:bits output their data via serial which is then read to create a realistic feel. Just like normal VR headsets, the visualization is split up into 2 parts, one slightly offset from the other to create a stereoscopic feel.

How does it work?

The headset uses a Raspberry Pi to show the animation on an LCD display. 2 micro:bit's track the position of the headset with one connected to a visual basic program running on a laptop which outputs the accelerometer value of the micro:bit connected to the headset and sends the data via real time serial data to the visual basic program. The micro:bit's are programmed in MicroPython using the Mu editor.

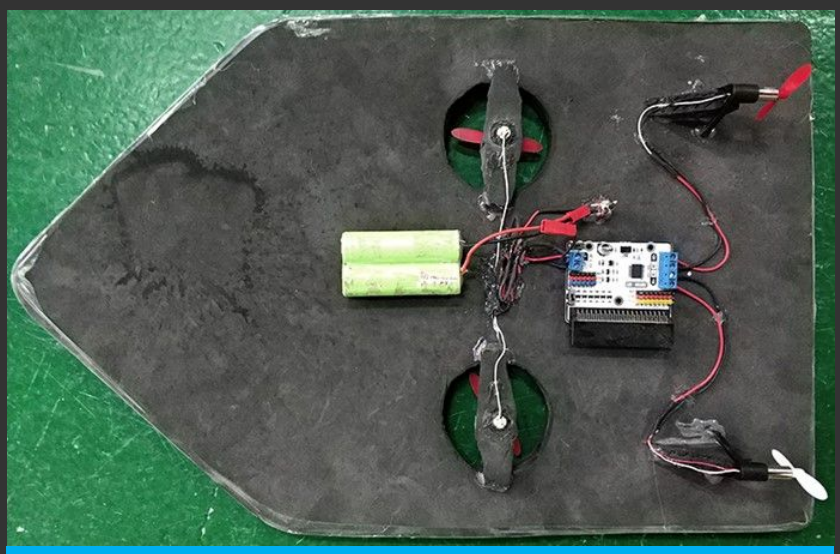
Find out more

Colin has written a blog post here:
go.micromag.cc/vrheadset

micro:bit Hovercraft

Made by:

ElecFreaks



Look at them go! Picture: Bloodhound Edu

What is it?

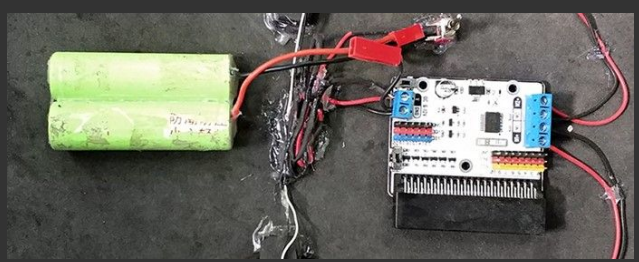
We have all seen micro:bit powered robots before, they are everywhere! But what about a micro:bit powered hovercraft? Well, the folks over at ElecFreaks have made this awesome hovercraft that you can build yourself. Using their neat little micro:bit motor driver board, they have taken some cut out foam, 4 motors, propellers a micro:bit and their driver board to create a hovercraft that can run on ground or even in water. The great thing about this project is that it's really simple to build at home and it is cheap too yet at the same time it is a fun little project that can get anyone engaged with coding and the micro:bit.

How does it work?

The hovercraft sports a foam board, motor controller, 4 hollow cup motors, 4 propellers, batteries and a micro:bit. This project also features a controller which is really cool, this is powered by another micro:bit. This uses the ElecFreaks Joystick:bit which is a DIY soldering kit that allows you to make a nice little controller for things like this. It is all programmed in MakeCode and the 2 micro:bits communicate over Radio.

Find out more

Learn how to build your own at:
go.micromag.cc/hovercraft



1000 micro:bit Display

The 1000 micro:bit display doing it's thing. Photo: Kitronik

Made by:

Kitronik and micro:bit partners

What is it?

Imagine if you had a wall made from micro:bits that you could scroll your name on and make large scale games. Well, the folks over at Kitronik built a huge display of 1000 micro:bit's all connected together to let you play games and scroll text on this huge display. Countless hours and huge amounts of effort went into this. It was originally built for the first public showing of the micro:bit back in 2016. The result is phenomenal, Kitronik often take this to events like Makerfaire UK and BETT (the show it was originally made for). When at these events, it runs a number of different demos from Space invaders to just scrolling text. This project is the work of many volunteers linked to the original BBC micro:bit project and it took a few months of testing and numerous smaller prototypes, but this display is always a huge hit at events we have been to and seen it at.

How does it work?

The way it all works is that there is 1 master micro:bit that uses serial to talk to 8 other micro:bits, each of these pass a message onto 5 individual micro:bits at the bottom of the display and these are relayed up the display in columns and gives an amazing result of a display made from 100 micro:bit's on which you can play games, display images and scroll text.

Find out more

Kitronik wrote a post about the process here:
go.micromag.cc/1000mb

MicroBike

Made by:

Musab Kılıç

The micro:bike controlling a game

What is it?

This project allows you to control PC games using a micro:bit as the game controller. The micro:bit is used to show directions and get acceleration data. It was developed by Musab Kılıç to show on Istanbul Maker Faire. Musab plans to make the project available by creating a interface for everyone to use it for any game or software.

This is an ideal project for those of you who want to build cheaper versions of game controllers for games like train or flight simulators. As Musab has not created the interface as of yet for people to easily create their own, all the code is on GitHub and it should be easy to adapt into any type of controller you'd like.

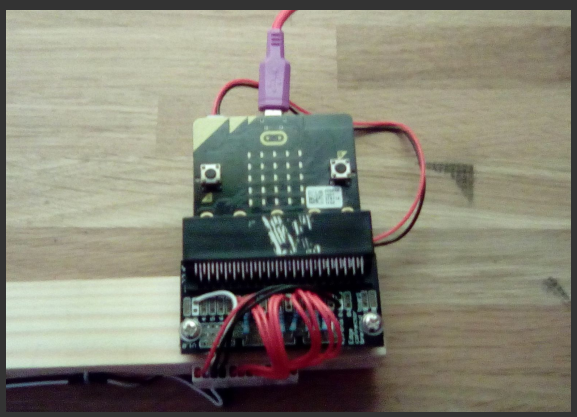
How does it work?

The micro:bit is connected to a computer by a USB cable. A program runs on the computer, which uses David Whale's bitio module, to get information from the micro:bit. The program simulates key presses on the computer to control other programs, such as games. To send these keypresses to the computer, the PyUserInput library, "a module for cross-platform control of the mouse and keyboard in python"

Find out more

Visit Musab's website here:
go.micromag.cc/microbike

micro:bit Midi Guitar



Pictures of the guitar and it's wiring.

Credit: David Whale

Made by:

David Whale

What is it?

The micro:bit MIDI guitar is a musical instrument. You can play music on it just like a real guitar. It is made from two pieces of wood left over from a wardrobe and is decorated with a number of strips of copper tape (slug barrier tape), uses two Kitronik breakout boards, and two micro:bits. It simulates a single string on a guitar, which is enough for most pop songs! But, it does not generate any sound itself! Instead, it connects to a computer via a USB connection used as a MIDI (Musical Instrument Digital Interface) instrument, and can be used with any music production software (David used Garage Band, which is pre-installed on Apple Mac). So, it is really just an input device. In the future, David wants to add more left hand pads (giving a wider note range), add a 'palm mute' pad to improve playing style and add a moveable capo mode.

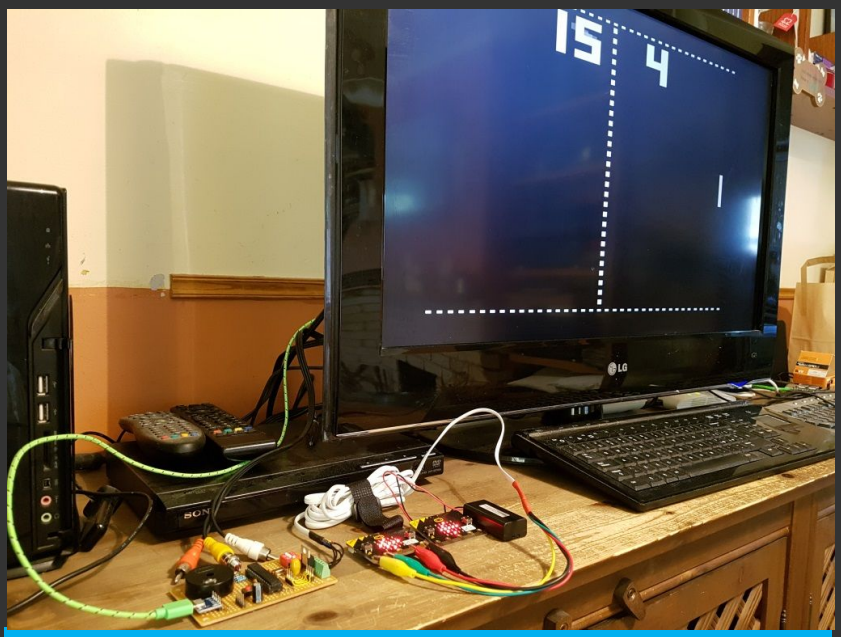
How does it work?

One micro:bit senses the left hand frets, and the other senses the right hand touch pads and also sends MIDI messages to the computer. Touch inputs on the micro:bit are resistive touch, so you have to hold the GND connection to complete the circuit. On the back of the wood is a long strip of copper tape connected to the GND pin on the micro:bit, and putting your thumb here naturally completes the circuit.

Find out more

Visit the GitHub repository:
go.micromag.cc/midiguitar

micro:bit TV-Pong



Cubert in action! Photo: Helene Virolan

Made by:

Nigel Kendrick

What is it?

Play the classic Pong game on a TV (composite video) - using BBC micro:bits as paddles! Using a Grant Searle's pong game for the Arduino, Nigel has hacked the existing code to use 2 micro:bit's as up and down paddles for the game. The aim of this project is for use in STEM coding workshops using micro:bit's as the game paddles. The micro:bit's are wired to the Arduino (a custom one built on some stripboard). Nigel used the micro:bit's edge connector to do this. The micro:bit's can send a few different commands to control the Pong game like, Left player down/up, Right Player up/down. Nigel has also produced a few different version of this project, for example, you can use the Paddles remotely using the Radio feature on the micro:bit or you can use the Bitty controller software via bluetooth too.

How does it work?

The project consists of a custom arduino with an ATMEGA328. This has a composite video out cable that plugs into a PC. 2 micro:bit's are used as paddles and these are connected via the edge connector to the arduino via cables, you can also use it wireless though if you wanted. In the GitHub repository, the 'paddle' files use the A and B buttons on the micro:bit as digital paddle controls. The code is written in MakeCode so will be easy to adapt.

Find out more

Visit the GitHub repository here:
go.micromag.cc/tvpong

Alan Yorinks

@BrassFigLigee
github.com/MrYsLab

Outside of the office Alan likes to read mystery novels, go to the gym, and volunteer as a literacy tutor for adults. On a warm sunny day, you might find him in the rough, foraging for a little white ball that refuses to stay on the fairway.

s2m

*Bridging **Scratch** and **micro:bit***

s2m - The Scratch/ micro:bit Bridge

The Why, What and How of s2m.

In a time, long, long ago, - well not all that long ago – 2013 to be exact, I had discovered the Scratch programming language. Roughly at the same time, I discovered the world of microcontrollers. "Hmm", I thought, "wouldn't it be great if I could use Scratch to program and control the microcontroller"? Little did I know that this innocent pondering would eventually lead to the development of *s2m,* the Scratch to micro:bit bridge.

Installing

If you would like to run s2m on Windows, Raspberry Pi Raspbian, or macOS as you read this article, full installation instructions can be found at https://mryslab.github.io/s2m/install/

The Why

In the fall of 2017. I received an email from a user asking if I would be willing to create a program that would allow Scratch to control a micro:bit, similar to some of the work I have done for Arduino and Raspberry Pi microcontrollers.

Not knowing much about the micro:bit, I purchased one to see what all the fuss was about. I could not believe what an amazing little microcontroller it turned out to be! It is packed with functionality, it's easy to program and it's easy to use. Best of all it is inexpensive.

The What

So, I immediately got to work on s2m, making sure that it would be compatible with Windows, macOS and Raspberry Pi. Connection to the micro:bit is accomplished using a USB/serial cable because not everyone has a Bluetooth interface on their computer.

In a nutshell, when you start the s2m program, it launches Scratch and then connects to your micro:bit.

Scratch is loaded with custom blocks that allow you to:

- Choose and display standard micro:bit images
- Scroll text on the display
- Control any pixel on the display, including its intensity
- Clear all the pixels on the display
- Detect when button A or B is pressed
- Detect the 4 tilt directions
- Allow you to connect external analog and digital input devices to the micro:bit and have Scratch read the values.
- Allow you to connect external analog (PWM) and digital output devices to the micro:bit and have Scratch set their values.

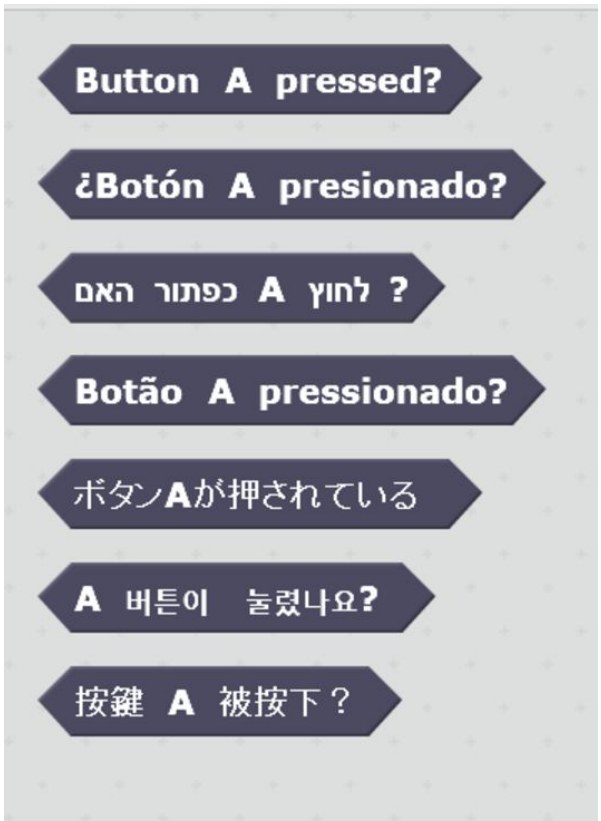
The s2m Scratch programming blocks.

Thanks to the generosity of the s2m user community, s2m comes with blocks translated to:

- Brazilian-Portuguese
- Hebrew
- Japanese
- Korean
- Spanish
- Traditional (tw) Chinese

These blocks can be loaded at startup by simply providing a command line argument to select the language of your choice.

If you would like to submit blocks translated to your native language, you can find instructions on how to do so in the s2m User's Guide at https://mryslab.github.io/s2m/.

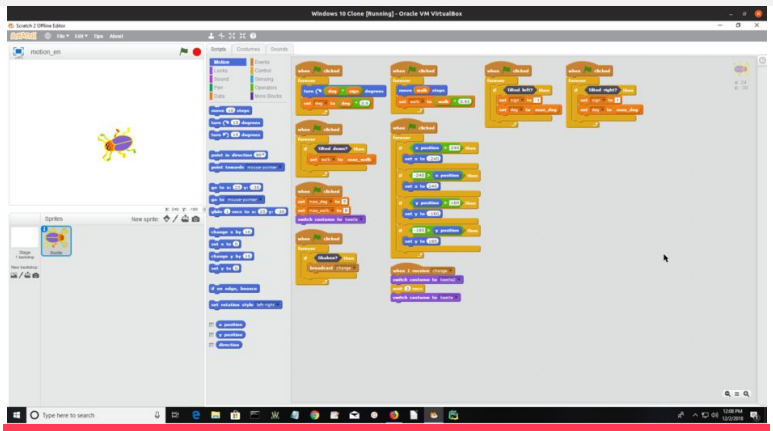


A Scratch block translated for various languages.

Once Scratch appears on your screen, you create an s2m project just like you would any other Scratch project, except you will find the s2m blocks under the More Blocks section of Scratch.

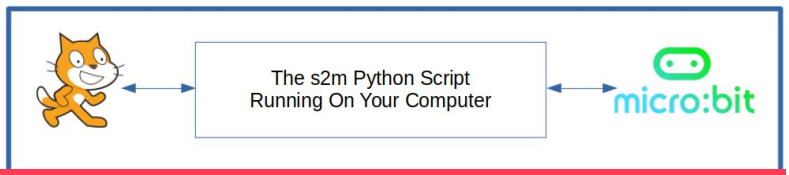
Here is a screenshot of a demonstration program, provided by Edson Sidnei Sobreira,

that is included in the s2m distribution. By tilting your micro:bit, it allows you to move and interact with a Scratch sprite.



A demonstration program included with s2m, that controls the motion of a Scratch sprite by tilting a micro:bit.

The How



The components of s2m.

The s2m program consists of 3 major parts – the Scratch 2.0 offline editor, the s2m Python code that runs on your computer, and a custom MicroPython script flashed on to your micro:bit.

Don't worry if this all looks and sounds a bit complicated. When you launch s2m, it will start everything for you automatically. Just start s2m, and you are ready to have fun programming a Scratch/micro:bit creation.

By the way, I mentioned that s2m uses the off-line version of Scratch. You might be wondering why? The reasoning behind this is, that not everyone has a reliable or fast internet connection. By using the off-line version, virtually everyone can use s2m, including Raspberry Pi users. Pretty neat, if I do say so myself.

Want More Information?

An online User's Guide is available that explains how to install and use s2m. You can find the guide at https://mryslab.github.io/s2m/

Why not give it a try? See what get creations you can make by combining Scratch with micro:bit!

Questions or Comments?

Questions and comments are always welcome. You can find my contact information on my Github page shown at the top of this article.

Carlos Pereira Atencio

Carlos first encountered the micro:bit at a Python meetup 3 years ago and immediately got hooked. He is now a Software Engineer at The Micro:bit Educational Foundation.

@carlosperate

Python Editor beta: Connect your micro:bit to the browser

The beta version of the editor allows you to program your micro:bit with a single click and access the REPL from the web.

You Will Need:

- A micro:bit
- A USB cable
- Chrome v65 or newer

Wouldn't it be great to program your micro:bit directly with a single click or access the REPL from your browser? Thanks to WebUSB we can do just that, and we will show you step by step how easy it can be.

You might be familiar with the online Python Editor, but did you know there is a beta version with exciting new features?!

If you follow the Micro:bit Foundation Beta Programmes (https://microbit.org/testing/), you might be already aware of the new WebUSB features on the MakeCode and Python editor. WebUSB is an exciting new standard that provides access to USB devices directly from the web. This means you can connect your micro:bit directly to the online Python Editor, opening a new world of possibilities, like direct flashing and connecting to the MicroPython REPL.

Preparing your micro:bit

The first thing you will need to use WebUSB is

to update the firmware in your micro:bit. The micro:bit has two microcontrollers: the main nRF51 where all your programs run, and a secondary microcontroller (next to the battery connector) which handles the USB features. The micro:bit firmware runs in this secondary microcontroller and updating it to version 0249, or newer, will enable the WebUSB functionality.

Let's update it following these easy 4 steps:

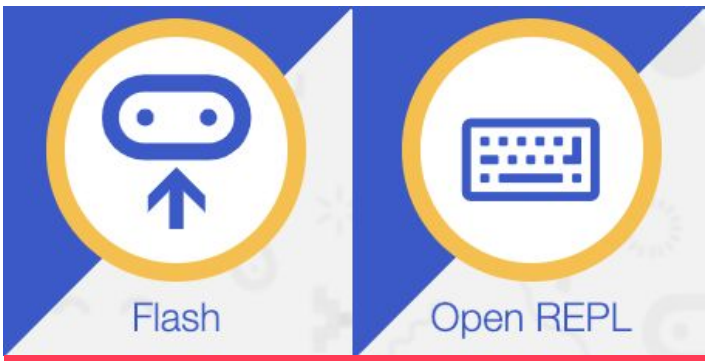
- Download the latest firmware hex file from https://microbit.org/guide/firmware/ (at the time of writing is version 0250)
- Plug your micro:bit to your computer via USB while pressing the reset button, this will enter MAINTENANCE mode
- Drag and drop the downloaded firmware hex file into the MAINTENANCE USB drive, the back LED should flash while the file is being copied
- The micro:bit will reboot back into the normal MICROBIT mode. You can open the DETAILS.TXT file to check that the "Interface Version" has been updated

WebUSB compatible browser

The WebUSB standard is still in Editor's draft status, and at the moment only Chrome has implemented the API. To use WebUSB with the online Python Editor (or MakeCode) make sure you use Chrome version 65 or higher.

Python editor beta version

Trying the beta version of the online Python Editor is easy, just go to https://python.microbit.org/v/beta and you can start coding right away ☺. If you are using a WebUSB-enabled browser, you will notice there are two new buttons on this version of the editor: "Flash" and "Open REPL".



Flash and Open REPL are the two new WebUSB features

These two buttons expose the new WebUSB features, and as their name implies you can flash your micro:bit directly from the browser or open a MicroPython REPL session.

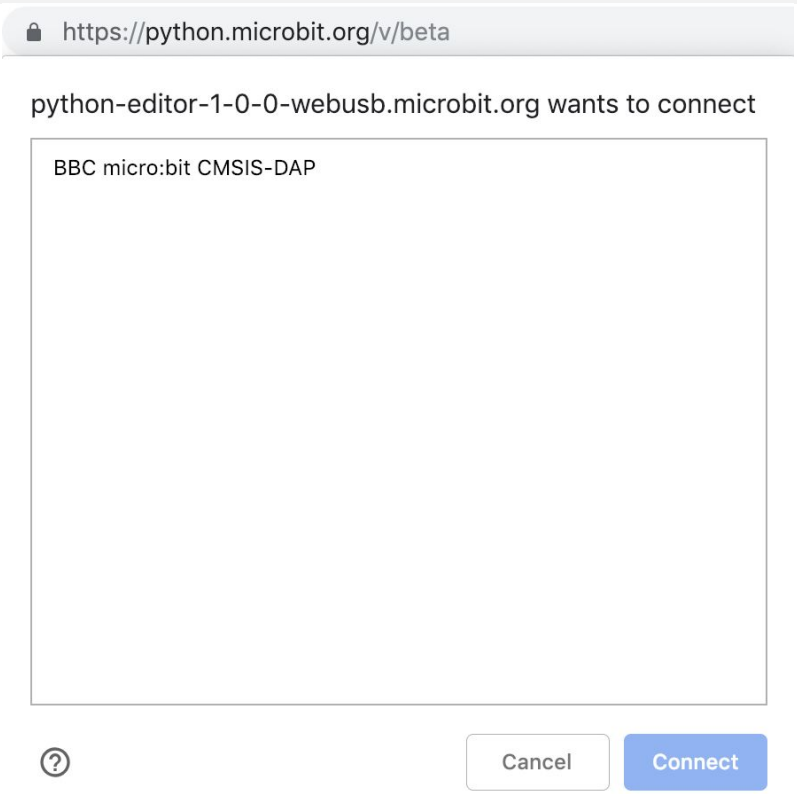
Please remember that this is beta software, the UX is not quite finalised yet, the editor might contain bugs and sometimes it might break, but that is half the fun of living on the edge!

On that note, if you like finding bugs and helping The Foundation make the editors better, you can become a micro:bit tester by signing up to the Beta Testing Programme at https://microbit.org/testing/. Your help and feedback are invaluable, and you can help shape these tools.

Pairing the micro:bit

Before any of the WebUSB features can be used the micro:bit has to be paired with the Chrome browser.

Clicking on the “Flash” or “Open REPL” buttons will open the pairing window. The micro:bit will appear with the name “BBC micro:bit CMSIS-DAP”.



Chrome browser pairing window.

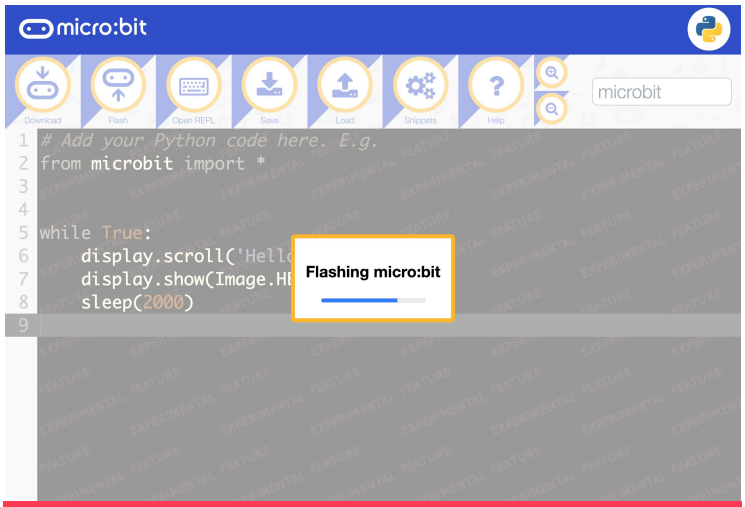
Select your device and click “Connect” to finalise the pairing. Every time you try to use the “Flash” or “Open REPL” functions it will ask you to pair, but don’t worry, this is a temporary implementation, in future versions of the beta editor, you will only need to do this once per session.

Flashing with WebUSB

If you have used the micro:bit before, you are probably familiar with the flashing process from the online editors. You click the “Download” button to get a hex file, and then drag and drop it into the micro:bit USB drive.

Wouldn’t it be great to program your micro:bit directly with a single click? Thanks to WebUSB we can do just that!

To flash your micro:bit simply click on the Flash button, you should see a progress bar covering the editor and a few seconds later your new program should start running on your micro:bit.



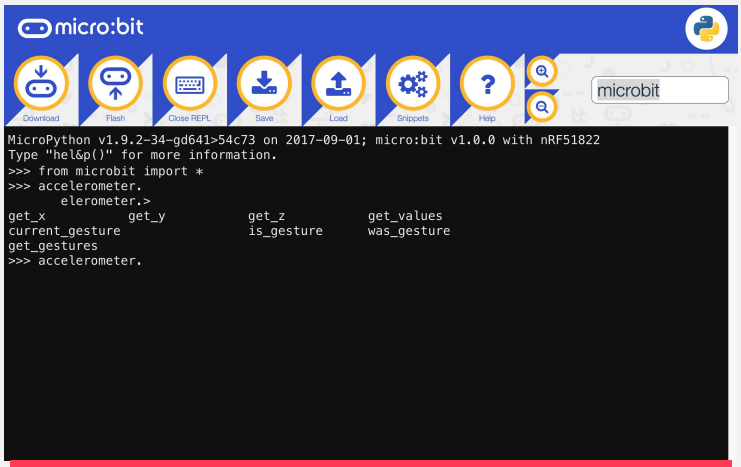
Flashing a micro:bit directly via WebUSB.

MicroPython REPL with WebUSB

The MicroPython REPL (Read Evaluate Print Loop) allows you to type code in a serial connection with the micro:bit and execute it instantly. This is a very interactive way to explore the micro:bit with Python, to quickly test code and, to discover the available functionality.

Until now, to access the REPL you needed to install a serial terminal or application on your computer and figure out which serial port corresponds to the micro:bit (or use a desktop editor like Mu [https://codewith.mu], but now thanks to WebUSB it couldn’t be easier! Click on “Open REPL” and the editor should be replaced by the REPL view. You can start typing code here,

send it to the micro:bit and, immediately get the result!



REPL view on the online beta Python Editor.

Not only that, but the REPL offers autocompletion. Start typing a variable, function, module or method, and press the “TAB”

MicroPython v1.0

At the moment the version of MicroPython included in the beta editor is the same as the one included in the normal editor. However, new beta versions of MicroPython are coming and they will be added to the beta editor first, so keep an eye for interesting new features in this area as well. It is also worth noting that the official v1.0.0 of the micro:bit port of MicroPython was released last September. Here are a few things this release included since version v0.9:

- Tons of bug fixes
- Better copy/paste on the REPL
- NeoPixel module is now compatible with ws2812b-mini devices
- Errors scrolling on the display can be terminated with a Ctrl+C on the REPL
- Running `help(‘modules’)` gives you a list of all available modules
- `machine` and `micropython` modules added with helpful functions from upstream MicroPython
- Pins can configure the internal pull-up/down resistors
- New calibration UX
- Light sensing capabilities added to the display module

Troubleshooting

Keep in mind that as beta software, there are still a few rough edges that need to be polished before it is ready for a public release, and it is quite possible some things might not work exactly as expected. The following support article contains a list of known issues with WebUSB and workarounds or fixes, so if something doesn’t work quite right, check that article first. If your issue is not already there, please open a new ticket in the support website. We will love to you help you and it will also help anybody else going through the same problems.

go.micromag.cc/webusb

Happy Coding!

Get fresh copies of micro:mag delivered to your inbox

Make sure you never miss an issue of the unofficial community magazine for micro:bit lovers - we'll email each issue as soon as it's released (and we won't ever spam you or give your details to anyone else)

To sign up, go to

micromag.cc/email

Diego Fonstad

Diego Fonstad is the co-founder of Imagination Supply Co. a Benefit Corp whose mission is to encourage children to explore STEM through hands-on projects.

@lectrifyit
lectrify.it

Designing an Electronic Pet

Explore physical computing with your micro:bit using an engaging and flexible framework built around the theme of an electronic pet.

You Will Need:

- micro:bit
- bit:booster

I have been working closely with educators since 2011 designing hands-on projects and a universal goal is to create a curriculum that is engaging and relevant to students while providing a vehicle for teaching foundational concepts. One guiding principle we often follow was coined by the inventor of Logo and pioneer of constructionist education, Seymour Papert, who said projects should be designed around the principle of "low threshold, no ceiling." We designed the bit:booster with this in mind and the ePet project is an engaging project that is easy to use outside of the box but also allowed students and educators to extend and explore every dimension of physical computing: coding, digital design, mechanical engineering and even AI and our relationship with technology.

The micro:bit is a powerful platform, enabling open-ended projects with almost unlimited possibilities, but that can be both a blessing and a curse: where to start? We've developed a curriculum sequence around the concept of building an electronic pet that provides an engaging foundation for exploring the full range of possibilities of the micro:bit and bit:booster.

Students start with a simple prompt "build an electronic pet using only the micro:bit and bit:booster." This leads to a discussion around what characteristics they want to build into their "pet." This discussion provides an opportunity to explore the capabilities of the micro:bit and bit:booster in the context of a specific goal: how can they be applied to their pet?

Feature	Type	Description
5X5 LED matrix	Output	The micro:bit has a 5X5 red LED matrix
2X5 NeoPixel	Output	The bit:booster adds 2X5 Neopixel matrix

Piezo Buzzer	Output	The bit:booster adds a piezo buzzer for playing music
Buttons	Input	The micro:bit has two buttons for input
Light Sensor	Input	The 5X5 LED matrix can be used to measure the light level (
Accelerometer/ gyro/ Compass	Input	The micro:bit has can measure acceleration, pitch, roll and compass direction.
Temperature Sensor	Input	The micro:bit can measure temperature in celsius

Building upon the above list, students brainstorm how to apply the inputs and outputs to their own electronic pet (ePet) program.

OUTPUTS GIVE THE ELECTRONIC PET PERSONALITY

The LEDs and NeoPixels are quickly identified as a possible face. Students next need to decide the direction of their ePet. If the ePet is upright, the NeoPixels can be eyes and the 5X5 LED matrix can be the mouth. Upside down, the 5X5 matrix becomes an eye and the NeoPixels become the mouth. You'll be surprised how expressive and creative children can be with the LED & NeoPixel palette.

The micro:bit LED matrix is controlled using the standard MakeCode blocks and the NeoPixels are on Pin 2. NeoPixels are individually addressable and are sequenced left to right, top to bottom on the bit:booster. Don't forget that the lights don't need to be static. By creating sequences of lights students can animate the face.

The bit:booster adds a piezo tone generator to the micro:bit and the bit:booster uses the same default P0 for the piezo. Students can explore how different sounds can be used to generate moods and give their ePet personality.

CARE AND FEEDING OF THE ELECTRONIC PET THROUGH THE INPUTS

The accelerometer provides many opportunities for interacting with the virtual pet allowing for different reactions whether mildly shake or dropped suddenly.

The light sensor can be used to explore how the pet responds to the dark and may be used as a way of letting it "sleep"

The buttons can be used to emulate concepts like feeding and bathing like a Tamagotchi pet.

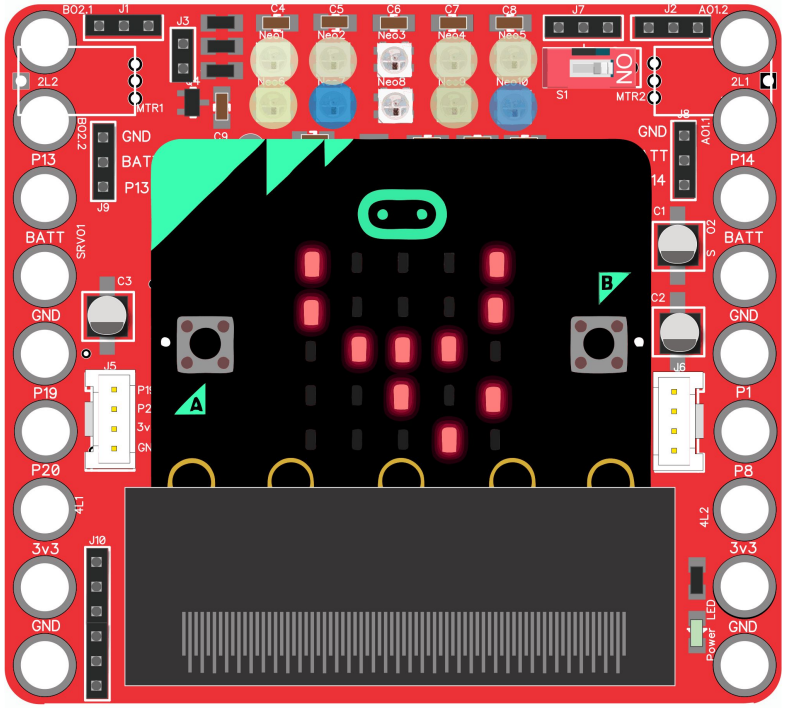
The power of the ePet project is how open-ended it is and can be driven by the students. It can also be build upon incrementally. Students can start with simple ePets of only a face responding to simple inputs. As they evolve their pets, they can evolve their coding to more complex concepts such as storing variables to track how many times the pet is "fed."

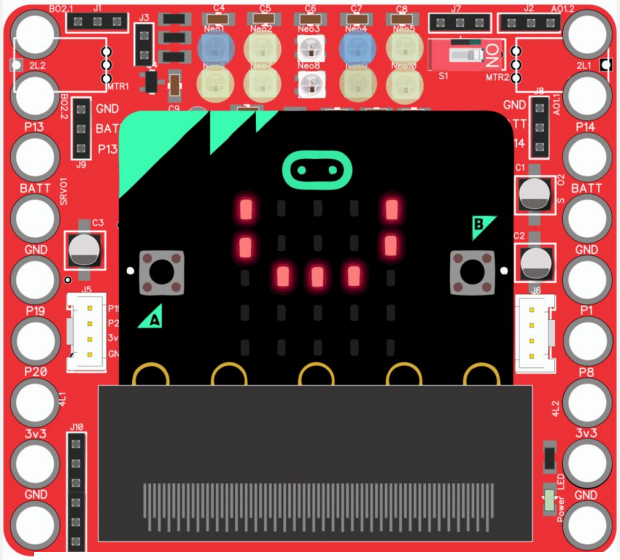
Designing an Electronic Pet

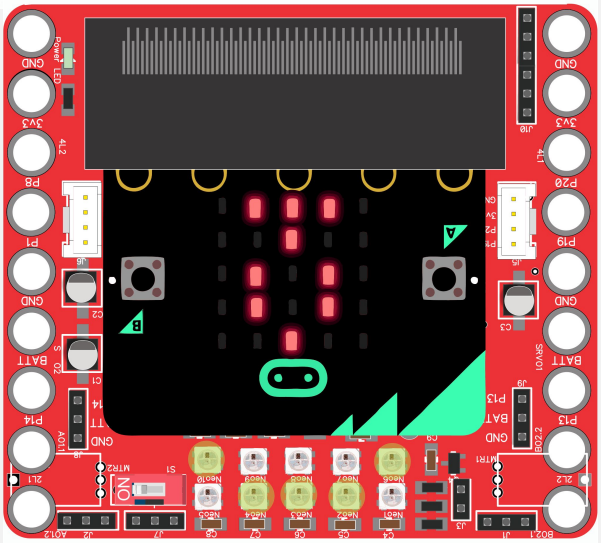
:feature

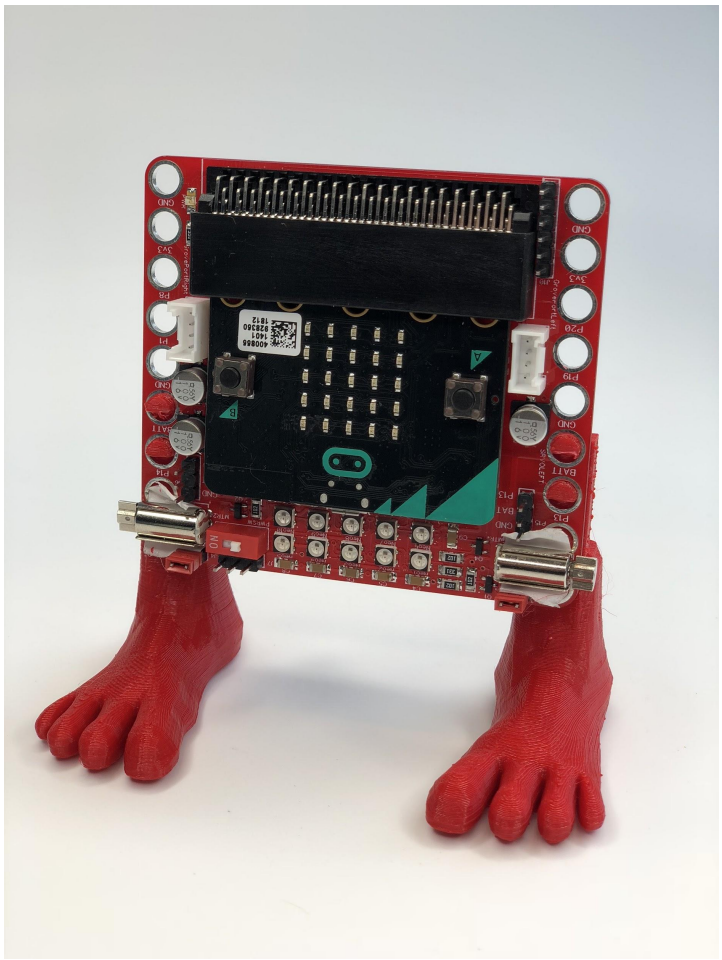
Here's a video of a relatively advanced-coded pet using only the bit:booster and micro:bit https://youtu.be/lwovnlFR5Ck
Here's the sample code in MakeCode https://makecode.microbit.org/_MAf3jCgaE6F0

This project can become a jumping point to more advanced coding and design by adding motors, sensors and add-ons. There is even a Design Thinking extension of this project where students explore therapy robots and use the knowledge they gained from designing their ePet and the Design Thinking process to design a therapy robot.



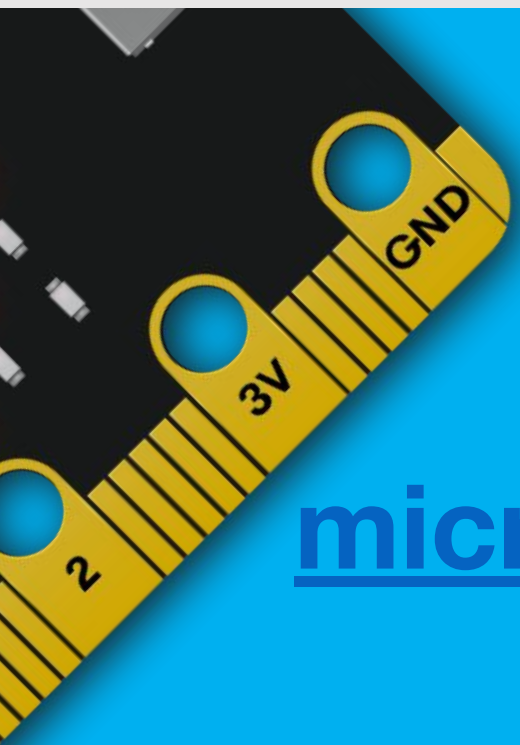






Help us cover the costs of micro:mag

micro:mag is run by a team of dedicated volunteers - but we still need to cover our costs. Donating helps us continue to deliver the community micro:bit magazine free of charge.



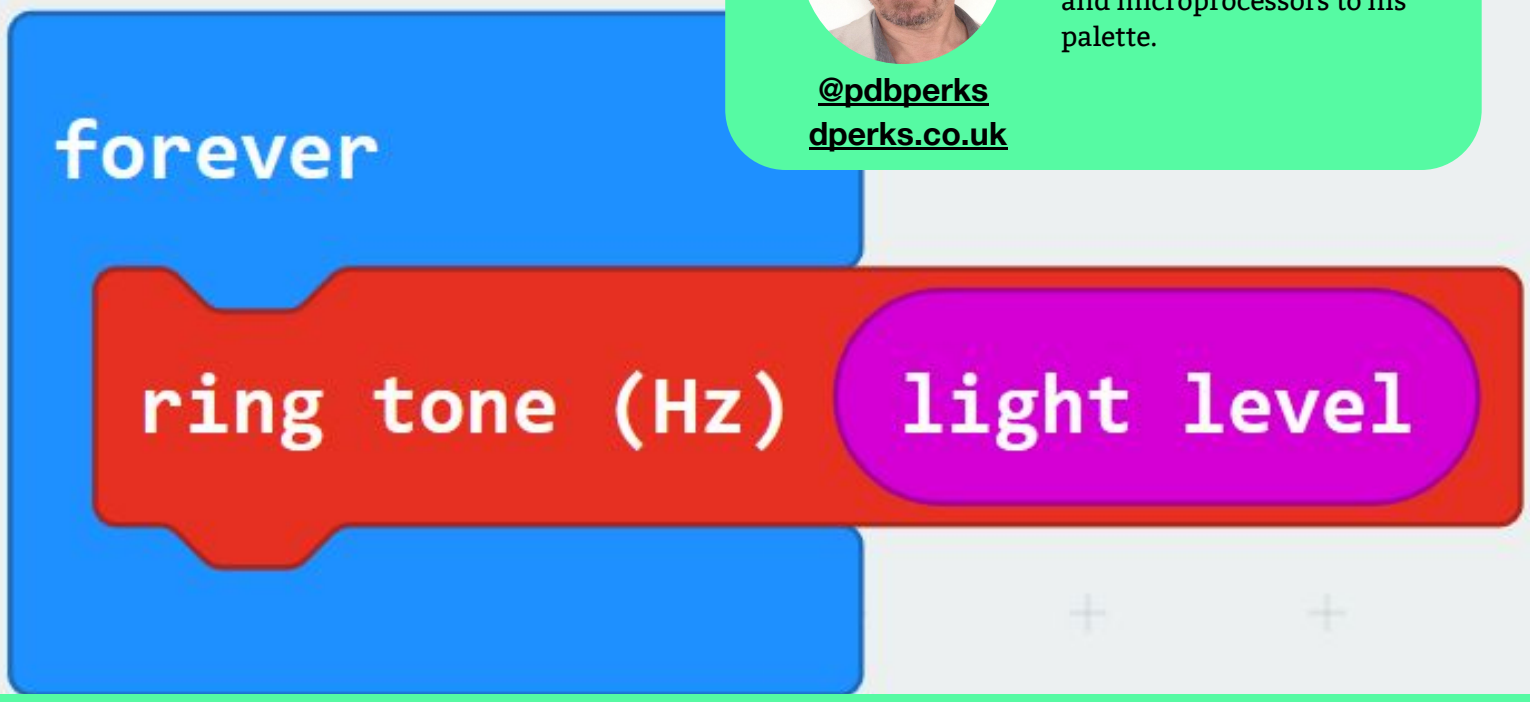
Donate on PayPal:

micromag.cc/donate

David Perks

David is an artist maker who has added electronic junk and microprocessors to his palette.

@pdbperks
dperks.co.uk



What can you achieve with the smallest number of code blocks?

A little light music

Turn your micro:bit into a photosensitive sound machine

You Will Need:

- micro:bit
- Earphones
- Paper
- Thin wire

Three blocks of code and an earphone are enough to create this musical instrument. Or perhaps you will use it as a directional sensor in a maze game. Its potential is up to your imagination. One of the delights of the micro:bit is the ability to explore complex ideas using simple software and hardware solutions. Use the MakeCode editor to create a program that would challenge a microPython or Arduino coder. Learn about the alternative functionality of LEDs and become a confident hardware hacker.

There are already guides to attaching headphones to the micro:bit and using the light sensor to create sound. This article will show you how to achieve impressive results with the minimum of code and a rather neat hardware solution.

Software:

Use MakeCode.

1. Start a new file. 'on start' and 'forever'

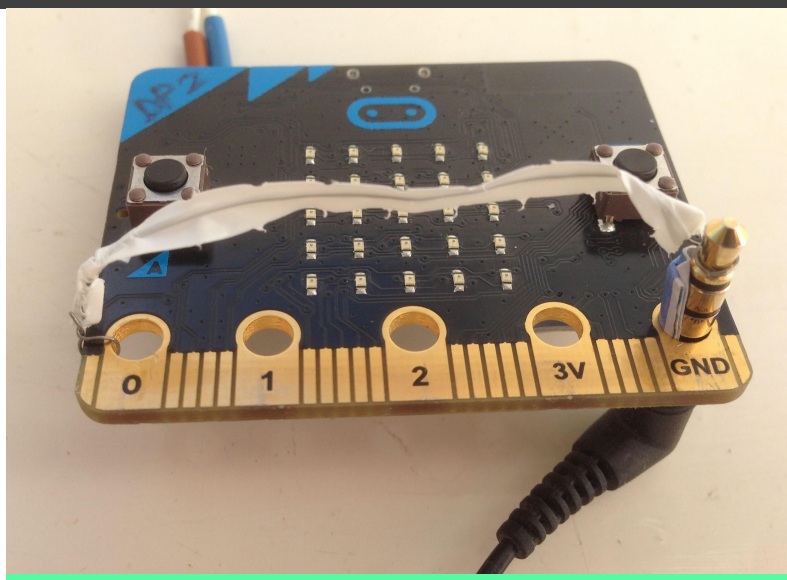
blocks are automatically created.

2. Drag the 'on start' block onto the menu; it becomes a bin; let go and delete it.
3. Click on Music menu; select 'ringtone (Hz)'; drag it into the 'forever' block.
4. Select Input menu; scroll to the bottom and select 'light level' block; place this block in 'ringtone (Hz)' block in the slot that currently reads 'Middle C'.
5. Save the project, open the save folder, copy the file to the micro:bit.

Hardware:

One of the prime features of the micro:bit is its 5 x 5 array of light emitting diodes. This allows low resolution but effective visual communication to be delivered without additional hardware. The designers have taken advantage of an interesting property of the LED; it will generate an electric current if a light is shone on it. It is the LED array that provides MakeCode' 'light level' reading: there is no need for an additional light sensor.

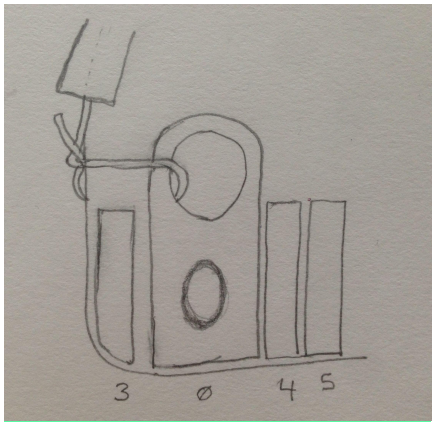
With the code in place, the virtual micro:bit on the MakeCode editor page shows Pin 0 and GND connected to the jack plug of a headphone. If you have crocodile clip leads then this is the quickest way to test the circuit but it would be a shame to miss out on hardware projects due to a lack of crocodile clips. A quick headphone socket can easily be improvised: indeed, it could be seen as a better option.



An early prototype using a food tie.

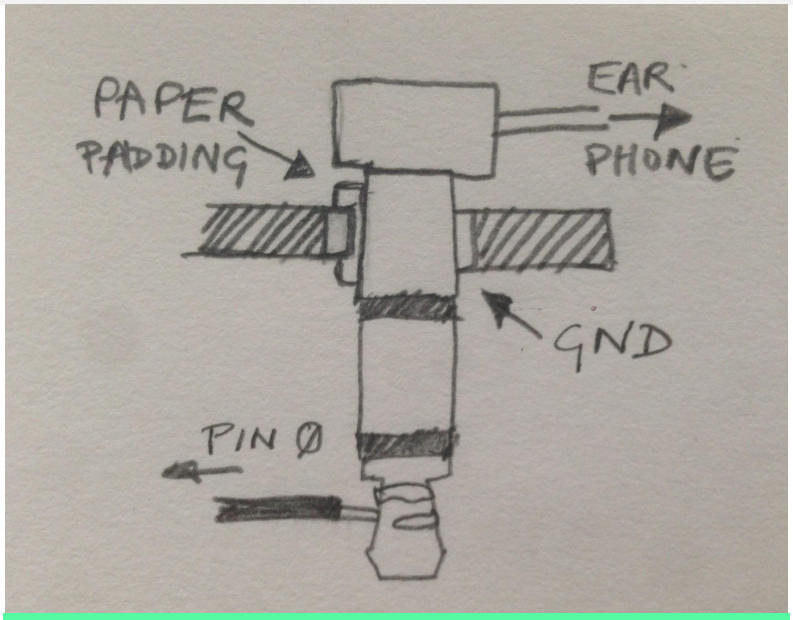
The large pin sockets are slightly larger than a headphone plug: so use a thin strip of paper, the width of the socket, and fold it back on itself two or three times. Use this strip to wedge the plug firmly into the GND socket. The paper will insulate part of the plug but there will still be enough contact for the circuit.

Connect a thin wire from Pin 0 to the tip of the headphone plug. If you do not have thin wire, try using a coated food tie. This is actually a good option; the wire is just the right gauge for the job. Strip 15mm of insulation from each end of the wire to allow a good connection.



Twist the wire tightly to the edge of Pin 0.

Start by attaching the wire to Pin 0. Put the wire through the hole to the side and twist it tightly: it should sit safely just above the small Pin 3 but you could cover that with tape if you wanted to be extra careful.



Finally, wrap the wire around the tip of the headphone plug.

Wrap the other end of the wire around the tip of the headphone plug: the tip has a notch that makes this task easy.

Action:

Power up the micro:bit and expose it to light to generate sound. The stronger the light, the higher the pitch. Use your hand over the LEDs to play the instrument. Torches with strobe effects are great fun. If you want louder noise then you can use the same method to connect the micro:bit to an amplifier using a suitable phono cable.

To experiment further, you could replace the MakeCode 'light level' input with 'rotation' or 'magnetic force'.

Pradeeka Seneviratne

Pradeeka is a technical writer, avid maker, and loves to design wireless projects, just like this Eddystone beacon.

@pradeeka7

Managing the micro:bit Eddystone Beacon Advertising URLs Remotely

Learn how to use Rebrandly Dashboard (third-party service) to update the advertising URL of an Eddystone beacon.

Update the advertising URL of your Eddystone beacon without re-programming the micro:bit.

You Will Need:

- BBC micro:bit
- micro USB cable
- Smartphone or tablet (with Bluetooth enabled) running Android OS
- Internet connection

In this article, you will learn how to update the advertising URL of your Eddystone beacon easily with the Rebrandly Dashboard (third-party service) *without re-flashing a new code* onto the micro:bit.

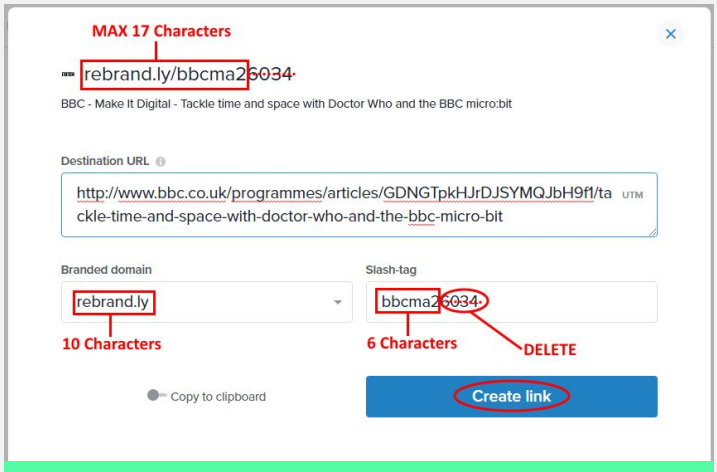
Getting Started

Prepare a list of URLs you want to advertise with your Eddystone beacon. For example, use the three URLs listed below;

- http://www.bbc.co.uk/programmes/articles/GDNGTpkHJrDJSYMQJbH9f1/tackle-time-and-space-with-doctor-who-and-the-bbc-micro-bit
- https://microbit.org/guide/safety-advice
- http://www.bbc.co.uk/programmes/articles/49tjW0qR05wXrdpK7ZbGTbs/strictly-micro-bit-live-lesson

Creating a Short URL

1. Create a free account with Rebrandly (https://www.rebrandly.com/).
2. In the Rebrandly Dashboard, click on the New link button. In the Rebrand, a new link modal box (window), paste the first URL from your advertising URL list. Once pasted, Rebrandly will automatically convert it to a *short URL* with the default domain, rebrand.ly. The Slash-tag can have a maximum of 6 characters. If you have got more than 6 characters for the Slash-tag, keep the first 6 characters and delete the rest of the characters in the Slash-tag text box. Then click on the Create link button to shorten the link.



Creating the short URL.

NOTE: An Eddystone advertising URL can have a maximum of 17 characters after the two forward slashes.

E.g. https://rebrand.ly/bbcma2

This includes the domain name, first forward slash (/), and Slash-tag.

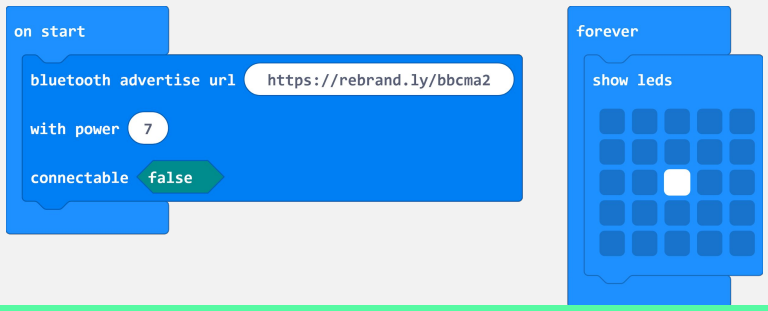
3. The notification will say Link rebranded.
4. Meanwhile, you will get another modal box (window). Click on the Copy button to copy the shortened URL to the clipboard of your computer.

Copying the short URL to the clipboard.

The Code

1. Go to the MakeCode editor for micro:bit (https://makecode.microbit.org/).
2. Under My Projects, click on the New Project.
3. In the Toolbox, click Advanced followed by Extensions.
4. In the Extensions page, click on the Bluetooth (Bluetooth services).
5. In the Some extensions will be removed modal box (window), click on the Remove extension(s) and add Bluetooth button. The Bluetooth (Bluetooth services) extension will add to the Toolbox.
6. In the Toolbox, click on the Bluetooth category. Then click and drag the Advertise URL block over and place it inside of the on start block.

7. Click on the default Bluetooth advertise URL (https://makecode.com) and paste the new URL you copied in the Step 5. Keep the default values as it is for power and connectable parameters.
8. In the Toolbox, click on the Basic category. Then click and drag the show LEDs block over and place it inside of the forever block. In the show LEDs block, click on the centre LED to select.
9. Once finished, your code should look something like this.



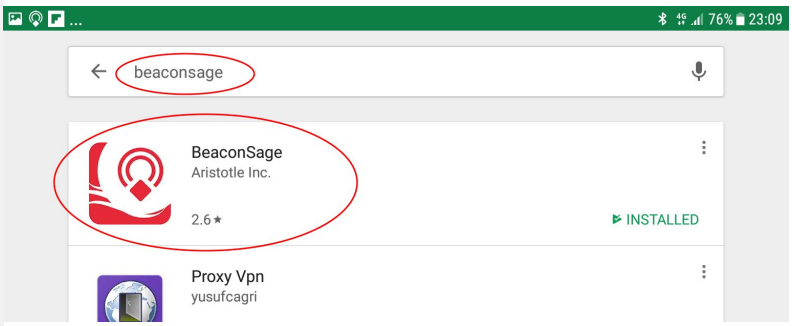
The code built with MakeCode for micro:bit.

10. Save your project with the name Eddystone-1. Once saved, a hex file will download to your computer.
11. Connect the micro:bit to your computer using the micro USB cable.
12. Drag and drop the hex file from your Downloads folder to the micro:bit drive. If you're using Google Chrome, drag it directly from the Download bar.
13. The micro:bit gets flashed with the hex file. After flash finishes, the centre LED of the micro:bit display will turn on.

Using BeaconSage App

1. Install the BeaconSage app on your Android smartphone or tablet. It scans and shows details about every nearby Eddystone beacons.

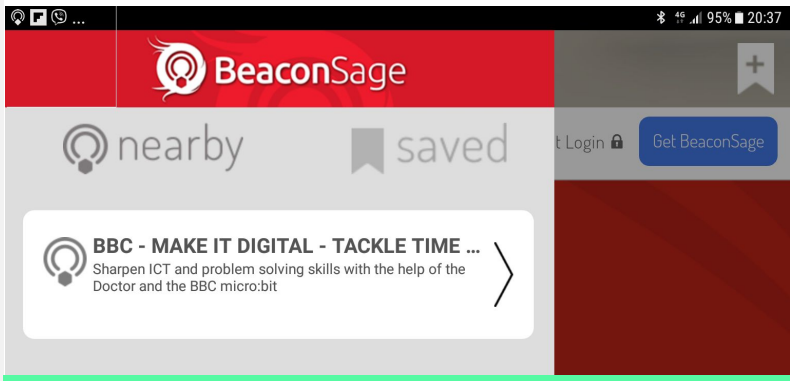
Warning: Turn the Internet on your Android device.



Installing BeaconSage app on Android

Warning: Turn Bluetooth on your Android device.

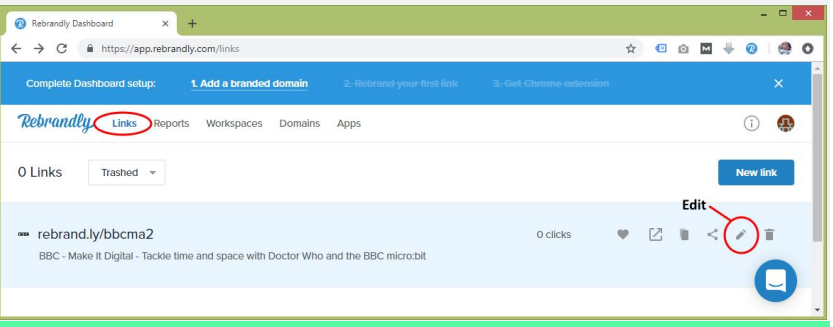
2. After installing, open the BeaconSage app. It will show the page title and the description belongs to the advertising URL of your Eddystone beacon. You can visit the web page by tapping on the left navigation menu under nearby.



Notification on the BeaconSage app.

Updating the Advertising URL

1. Now you're going to *update* the advertising URL. Choose the *second URL* from the above URL list (Step 1).
2. In your Rebrandly dashboard, click on the Links (https://app.rebrandly.com/links). Then click on the Edit icon.

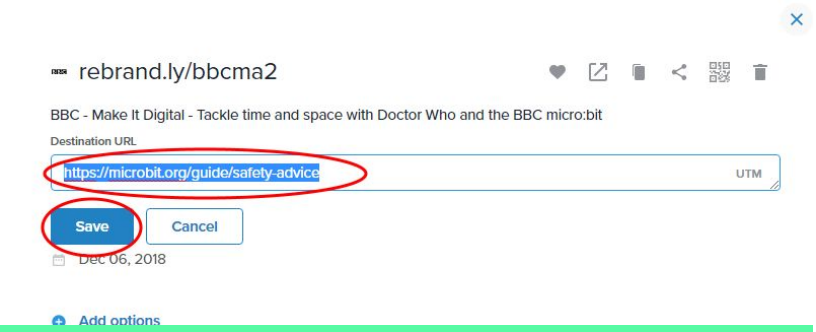


Edit button for editing the link.

3. In the modal box (window), click on the Destination URL.

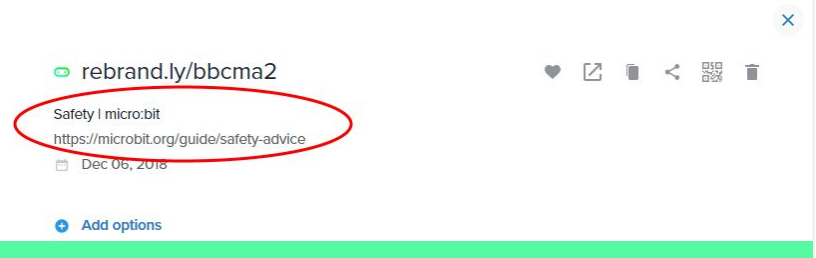
Editing the Destination URL – Step 1.

4. In the Destination URL box, paste the new URL. Then click on the Save button.



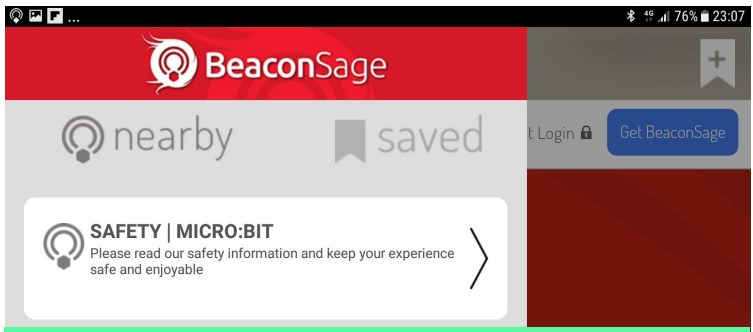
Editing the Destination URL – Step 2.

5. The notification says, 'Destination URL Updated'. Meanwhile, the modal box (window) reflects the details of the updated URL.



Updated Destination URL with the new link.

6. Now your Eddystone beacon will advertise the updated URL.



Notification on BeaconSage app with updated URL.

7. With the Rebrandly Dashboard, you can update the Destination URL unlimited number of times.

Do It Yourself

Update the destination URL with the third URL from the above URL list (Step 1).

How to make your own add-on boards with household materials

Tanya Fish

@tanurai

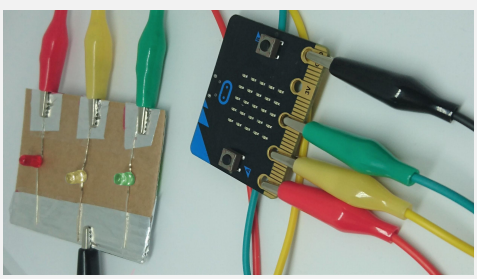
Tanya was formerly in schools, teaching maths, physics, & shouting loudly at paperwork. Pirate crew member since 2016 - making learning materials for schools, running workshops, & doing talks.

You Will Need:

- Piece of cardboard
- Silver foil
- Glue stick
- 3 x LEDs (red, orange, green)
- Sellotape

You may think that if you're starting out running a code club, you need a huge budget to buy a load of materials - the truth is, the hardware is, well, hardwearing and almost guaranteed to work, but sometimes a quick-and-easy homemade solution can work as a stopgap. I'd like to share some boards I have successfully made with Year 5's in my code club, and examples of what you can do with them. I have assumed you have micro:bits and crocodile clips.

Traffic Lights



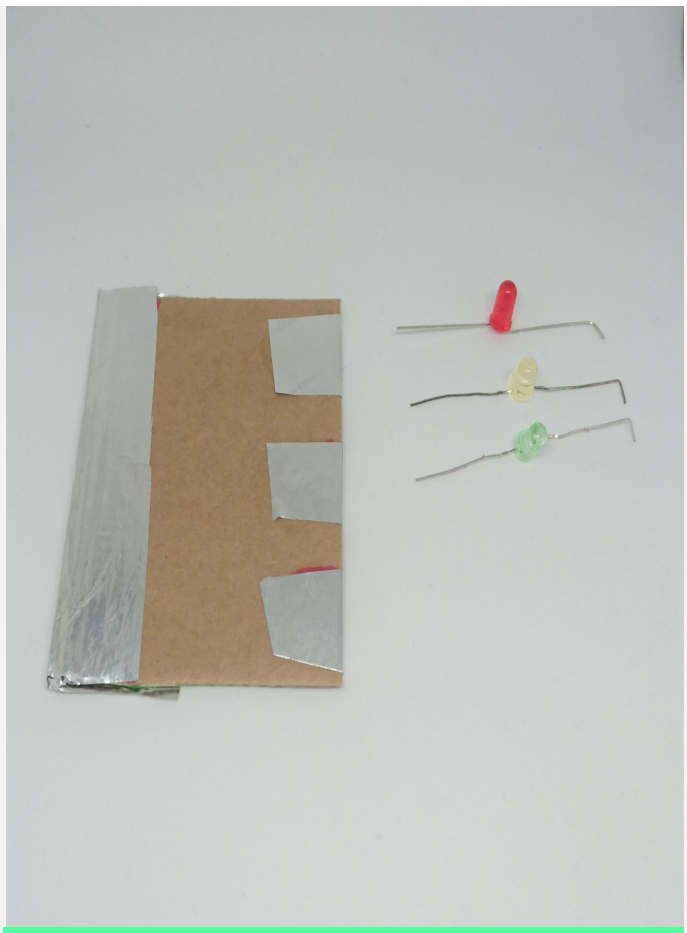
First, cut a piece of card about the size of a playing card.

Cut a strip of tinfoil the length of the card. Bend the legs outwards on the LEDs and give them a bit of a bend.

Card, foil and LEDs prepared for the next step

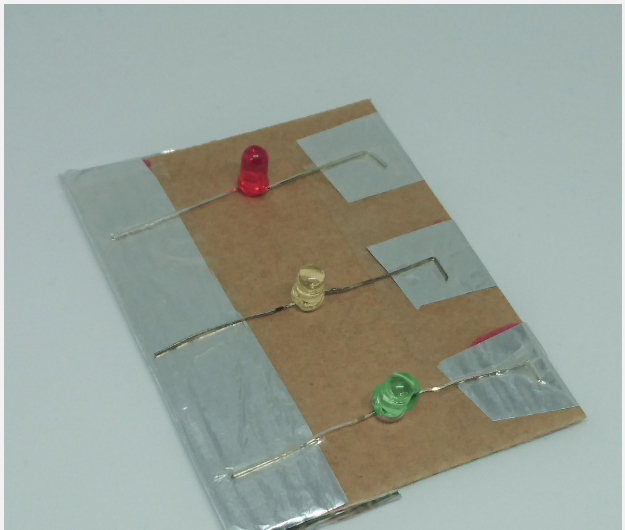
Glue the left third of the card as in the diagram, and smooth the foil down on it. Bend the foil around the back of the card and glue that down too.

Take some more of foil and glue three patches around the right-hand third of the card. Make sure they don't touch. Make sure they go round the back too.



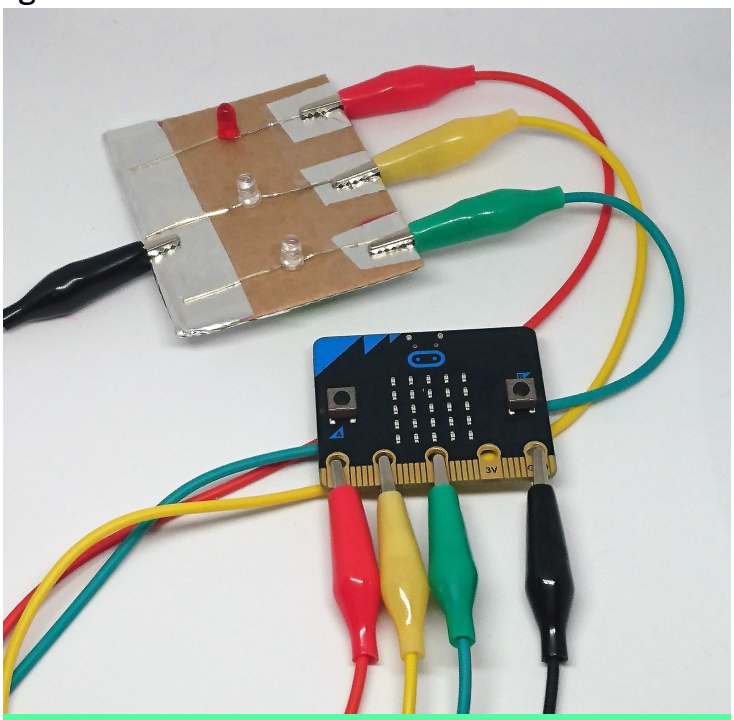
Front and back of the card with foil glued on

Now we're going to attach the LEDs. It is very important that the flat side of the LED (negative) is on the strip side, and the rounded side (positive) is on the patch side. Tape the LEDs down with one leg on the long strip of foil, and the other leg on a single patch, as shown in the diagram.



Tape down the LEDs, taking care to put them the right way around.

Finally, wire up to your micro:bit as in the diagram:



Use crocodile clips to attach your board to the micro:bit.

I used the following code to cycle through the pattern a real set of lights does. You might want to change the timings, or try to make different light patterns like running lights or a Knight Rider style Larsson scanner.

The red light is connected to pin 0, the yellow to pin 1, and the green to pin 2. Connect the ground strip with one crocodile clip.

Eun Jung (EJ) Park

EJ works on the Computer Science for All team in the NYC Department of Education. She loves making silly robots.

ejpark.com
cs4all.nyc

A High Five for Mr Boppy!

How to add unique user interactions to your project using micro:bit inputs.

Introducing Mr. Boppy. High five him, then he will open his candy vault! See him alive: https://vimeo.com/305829528

You Will Need:

- Cardboard/ Cereal box
- 1x Micro:bit go bundle kit https://www.adafruit.com/product/3362
- 1x Micro Servo Motor https://www.sparkfun.com/products/9065
- 3 x Alligator clips
- 3x Jumper wires
- 1x AA battery pack/ 4x AA batteries
- Masking Tape

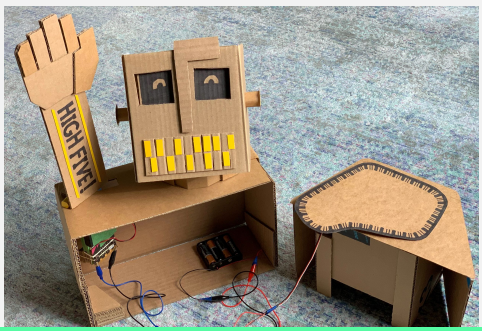
Mr Boppy is a little cardboard robot who offers candy from a secure vault when you give him a solid high five. The high five should be strong enough to make his head bop back and forth, which is his way of saying “I like your high five!” Using Mr Boppy as an example, I will challenge you to think of how you can use the

given micro:bit input options such as “shake”, “Tilt”, “Screen up” etc. to create a unique interface in your project. In this tutorial, you will also learn how to get started with a micro servo motor with a micro:bit.

How was Mr Boppy born?

I began this project by asking myself how I can create a micro:bit-powered robot that invites physical interaction. The micro:bits sensors allow it to detect different forms of inputs from its surroundings, such as shaking, tilting, button pressing, and changes in light. So how could I use these sensors to allow for unique human interaction? I found the answer by focusing on the project’s physical interface design.

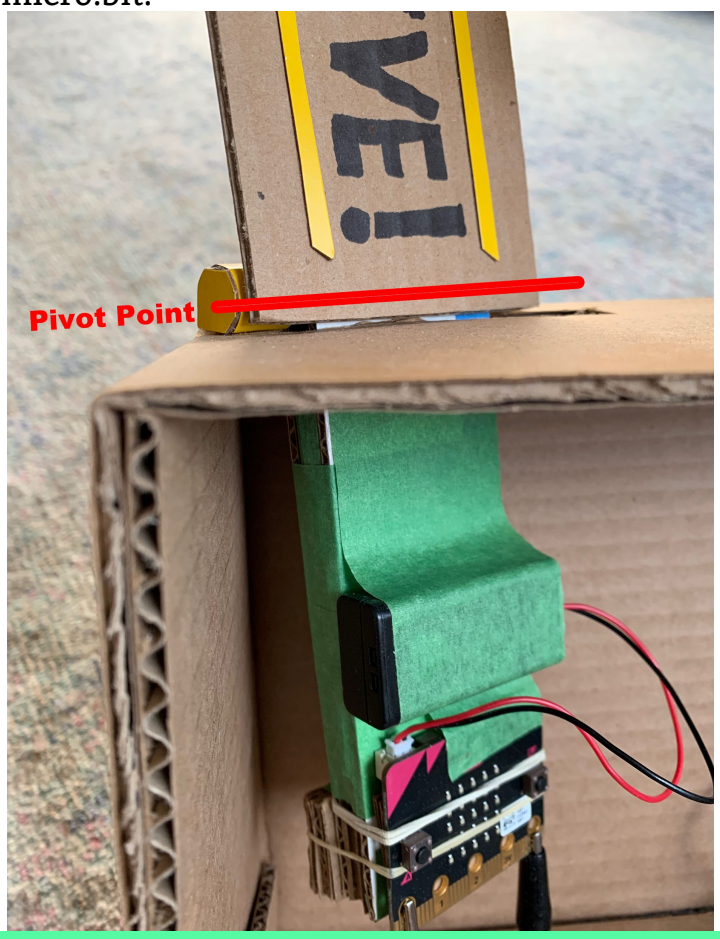
For example, if you wanted the physical interface of your project to capture the action of opening or closing a box, the tilting input that I described earlier can be used if the micro:bit is mounted on the lid. Similarly, tilting could also detect the act of “bowing your head” if a micro:bit is attached on your hat, which would make the way the hat attaches to your head the unique physical interface The “shaking” input can be applied in various creative ways, such as kicking, clapping, and jumping. In my design, it’s used to detect a high five.



Mr. Boppy has two parts: The body part with his arm and head and the candy vault with a lid that opens and closes.

Construction: Arm with a micro:bit

I started brainstorming the gestures that we observe in our daily lives and tried to recreate those gestures with a micro:bit. The high five was the clear winner as it is universally understood and fun! When you analyze the high five, you’ll see that involves slapping your hand against another person’s hand to make an impact before snapping it back again. I designed the arm in such a way that a micro:bit can interpret the impact from the high five as a shake. Think of it as a vertical seesaw: the higher part of the arm above the pivot point is where you high five and the part below is to where I attached the micro:bit. When a high five happens, the bottom part of the arm will make short swings back and forth and shake the micro:bit.



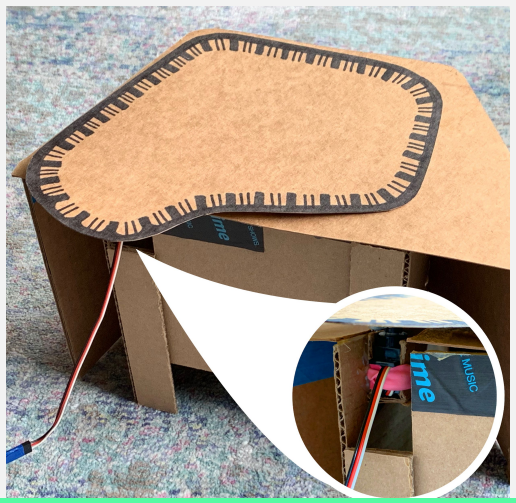
Divided by the pivot point, the top part of the arm is where you high five and the bottom part is where the micro:bit is attached.

Construction: Head

The impact from the high five will also make Mr Boppy's head slightly bounce and nod back and forth. This has nothing to do with a micro:bit or programming, it's purely for fun and character. His head is mounted on two springs and the physical force from the high five puts his head in motion.

Construction: Candy vault with a micro servo motor

I wanted the action of a high five to trigger a positive reaction for the user. That's how Mr. Boppy came to incorporate a candy vault. I used a micro servo motor to make a lid of the compartment rotate open and close.



The micro servo motor is placed on the body of the compartment and the lid is attached to the servo motor.

Most micro servo motors require 3v to 6v to operate. You may be tempted to have a micro:bit and servo motor share the same battery pack power source as it contains two AAA batteries (3V). The problem with splitting the battery pack between the two is that the motor may have enough power to move, but will be unstable and imprecise. For this reason, I connected an additional battery pack to the micro servo motor.

For additional instructions on how to do that, check my tutorial, Servo Motor with micro:bit. (https://demo.peblio.co/pebl/6nX293mr3)

Programming

Programming for this project is very simple. Basically, when the micro:bit is shaken, it needs to 1) move the micro servo motor to a certain degree (opening the lid), 2) pause for 5 seconds (holding the lid open) 3) move the servo motor back to its original position (closing the lid).

This is the code that makes the lid of the compartment open and close when the micro:bit is shaken.

When I tested my code, it was hard to tell if the micro:bit registered my high five as "shaking" or not, so I added a "show image" block to indicate that the micro:bit was shaken. In this way, I can easily pinpoint potential trouble. For example, if the icon is shown but the servo motor is not moving, then I need to troubleshoot my program and servo motor connection.

More to think about..

Hope you enjoy Mr. Boppy! It's now your turn to think about a unique interface design for your micro:bit project! How would you detect air currents with a micro:bit? How would you use a micro:bit to count the number of pinwheels you make? Think of fun and unique way with that people can interact with your micro:bit project!

Michael Rimicans

@heeedt

heeed.net

Michael has been tinkering with the micro:bit since it was released and using it for cool things. He is a STEM ambassador and Code Club volunteer

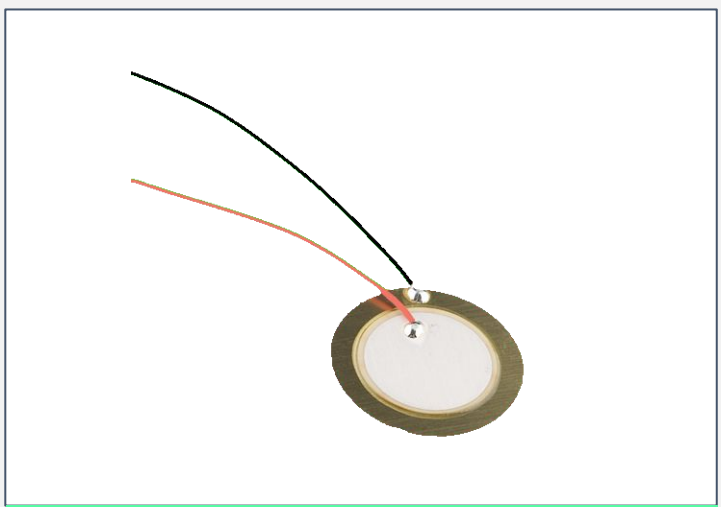
Nerf and micro:bits

You Will Need:

- micro:bit
- Piezo transducer
- Mu Python editor
- Nerf Gun

Nerf guns come in many sizes and playing with them is lots of fun. What would be even more fun though is having a set of targets to fire at and have a micro:bit keep track of the scores.

By itself a micro:bit is not able to sense being hit by a nerf bullet and requires a sensor to sense the impact. Whilst there are many types of sensor, a simple and cheap choice is a piezo transducer.



A transducer is a component that changes one form of energy into another, such as a microphone, which changes sound waves into electrical signals.

This type of transducer is normally built from a disc of ceramic material mounted on a brass disc. When this disc is deformed slightly a small electrical charge will be generated and, conversely, when a voltage is applied the disc will deform. When the voltage is removed the disc will revert to its normal shape whilst generating a slight sound.

The micro:bit is capable of sensing this small voltage generated by the transducer. The signal is then converted to a digital value by the Analogue to Digital converter (ADC for short).

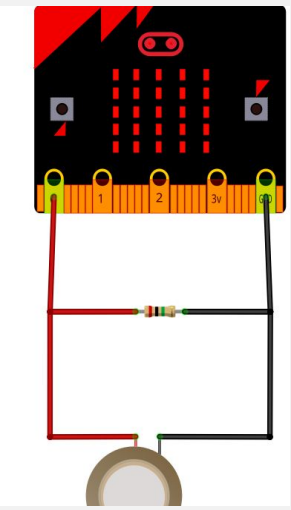
The micro:bit's ADC has a resolution of 10 bits meaning a 0v signal will be represented by 0000000000 and that a 3.3v will be represented by 1111111111. This means that it's possible to sense 1024 different voltage steps with each step being around 32mV which is the result of dividing the device's power, 3.3v, by the number of steps the ADC can count.

Putting this all together means that a transducer can be mounted on a target and any hits will be sensed by the micro:bit allowing a score to be counted.

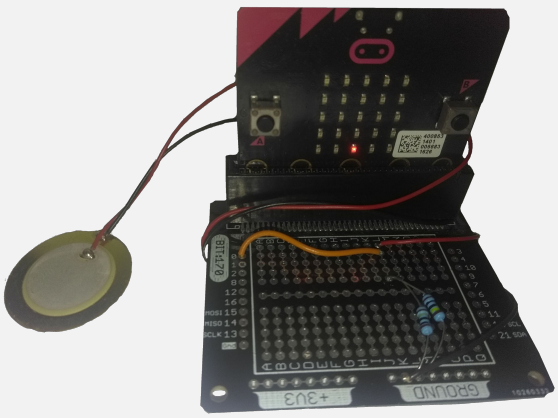
Construction

For this project, an old DVD case has been recycled to act as a target board with a bullseye type target printed onto paper and inserted behind the sleeve. On the inside of the case, the transducer has been taped in place with the brass side facing the target.

The transducer is then connected to the micro:bit using the following schematic.



The resistor in the circuit will keep the voltage at pin 0 at a steady level and allow the voltage from the transducer to be sensed more easily. This build uses a prototyping board with a built-in micro:bit socket but the circuit also be built using crocodile clips or a breadboard.



Writing the code

To sense the hits with the micro:bit, a simple microPython script, listed below, monitors pin 0 and when it breaks a threshold value it will display a smiling face on the micro:bit.

```
##beginning of code
from microbit import *

display.scroll("hello")
hitSenseLevel = 100

while True:
    hitSense = pin0.read_analog()
    print(hitSense)
    sleep(80)
    if hitSense > hitSenseLevel:
        display.show(Image.SAD)
        sleep(1000)
    else:
        display.show(Image.HAPPY)
```

Install mu-editor from here and enter the code. hitSenseLevel can be changed to adjust the sensitivity to the hits from the nerf bullets and when the transducer is hit with enough force to reach this level a sad face will be displayed and the script will pause for a second before resuming. If you switch to the REPL view in mu-editor and reset the micro:bit you will be able to see the actual values being read at pin 0 and the values achieved when the target is hit.

This script is quite simple so as a project you could do at home you could modify it to keep track of the number of hits.

Jackie Pease

Jackie is a maker who likes to have a go at everything. I'm especially interested in LEDs and wearable technology.
DoESLiverpool.com

Twitter: @jackie_pease @doesliverpool
Instagram: @jackiesees @doesliverpool

Beat-Reactive Drum Lights for Liverpool Light Night

Visualising the beats of different drums used by a Samba Reggae drum band, using micro:bits and WS2812 addressable LEDs.

Batala Mersey drummers performing at Liverpool Light Night

You Will Need:

Per drummer:

- Carnival backpack
- 2 × micro:bits per drummer
- 2 × AAA double battery packs for micro:bits
- 2m "NeoPixel" (WS2812B) RGB LED strip, 30 LEDs per metre
- 3 × D-cell battery
- 1 x battery case for 3 D-cell batteries
- 1 × multi-wire connector of your choice – we used Valcon 5.08mm PCB power connectors, because they had space for all 3 wires (Ground, Live, and Signal), they can be constructed with just a crimping tool, they're very sturdy, and, once in place, they're easy to connect and disconnect by hand.
- 3 × 40cm insulated multi-strand electrical wire (which we connected to one of the terminals of the battery box, and to the Pin 0 and Ground terminals of the micro:bit)
- 1 × 33cm insulated multi-strand electrical wire (connected to the *other* terminal of the battery box – it's shorter so that, when the wires are taped down, it'll be the same length as the 40cm one on the other side of the box)
- 3 × 100cm insulated multi-strand electrical wire (connected to the "NeoPixel" LED strip)
- Heavy duty sticky tape (fibreglass tape is ideal)

Assorted tools, including:

- Wire cutter
- Wire sheath/insulation removal tool
- Solder and a soldering iron

For anyone who hasn't heard one, Brazilian-style Samba Reggae bands can be extremely loud, and it can be difficult to work out what the different drums are adding to the mix.

Batala Mersey, a Liverpool-based drum band, and Brouhaha International, who organise Carnival events and make costumes, came up with a plan to use coloured LED lights to show which drums were playing at any one time. They came to our Makerspace at DoES Liverpool for help to bring the project to life in time for Liverpool Light Night on 18th May 2018.

Uma from Brouhaha International got in touch with me in early 2018 to ask if DoES Liverpool could help with a project to create light-up backpacks for the Batala Mersey drum band who were taking part in Liverpool Light Night, a free one-night arts & culture evening when museums, art galleries and other venues stay open late and there are lots of indoor and outdoor performances.

We met up with Dave and Ilsa from Batala Mersey. The band has 4 different types of drums (they use the Brazilian names for the different types). 12 of the drummers would have backpacks:

- 2 x Repinique
- 2 x Caixa (snare)
- 4 x Dobra (medium size drums)
- 4 x Surdo (large bass drums)

Mandy and Uma from Brouhaha International measuring out LED strips

There were 2 groups of Surdos as they play alternate beats, so that meant 5 different groups altogether. We decided that each group's backpacks would light up in a different colour. I returned to DoES Liverpool with one large Surdo and a smaller Repinique for us to experiment with.

Another member of DoES Liverpool, Zarino, and I had been using micro:bits and WS2812 LEDs at Liverpool Code Club. Zarino quickly realised that the micro:bit's inbuilt accelerometer would be a great way to tell when the drum was being hit. We attached a micro:bit to the side of the drum and wrote a simple MakeCode program to read in the values and display them on the micro:bit's LED display. We found that the two different drums we had output different ranges, so all looked good.

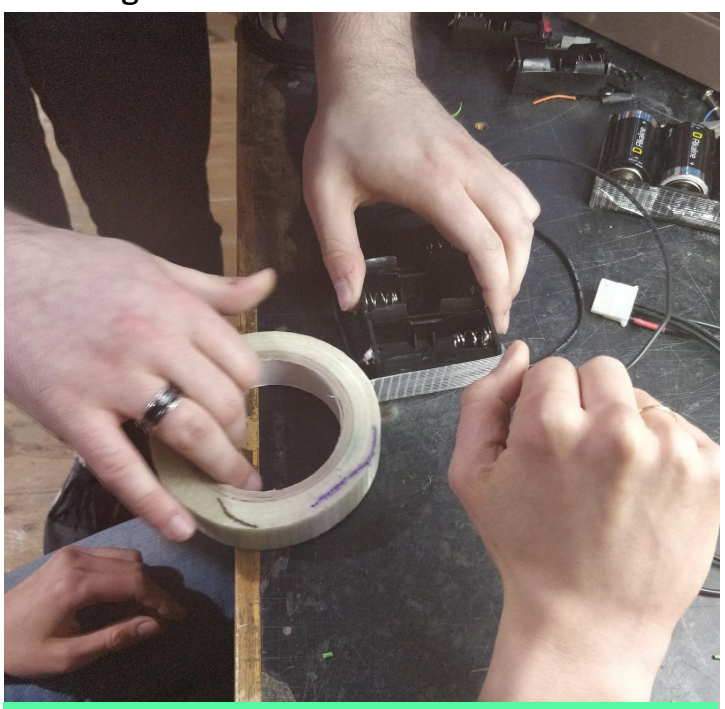
Denise from Liverpool Libraries was happy for us to borrow the library's micro:bits so we were

able to use two per drummer, one attached to the back of the drum with duct tape and one in a pouch attached to the backpack to control the lights. We used micro:bit radio (Bluetooth) to communicate between the two, with each pair of micro:bits being assigned a separate channel. The drummers move around a lot when they're performing, so it was good not to have wires between the drum and backpack.

Mandy sewing LED strip onto one of the disks. The disk consists of a wire ring with reflective fabric hot-glued onto it. These were then attached to the backpacks.

Dave invited us to one of Batala Mersey's practice nights. Everyone has to wear earplugs to protect their ears as the drums are really loud. We couldn't hear each other talking, and we had to type stuff in on our phones to communicate. We had changed the drum program so that we could use the micro:bits buttons to adjust the range for each drum, and were able to get some good results, although there was some interaction between the different drums – beating one drum could make the skin of another vibrate, even if it was at the other end of the room.

Separately, we worked with Uma and Mandy from Brouhaha on the carnival backpacks. Ilsa had come up with an idea to have a flat disk behind each drummer's head, with the LEDs forming a spiral on the disk. The backpacks are made from light aluminium tubing and normally decorated with lightweight elements like sequins and feathers. Although the LEDs weren't heavy, Mandy had to find a thicker than usual fabric for the disks to make sure that they didn't rip out of the fabric – it also needed to be reflective to make the most of the light. She had to source pouches that could be attached to the back of the backpacks to carry the micro:bits and batteries too – she ended up using insulated lunch bags!



Paula and Zarino taping together battery holders for the D-cell batteries

We ordered all the parts we needed to connect the micro:bits, batteries and lights and were

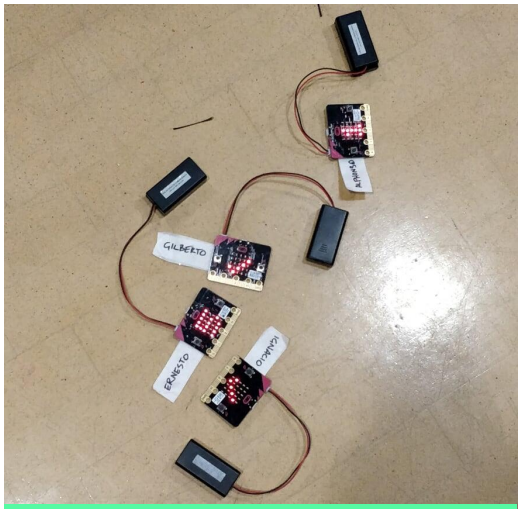
able to draw on a team of volunteers at DoES Liverpool to help us with the electronics, including Mike, Adrian, Paula, Paul H, and Chris T. Then it was up to Mandy to sew on the LEDs and decorate the backpacks.

Soon it was dress rehearsal night. We attached the electronics to the newly decorated backpacks and then attached the backpacks to the drummers! It took some of them a bit of time to get used to drumming while wearing a backpack. One thing we hadn't taken account of was that some of the signals we were getting were caused by sound reflecting off the walls of the practice hall. Zarino ended up going along to one of Batala Mersey's outside performances and taking measurements to make sure that there wouldn't be any issues.

Collars and belts that form part of the drummers' outfits. Mandy uses hot glue to attach the braid to the fabric.

On Light Night, Batala Mersey had 3 sets. Our lights were only involved in the third, at 10pm (as it was late May, it wasn't dark enough until then). We did have a few problems with some of the connections between the batteries and lights, but fortunately, we'd brought along extra supplies for any emergencies, and we were able to replace them temporarily with crocodile clip test leads.

The band's performance in front of a large crowd outside the library was fantastic and the lights flashing in time to the rhythm of the different drums really added to the effect.



micro:bits waiting to be attached to drums at the dress rehearsal. They were given names so that we could easily match them up with the correct backpacks.

We enjoyed the project and learnt a lot. There are some things we wouldn't do again, like using heavy D-cell batteries – phone power banks are getting really cheap, are lighter and rechargeable, so we'd use those, and we would look again at the connectors we used. If you do want more information on what we did, then there's a Git repository here: https://github.com/liverpoolcodeclub/drum-lights (it also contains some useful work by Zarino on developing and deploying MakeCode to a large number of micro:bits).

We're hoping to work with Batala Mersey, Uma and Mandy again in 2019, and to develop new projects that combine light and sound.

justaboutfine

@justaboutfine
jafine.github.io

Justaboutfine has been helping run a Code Club in Australia and blogs about it at various times, he's done some technical review for a couple of No Starch Press books, and has written projects for Code Club Australia. He likes open source, hacking things, and is involved in various communities and conferences, and does things for https://smalldevices.com.au

Two-way radio emoticon communicator

Here's a simple MicroPython script that lets multiple micro:bits wirelessly send emoticons to each other!

You Will Need:

- 2 or more micro:bits each with power, and a cable for programming
- Access to a suitable MicroPython editor, e.g. the online micro:bit Python editor, Mu editor, or Chrome plugin editor.

This fun MicroPython script uses a dictionary to map a number to each emoticon image, and sets a wireless channel, to broadcast and listen on. The main loop lets you use the A and B buttons to select an emoticon, and pressing both at once sends it on the selected wireless channel. Every other micro:bit in range, running the same script, will then receive the emoticon!

Let's get coding!

In our editor, let's begin writing the script and import the required micro:bit and radio modules. Add the following to a new file in your editor.

```
from microbit import *
import radio
```

Put the emoticons into a dictionary

The micro:bits image class has built-in images that we can associate with numerical keys in a dictionary structure. We'll create a dictionary called *images*.

```
images =
{1:Image.HEART,2:Image.HEART_SMALL,3:Image.HAPPY,4:Image.SAD,5:Image.SURPRISED,6:Image.ANGRY,7:Image.ASLEEP,8:Image.BUTTERFLY,9:Image.DIAMOND,10:Image.CONFUSED,11:Image.COW,12:Image.PACMAN}
```

Create an index

We'll use a variable to keep track of which emoticon image we have 'selected' and set this to

Two-way radio emoticon communicator


```
index_num = 1
```

Configure and enable the radio

To set up the radio functionality we need to set the radio channel to 10 and turn the radio on.

```
radio.config(channel=10)
  radio.on()
```

Main loop and receive radio data

Now we come to the main loop that keeps repeating while the script is running. Received radio communications will be captured in the *incoming* variable. To keep things simple, we haven't done any validation on the incoming radio data - you should be aware that this is reading unchecked radio data into a variable, and a rogue radio transmission could potentially exploit this. However, for the sake of this exercise, let's assume you're well away from such transmissions and possibly in a Faraday cage or bunker of your choice ;)

```
while True:
      incoming = radio.receive()
```

Do things when events occur

The rest of the script is essentially saying "if something happens, do this" followed by displaying the select image and a delay. Without the delay, the micro:bit was checking the buttons too often, making it hard to read a single press. If you find that you have to hold down the buttons too long, you can reduce the *sleep()* delay value to make it a little quicker. If it flickers between different images when you press buttons, increase the delay.

Check for single button presses, and increase or decrease the *index_num* value. Since the *if* statements are inside the *while* loop, we need to make sure they are indented (moved right using spaces) so they line up with the incoming line above.

```
if button_b.is_pressed():
      index_num += 1
if button_a.is_pressed():
      index_num -= 1
```

Send the current *index_num* value, and display 'sending...', if button A and B are pressed together. The *str()* function converts the number into a character to send. We could really send raw bytes here let's keep it simple for now.

```
if button_a.is_pressed() and
button_b.is_pressed():
         radio.send(str(index_num))
         display.show('sending...')
```

If we have an incoming radio broadcast, display the TARGET image, wait for half a second, then convert the incoming data back to a number.

```
if incoming:
         display.show(Image.TARGET)
         sleep(500)
         display.show(images[int(incoming)])
         sleep(2000)
```

Make sure that *index_num* stays within the key values for the images dictionary. In this case, if *index_num* is too high, we set it back to the first image; and if it gets too low, we set it to the highest image index.

```
if index_num > 12:
      index_num = 1
    elif index_num < 1:
```

```
    index_num = 12
```

Finally, display the currently selected image from images and wait for half a second.

```
display.show(images[index_num])
sleep(500)
```

The entire script

So, the whole script should look like (I've added comments, so don't worry about any lines starting with #:

```
# A micro:bit emoticon chat script
# By @justaboutfine
from microbit import *
import radio

# Create a dictonary of our emoticon images
images =
{1:Image.HEART,2:Image.HEART_SMALL,3:Image.HAPPY,4:Image.SAD,5:Image.SURPRISED,6:Image.ANGRY,7:Image.ASLEEP,8:Image.BUTTERFLY,9:Image.DIAMOND,10:Image.CONFUSED,11:Image.COW,12:Image.PACMAN}
index_num = 1

# Set the radio channel to 10
radio.config(channel=10)
radio.on()

while True:
    # Capture received radio data
    incoming = radio.receive()

    if button_b.is_pressed():
        index_num += 1
    if button_a.is_pressed():
        index_num -= 1

    # Send the current emoticon if both buttons pressed together
    if button_a.is_pressed() and button_b.is_pressed():
        radio.send(str(index_num))
        display.show('sending...')

    # If there's incoming data, show the emoticon that's been received
    if incoming:
        display.show(Image.TARGET)
        sleep(500)
        display.show(images[int(incoming)])
        sleep(2000)

    # Keep the index_num within the valid dictionary key range
    if index_num > 12:
        index_num = 1
    elif index_num < 1:
        index_num = 12

    # Show the current image
    display.show(images[index_num])
        sleep(500)
```

Now flash this to two or more micro:bits and you should be able to use the controls listed above, to select and send emoticons from one micro:bit to another. On the Mu editor and Chrome plugin, you can just click on 'flash', with other editors you can compile your project to a *.hex* file and drag or copy it to your micro:bit once it is plugged in. At the time of writing, flashing via BLE doesn't work if you're using Python. Also, using the radio in other languages can interfere with BLE functionality. If you have a group, you can send to more than one micro:bit. You can set separate radio channels on different pairs of micro:bits to have separate teams or emoticon conversations. From here, you could even use this to send text messages to one another as well, or use *speech()* to 'say' a message on another micro:bit equipped with a speaker. The possibilities are fairly broad, so go and experiment!

Chris Penn

Chris is a teacher from Warwickshire who teaches Computer Science and IT. He also runs the Coventry and Warwickshire Raspberry Jam.

@ChrisPenn84

A scalable cube of TNT controlled by the micro:bit that you should see by pressing 'b' and using 's' on the keyboard to walk back so you can see the cube you have created.

'For Scalable blocks press b'

In this simple tutorial, you will use the BitIO python library to create a scalable block of your choice in Minecraft.

You Will Need:

- 1 x micro:bit
- 1 x USB cable to power the micro:bit
- 1 x Raspberry Pi / Java edition version of Minecraft
- 1 x copy of the BitIO library which you can download from here: link

You will create code that allows you to create a scalable cube of a block of your choice. Press 'a' to increase the size of the cube and 'b' to build it. Press 's' on the keyboard to walk back and admire your creation then watch it disappear a few seconds later.

1. Turn on your Raspberry Pi and make sure that you are connected to the internet.
2. Download the BitIO library as a zip file from here: link
3. Right click on the zip file and extract it.
4. Open Minecraft and create a new world, then minimise this for later. Tip, use the tab key to allow your mouse to minimise the Minecraft window.
5. Click on the Raspberry Pi logo top left.
6. Then select 'programming' from the menu and then Python 3.
7. Click on 'file' > 'new file'.
8. Type out the following code:

‘For Scalable blocks press b’ :make

```
"""
Inspiration Taken from the original idea by Craig
Richardson in his 'Learning to program with Minecraft'
available from Amazon.
book.

"""

import microbit
from mcpi import minecraft as minecraft

from mcpi import block as block
from datetime import datetime
import time
import serial
import random
mc = minecraft.Minecraft.create()

def bigBlock(Increase,block):
  pos = mc.player.getTilePos()
  mc.postToChat("Build")
  mc.setBlocks(pos.x, pos.y, pos.z,pos.x+Increase,
pos.y+Increase, pos.z+Increase, block)
  mc.postToChat("Wait")
  time.sleep(5)
  mc.postToChat("Clear")
  mc.setBlocks(pos.x, pos.y, pos.z,pos.x+Increase,
pos.y+Increase, pos.z+Increase, 0)


Increase = 0
pos = mc.player.getTilePos()
while True:

  if microbit.button_a.was_pressed():
    Increase = Increase+1
    msg = "Size = 1 block+"+str(Increase)
    mc.postToChat(msg)

  if microbit.button_b.was_pressed():
    bigBlock(Increase,46)

  time.sleep(1)
```

9. Click ‘file’ > ‘save as’…. Then find the location of your folder that you extracted earlier. Name the file ‘cube.py’

10. Save this file in the ‘src’ folder inside the BitIO folder.

11. Next in idle press f5 to try and execute your code.

12. If you have any errors check through your code and see if you can spot any mistakes and repeat step 11.

13. Once your code is working, here is the process you will need to go through to get it working with the micro:bit:

14. Plug in your micro:bit.

15. First of all, it will display this message:

No micro:bit has previously been detected Scanning for serial ports remove the device, then press ENTER

16. So unplug the micro:bit and press enter.

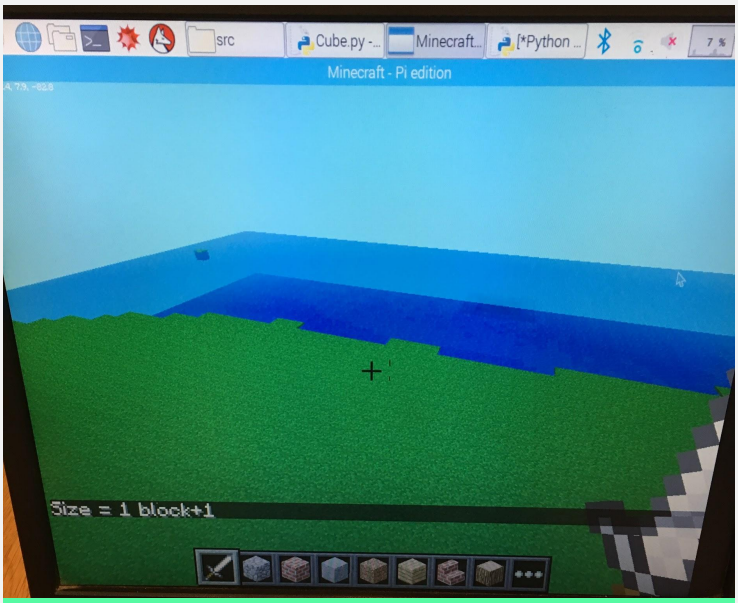
17. Next, the following text will display:

Scanning… found xx device(s) plug in the device, then press ENTER

18. Now press enter and the following text will display Scanning… found xx device(s) found 1 new device selected:/dev/ttyACM0

Do you want this device to be remembered? (Y/N)

19. Press 'y' and enter to confirm and the following text will display: Your micro:bit has been detected.
20. Now your program should be working. If you put Minecraft back on-screen and press the ‘a’ button on the micro:bit, you should see the image below.



If your program is running correctly, in Minecraft, you should see the following.

21. The block starts at 1 x 1 so by adding one it will be a 2 x 2 cube.
22. To see the 2 x 2 cube press ‘b’ on the micro:bit.
23. On the keyboard press ‘s’ to walk back to be able to see cube you have created.
24. It will be there for 5 seconds and then will disappear.
25. It should appear as shown in the image at the beginning of this article.

Well done, you are a successful coder :)

Extensions:

1. Try and change the block type.
2. Try changing the size of the cube created
3. Try changing the position of the player when the block is placed.
4. Try making the TNT active by adding a ‘1’ at the end of the following line of code

```
mc.setBlocks(pos.x, pos.y, pos.z,pos.x+Increase,
pos.y+Increase, pos.z+Increase, block, 1)
```

Mr. Zbit

Is a micro:bit enthusiast who writes code and invents accessories for the micro:bit designed to help everyone 'Have fun while you learn!'

@ZbitConnect
zbit-connect.co.uk

Speech Recognition on a micro:bit

Part 1 - The Challenge of Speech Recognition and Building the Electronics

You Will Need:

- 1 x Adafruit ADA1713 Electret Microphone Board
- 1 x Schottky Diode – e.g. 1N5817
- 1 x 47uF Aluminium Electrolytic Capacitor
- 1 x 10k Resistor
- 1 x Breakout & Prototyping Board – e.g. zbit:breadboard

For years we've communicated with computers by typing on a keyboard and clicking a mouse. Then we started communicating with them in other ways such as touching them, tilting them and shaking them. But until recently we've not communicated with them in the main way we communicate with other people. We've not spoken to them! Now that is changing. Early attempts at speech recognition included telephone systems where you could ask for a ticket to the cinema in Carlisle and it would respond by saying "Did you say Wolverhampton?" But more recently the technology has improved to give us speech control of our mobile phones with features such as "Siri" and digital assistants that sit in the room with us that we talk to such as "Alexa". And it could be that in the future speaking to computers becomes the main way we

communicate with them! So why has it taken so long for us to be able to speak to computers? The answer is that speech recognition is *very* difficult for a computer! But why is speech recognition so difficult? What are the challenges that a speech recognition 'algorithm' must deal with? And, can we do speech recognition on a micro:bit?

The micro:bit has limited processing power and limited memory so doing "Alexa" levels of speech recognition is not going to be possible! But just as the micro:bit's limited display of just 25 LED's forces us to be creative as to how we display text and images, by being equally creative there ***is*** a way to do speech recognition on a micro:bit and this article shows you how!

The article is in two parts...

Part 1 describes the **Challenge of Speech Recognition** and shows you how to **Build the Electronics** and test it works as a ***micro:bit Volume Meter***!

Part 2 describes the **Challenges faced by Speech Recognition Software** and shows you how **Code you own Speech Recognition Algorithm** in ***MicroPython*** to turn your ***micro:bit Volume Meter*** into a ***micro:bit Speech Recognition Device***!

The Challenge of Speech Recognition

Speech, when captured by a microphone, is converted into a complicated analog waveform containing a jumble of frequencies ranging from below 100Hz to over 10kHz. To accurately capture such a waveform and convert it to digital 'samples' ready for a Speech Recognition to be sampled, according to the ***'Nyquist Rule*'***, at twice the maximum frequency of the sound. This would mean making over 20,000 samples a second, or one sample every 50 ***microseconds***! Theses sample must then be processed in ***'Real Time*'*** by algorithms such as the ***'Fast Fourier Transform*'***. This normally requires specialist ***'Digital Signal Processor*'*** chips which have built-in hardware blocks such as ***'Multiply Accumulate Units*'*** and ***'Neural Networks*'*** to speed up the processing. "***Alexa***" may have such chips but the micro:bit does not! So, what can a micro:bit capture? Let's see!

[*An explanation of the meanings of ***'Nyquist Rule'***, ***'Real Time'***, ***'Fast Fourier Transform'***, ***'Digital Signal Processor'***, ***'Multiply Accumulate Units'*** and ***'Neural Networks*'*** is beyond the scope of this article, but look them up on ***Wikipedia*** if you'd like to know more!]

Elements of the Spoken Word

Spoken words contain 3 basic elements. ***Volume***, ***Duration*** and ***Frequency Content***.

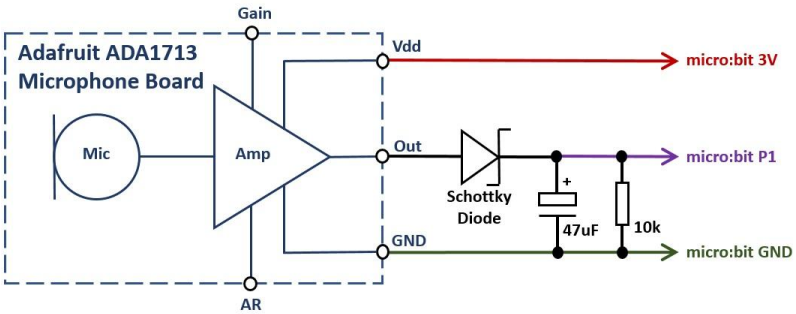
- ***Volume*** is how loud each word sounds
- ***Duration*** is how long each word lasts
- ***Frequency Content*** is what combination of audio 'pitches' each word contains

By adding just 3 electronic components to the output of the Adafruit Microphone board it is possible to create an analog waveform representing the ***Volume*** of the speech. Since this volume waveform changes more slowly it can be sampled by code at a rate of about one sample every 50 ***milliseconds***, or 20 samples per

second, which ***is*** possible on a micro:bit running Micro-Python! So, we'll be able to capture samples of the ***Volume*** of each word on the micro:bit, and by counting the samples we'll be able to calculate the ***Duration*** of each word. We will, however, have 'lost' the ***Frequency Content*** of the words! But this is the compromise we must make!

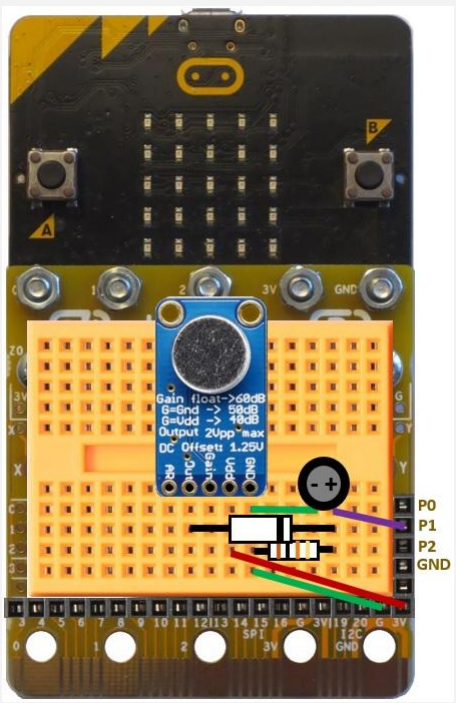
The Electronic Circuit

The circuit diagram below shows how the **Schottky Diode**, **Capacitor** and **Resistor** are connected to the **Adafruit Microphone Board** and how they connect to the **micro:bit**.



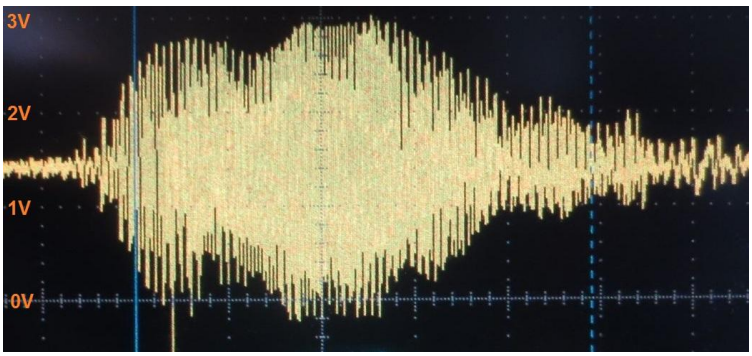
Wiring up the Breadboard

The diagram below shows how to wire up the microphone and other components using **zbit:breadboard**. The **Capacitor** and the **Schottky Diode *must*** be plugged in orientated as shown. The **Resistor** can be plugged in either way round. There are no connections to the 'AR' and 'Gain' pins of the Adafruit board.



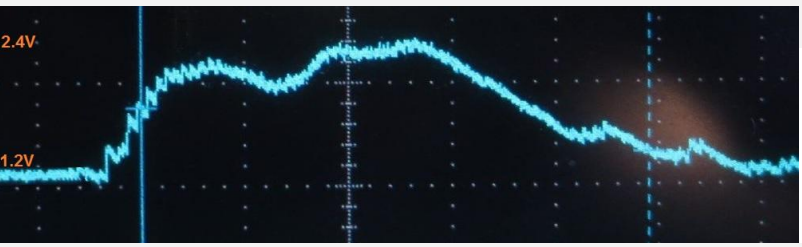
Understanding the Electronics

When you speak into the microphone the analog output of the microphone board (**Out**) will have an alternating voltage with a jumble of frequencies representing your speech with voltages that swing between 0 Volts and 3.0 Volts as shown by the Oscilloscope waveform below.



When the voltage waveform (**Out**) is at its peak the Schottky diode will be 'forward biased' which means the current will flow through the diode and charge up the voltage on the capacitor. When the voltage waveform is at its trough, the Schottky diode will be 'reverse biased' and since current can't flow back through a diode, the capacitor will maintain its voltage. When there is silence however there will be no output from the microphone to keep the capacitor charged so the capacitor will slowly discharge through the 10k resistor. Typically, it will take about 50 milliseconds to charge up the capacitor when you start speaking into the microphone and about 150 milliseconds for the capacitor to discharge when you stop speaking.

This results in the capacitor's voltage looking like the Oscilloscope waveform below.



This waveform represents the **volume** of the spoken word. The micro:bit can now sample this voltage. Since it is an analog voltage it must be connected to one of the analog GPIO pins on the micro:bit (P0, P1 or P2). In Part 2 we will be using P0 to output sound effects, so we will connect it to **P1**.

The volume can be read using the Micro Python command:-

volume = pin1.read_analog()

This will read the analog voltage and store it in a variable called '**volume**'

Testing the Electronics

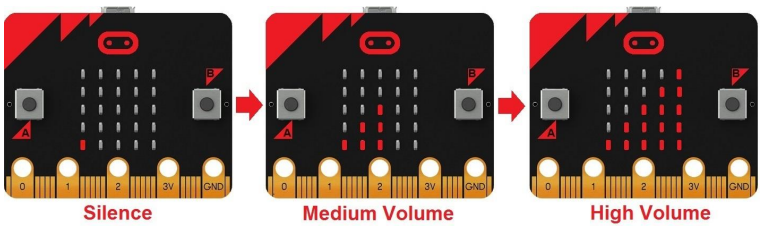
The Micro Python command **'volume = pin1.read_analog()'** measures the voltage on GPIO **P1** and returns a value between **0** (representing **0 Volts)** and **1023** (representing **3.3 Volts)** into the variable **'volume'**. The output voltage will be about **1.2 Volts** when there is silence up to about **2.4 Volts** when it detects a loud sound, hence the value loaded into the variable **'volume'** should be about **400** when there is silence up to about **750** when you talk loudly into the microphone.

To test your electronics, download the program **Python-Volume-Meter.py** from:-

www.zbit-connect.co.uk/SpeechRecognition

and load it onto your micro:bit using the micro:bit's online Python editor or using the 'Mu' Python editor.

If everything is working you should see your speech volume represented by a triangle of LED's on the micro:bit's display from one LED when there is silence, up to 15 LED's when you talk loudly into the microphone.



Since your circuit may have slightly different voltages, read the comments in the code to help you make the necessary adjustments.

You now have a working *micro:bit Volume Meter*!

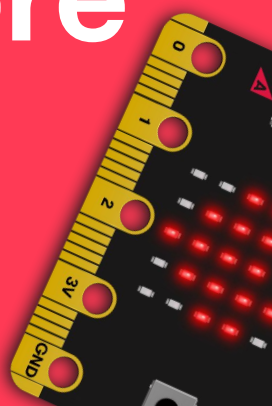
In **Part 2** we will look at the **challenges faced by the Speech Recognition Software** and show you how to **code your own Speech Recognition Algorithm** in ***MicroPython*** to turn your ***micro:bit Volume Meter*** into a ***micro:bit Speech Recognition Device*** and in doing so you will have to deal with many of the challenges that ***all*** speech recognition algorithms must overcome!

Advertise in micro:mag!

micro:mag is the community magazine for micro:bit lovers - if you want to reach an audience of students, teachers and hobbyists at reasonable cost, micro:mag is the place to do it! We've got full and half pages available, at reasonable rates, and you'll help cover the costs of producing the magazine.

Get in touch for more info - email us on

hello@micromag.cc



Create your own flood detector!

Make your very own micro:bit flood detector in 9 easy steps

Les Pounder

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

@biglesp bigl.es

The risk of flooding has increased in places across the world. But what if we need to alert people to an oncoming flood? How can we detect when water reaches a certain level? Well, we can and all we need are a micro:bit, two crocodile clips and some aluminium foil. This simple project uses the conductivity of water to close a circuit when water touches the two foil sensors. For this, we read analog values, ranging from 0 to 1023 which relate to 0V to 3V. Anything over 1000 will trigger our project to life.

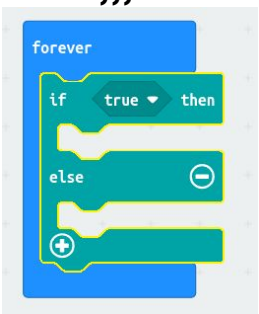
Step 1: Startup Screen

When our micro:bit starts up, we need to see that it is ready to go and we use the “Show Icon” block from Basic. In this case, we chose to use a tick to indicate readiness.

Step 2: Slow down!

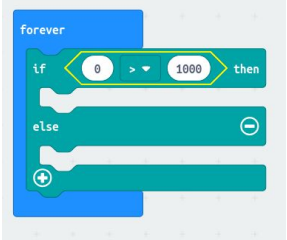
We need to slow down the code, using “pause” from the Basic menu we set the delay to 1000ms, which is 1 second. This enables us to see the tick before the main code runs.

Step 3: Forever,,,and ever..and ever!



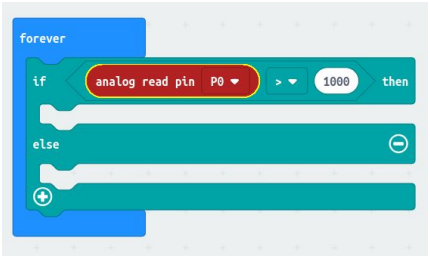
Inside our forever loop we need to place an “if true then..else” block found in Logic. This block handles decisions. Such as if the water is touching the foil.

Step 4: Greater than?



From Logic, we need to grab the “0 < 0” comparison block and place that over the True of the if block. Then change the < to a >.

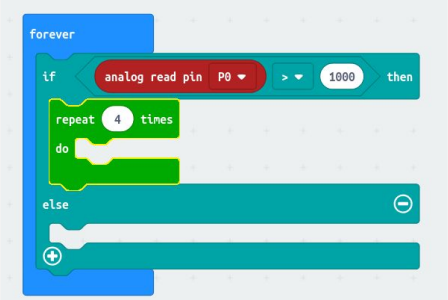
Step 5: Checking values



Create your own flood detector!

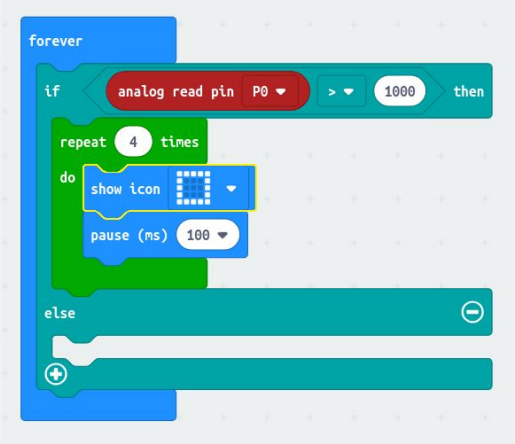
From the Pin section, found in Advanced. Drag “analog read pin P0” and place it in the first zero. In the second change the value to 1000 which is the trigger for our sensor.

Step 6: Repeat after me...



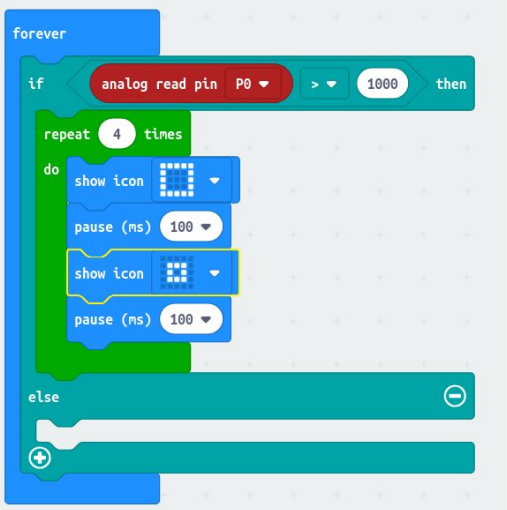
So if the sensor is triggered, we need to alert the user, and using the “repeat 4 times” block from Loops we can start an animation loop.

Step 7: Big Square



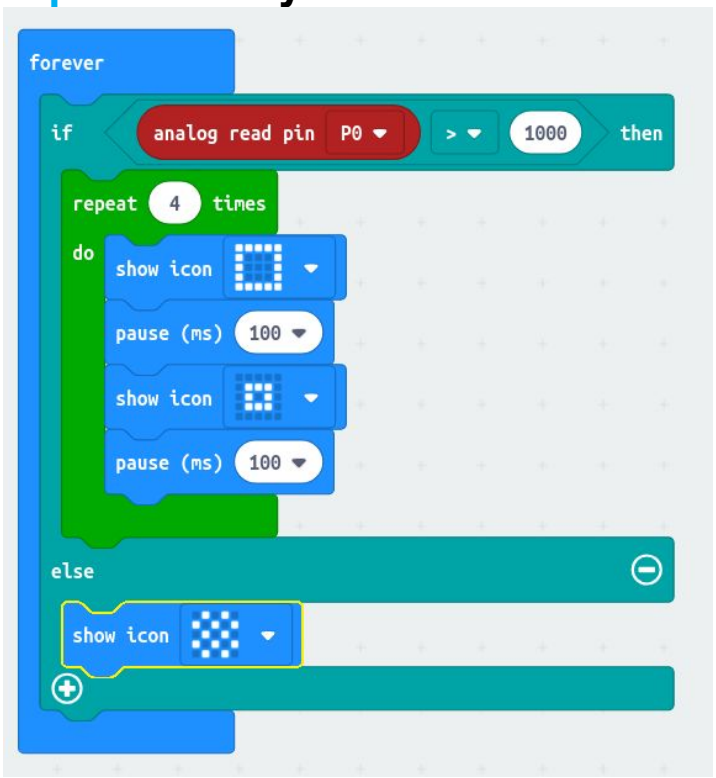
These blocks from Basic are “Show Icon”, which we use to display a large square. Then we delay for 100 ms using “pause.”

Step 8: Small Square



Again from Basic, we use “Show Icon” but this time we have a small square, and another “pause” when the loop runs it will look animated!

Step 9: Standby Mode



Inside the else part of our condition, we add another “Show Icon” with a chess board design. This is the default that shows when the water level is low.

That’s it, now download the code to your micro:bit and when it has finished, drop the foil sensors into a cup or bowl and slowly fill it up with water. When the water touches the sensors, the LED matrix on the micro:bit will change and alert you to high water.

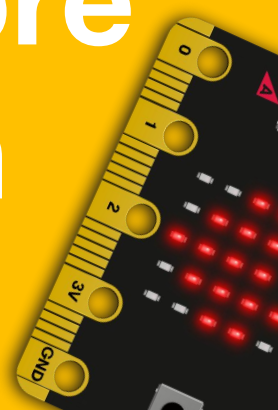
Well done you have made your own flood sensor! Can you change the code to make a buzzer sound for a blind person? Or perhaps send a radio message to a remote micro:bit?

Product Reviews

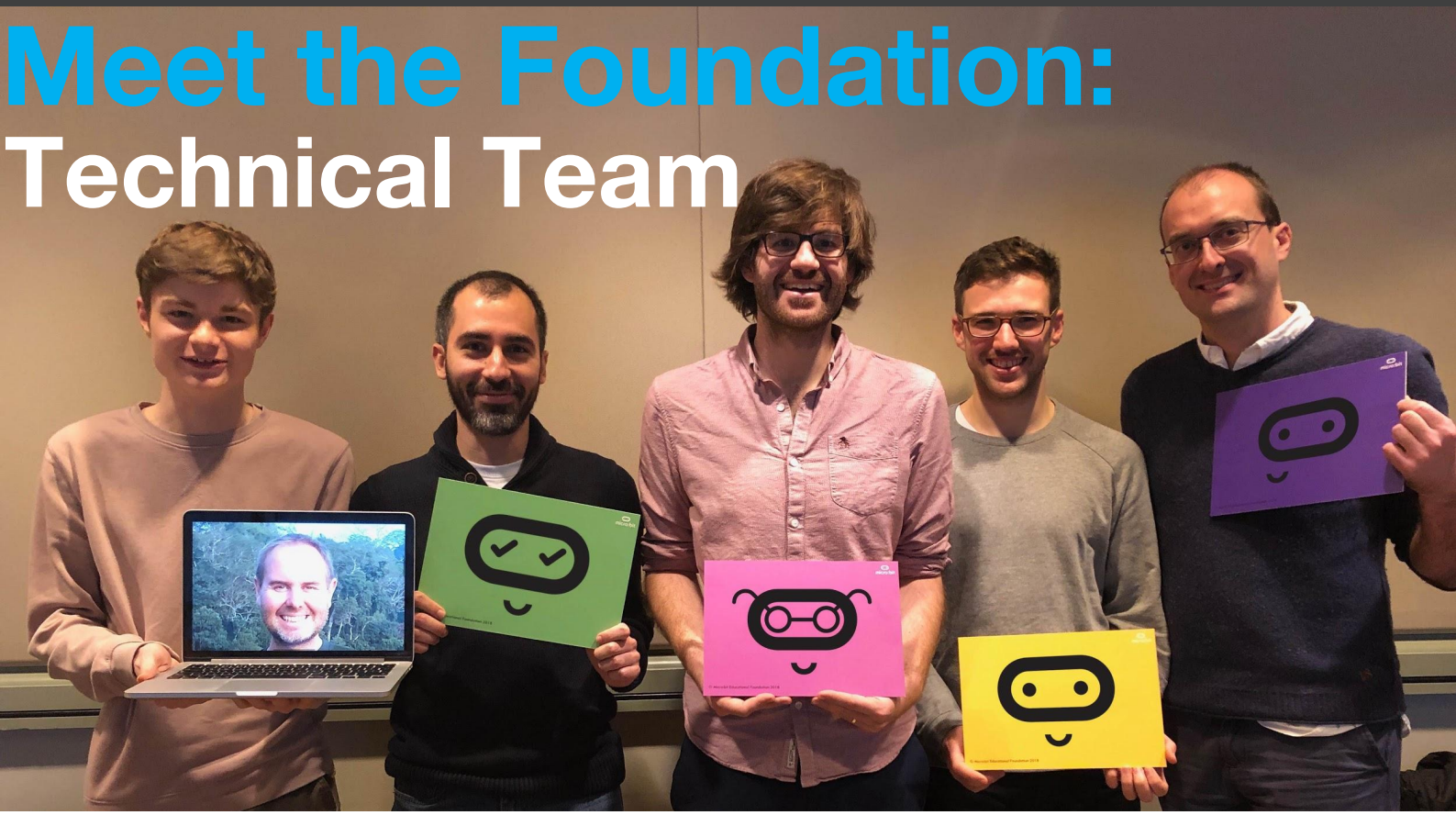
If you make cool products for the micro:bit, then micro:mag is the place to get it reviewed! With thousands of community readers per issue, it's also a great place to get your product/addon noticed.

Get in touch for more info - email us on

hello@micromag.cc



Meet the Foundation: Technical Team



Meet the people who keep the techy side of micro:bit going!

If you've ever wondered who looks after the zeros and ones at micro:bit headquarters, then wonder no more: come and meet the micro:bit technical team! As well as hardware projects, like making sure that the micro:bit motion sensor revision went smoothly, the team keeps the website up and running, troubleshoots user problems and helps find, file and fix bugs in the micro:bit editors and software layers. They're always busy behind the scenes, smoothing your path to digital creativity. Here's who they are and what they get up to...

Jonny Austin, CTO

Jonny first got involved with micro:bit at Arm, who was one of the 30 partners that supported the BBC to deliver the micro:bit. As the project rolled on, he got increasingly involved and is now seconded from Arm, to lead the Tech Team and help support the ecosystem of fantastic partners and community members around the micro:bit. He previously worked on Linux before moving to Arm's mbed team to focus on Bluetooth Low Energy. Jonny is also a co-founder of Makespace Cambridge, which has informed and inspired his work at the Foundation ... as well as providing him with a lot of fun!

Matt Hillsdon, Web Technology Lead

As his job title suggests, Matt leads work on the Foundation's websites. He's an experienced software developer with a love of programming that dates all the way back to the BBC Micro. Away from the keyboard, Matt enjoys cycling and playing pool – as well as having fun learning about the hardware side of micro:bit.

Mark Williams, Tech Support Engineer

As you might have guessed, Mark provides technical support for micro:bit users. He's done lots of work with educators and students at all levels and enjoys finding ways to explain complex technical issues to a non-technical audience. With a background in music education, Mark is often found tinkering with technology to find new and interesting ways to use it.

Carlos Pereira Atencio, Software Engineer

Carlos first got involved in STEM education initiatives as a STEM ambassador ... and when he met the micro:bit at a Python User Group three years ago he got completely hooked! Within the Foundation he concentrates on embedded software, but also enjoys collaborating on other software projects. Carlos recently moved to Oxford to join the tech team in their new office.

Ross Lowe, Junior Software Engineer

Ross has been involved with the micro:bit for many years in a variety of ways – from contributing to the official documentation to presenting it on BBC Radio 4's *Today Show* and BBC TV's *The One Show*. Ross is currently using his gap year to work on the microbit.org website and to help run the micro:bit Global Challenge competition.

Joe Finney, Device Software Lead

Joe developed and maintains the micro:bit Device Abstract Layer (DAL) – a bit like a tiny operating system for the micro:bit that's used by user-friendly languages, such as MakeCode and Micropython, to gain easy access to all the features on the micro:bit. By night, Joe is also a senior lecturer at the School of Computing and Communications at Lancaster University and plays bass guitar in a local rock band.

Sam Kent, Graduate Software Engineer

Sam also works on developing features and improvements for the micro:bit DAL, along with various other bits of software within the ecosystem. He has an engineering background, and in his free time, there's nothing he enjoys more than building and fixing things.

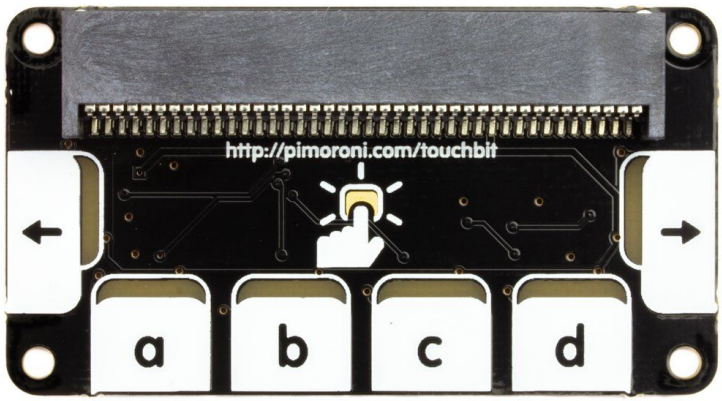
Feedback

We'd love to get your feedback on this issue of micro:mag. It helps us improve the magazine. If you have anything you'd like to share with us, please do get in touch with us, we really appreciate it

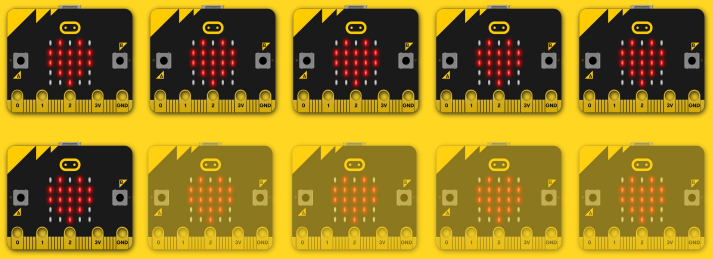
Get in touch!

micromag.cc/contact
hello@micromag.cc
@micro_mag

Pimoroni touch:bit



Six handy touch-sensitive buttons and LEDs for your micro:bit.



6/10

Another useful device for the micro:bit is Pimoroni's touch:bit. This has 6 capacitive touch pads - these work a bit like the touchscreen of your phone, and allow you to provide more buttons to use to provide inputs to the micro:bit. For a full explanation of how capacitive touch works, you can check out Tanya Fish's Pimoroni blog post all about it at **here**. The board also has LEDs mounted to allow you to give some basic feedback when buttons are pressed.

The install and set up of the touch:bit is really easy, with a plug and play connector to connect to the micro:bit and the software install a cinch. Again though, there is no support for micropython - although this is less of an issue here because the board is far more geared to beginners than advanced users, as the automation:bit was.

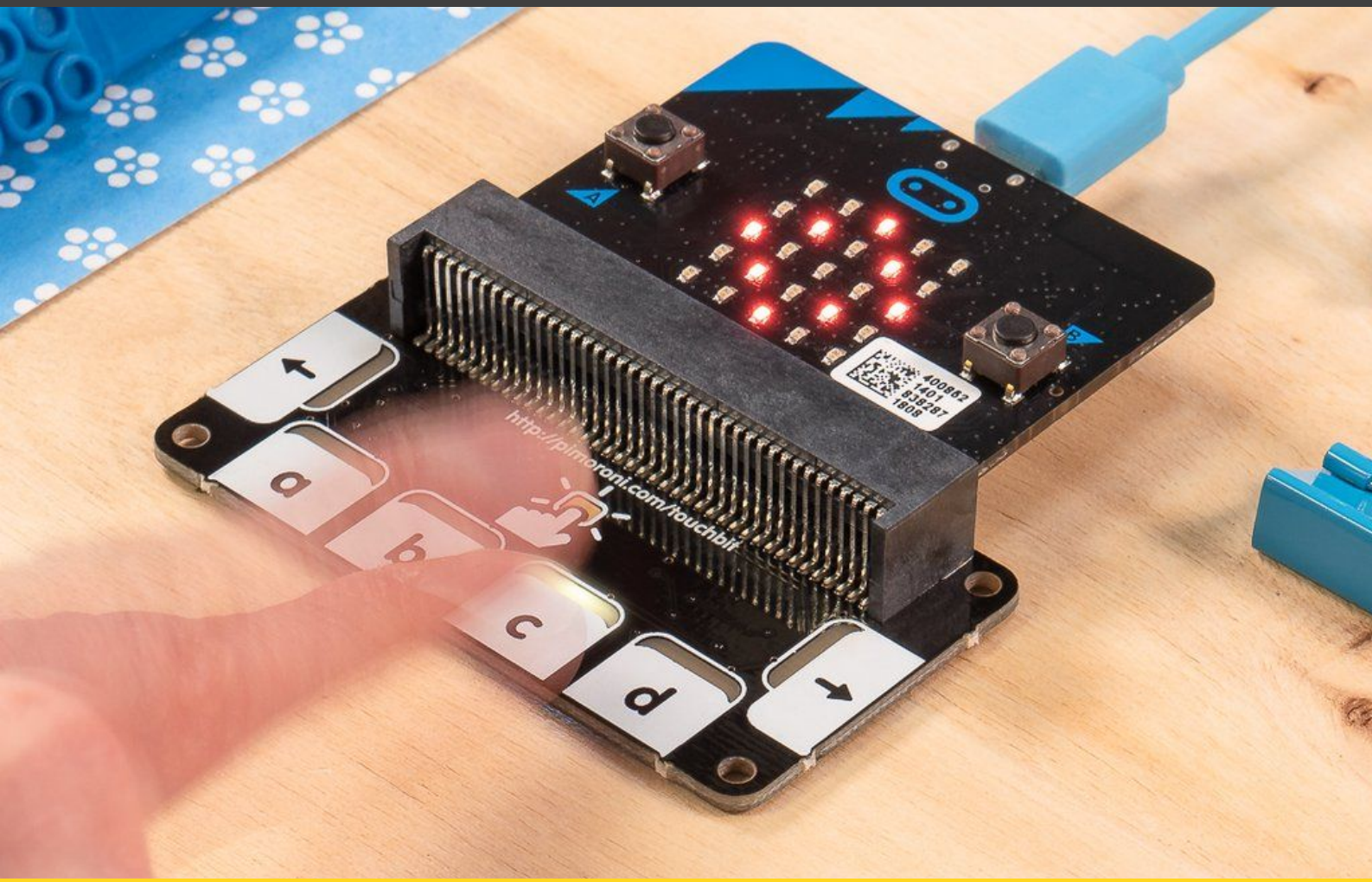
The provided block library allows you to independently control the LEDs beneath the touch pads (providing you set them to manual mode first), as well as sensing the input from the touch pads themselves, both as event triggers and logic blocks (which is nice).

Back of the touch:bit, we love the handy instructions and project ideas.

Maker Quote

"Use touch:bit as a controller for games on micro:bit's LED matrix, or combine it with the radio functionality and use it as a controller for your robot."

Pimoroni touch:bit :review



The touch:bit provides a nice little touch interface for your micro:bit projects. **Photo: Pimoroni**

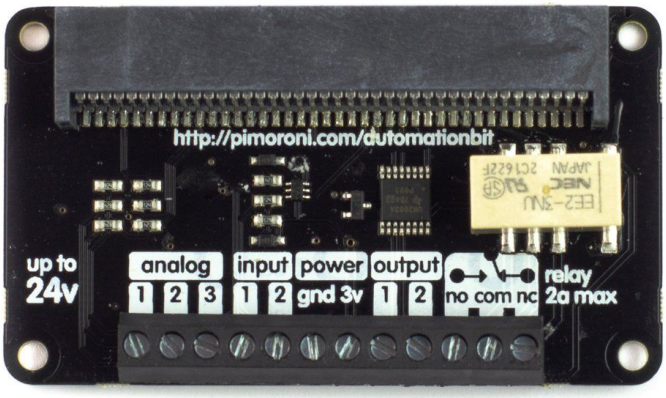
A major issue in our eyes though, is that there is no bare copper on the capacitive touch pads like on other competing boards (and even some of Pimoroni's own), and even the micro:bit itself. Whilst this probably helps the aesthetics, it means you can't connect the pads to other sensors with alligator clips - sadly this means no fruit pianos, foil sensor-stairs or human chains. This is quite disappointing - as these types of projects are amazing for engaging kids, and it would've been an easy inclusion as far as we can see. Combined with the fact that you lose the 3 capacitive touch pads built in to the edge connector, this really takes away from the potential uses for touch:bit in our eyes.

Overall thoughts:

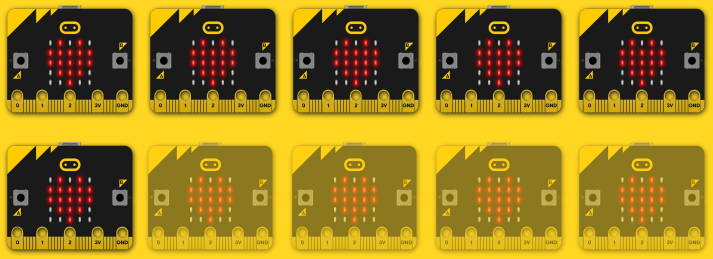
After playing with this addon board for a while now, we think this is a really nice "first add-on board" for beginners, but a couple of improvements we've highlighted in this review would make this perfect.

Buy the Pimoroni Touch:Bit: go.micromag.cc/touchbit
Also visit Pimoroni's learn site for some tutorials.

Pimoroni automation:bit



Control and monitor your world with automation:bit!



7/10

One of Pimoroni's newer add-ons for the micro:bit is the automation:bit. Slightly confusingly named (the automation comes from your code rather than the hardware), this neat little circuit board packs a relay, some 24V-tolerant inputs and 24V-sinking outputs and a 24V-tolerant 3-channel ADC (don't worry, we'll explain all that later!)

Put simply, this board allows you to control electronics that have a higher voltage than the micro:bit is normally capable of using. This means that you could use things like water pumps, USB devices, door locks and LED strips with a micro:bit - neat!

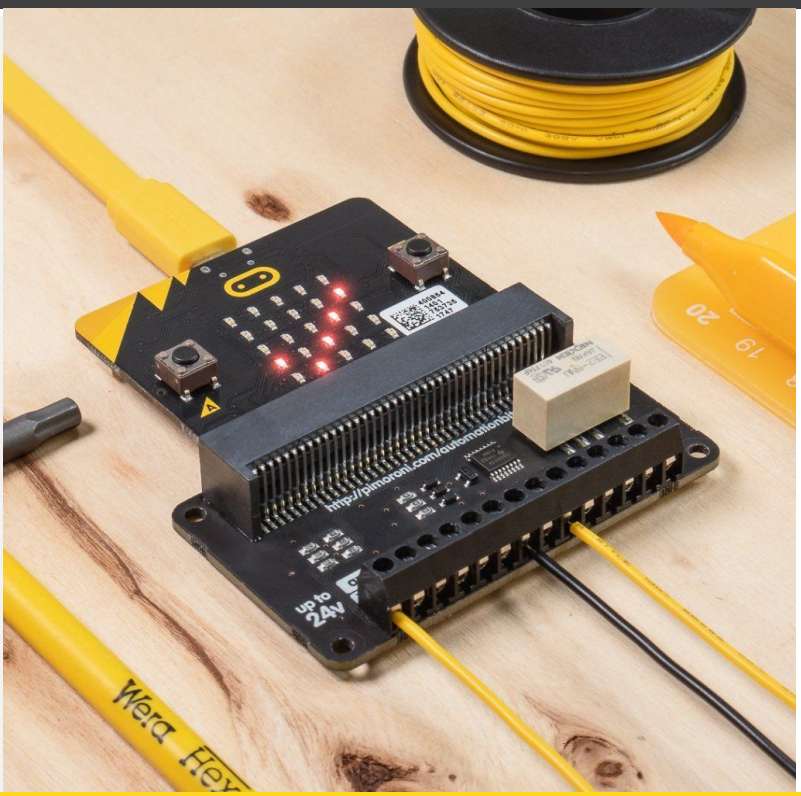
In's & Out's

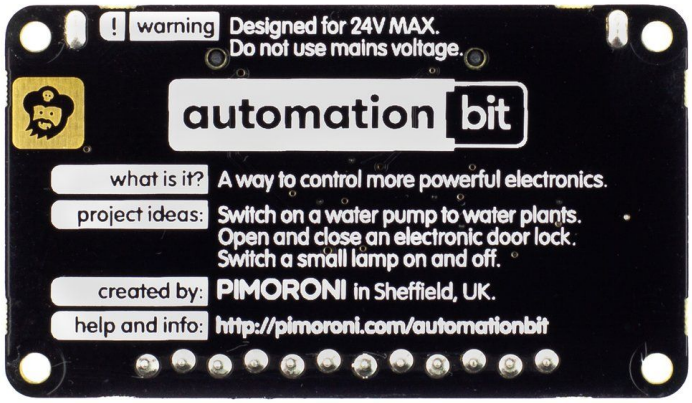
Relays switch external loads - they're like a sort of electronic switch. This means that the electronics from whatever you're turning on don't come into contact with the micro:bit. This is good for switching anything up to 24v - note this means not mains devices, and unless you have the relevant knowledge you shouldn't put them anywhere near your micro:bit! Low-voltage things are all good to go though.

The ADC converts analogue signals (from sensors - light sensors, dimmer switches, knobs or dials...) to digital number signals that you can use in your code. The extra special thing about the ADC on the automation:bit though, is that you can again use it with anything up to 24v. This means it's useful for experimenting with things like cars and boats (again, not unless you know what you're doing), which use 12v power systems.

There are also 24V tolerant inputs and outputs, but they're sinking rather than sourcing. This means that you need to connect the positive terminal of whatever needs turning on to your power source, but the negative terminal can be

Pimoroni automation:bit :review





The automation:bit connects straight to the edge connector on the micro:bit. Plug and Play! Photos: Pimoroni

attached to the output on the automation:bit.

The automation:bit is well designed, although the screw terminals can be a little hard to use for fiddly fingers (we'd have preferred the push-connector type which are easier to use). The design is nice and clean, with some helpful hints for where to find out more info as well as project examples and ideas.

The support for MakeCode is plentiful, with clear instructions available on the github page, although not in a particularly child-friendly format. Some more example project tutorials (a la adafruit) would be great, but Pimoroni's new kits will partially cover that need.

However, the lack of support for python is a little disappointing - this would be a great addition especially as it's the language of choice for most advanced projects.

We tried using the MakeCode library to switch a set of USB (5v) fairy lights on and off depending on the position of the compass - and it worked fantastically. What could you control based on the micro:bit's inputs?

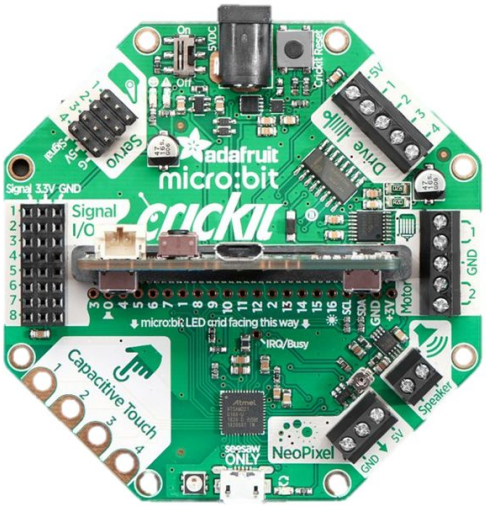
Overall thoughts:

A really versatile piece of kit, but needs micropython support to be truly useful for more advanced projects and some more project ideas would be welcome.

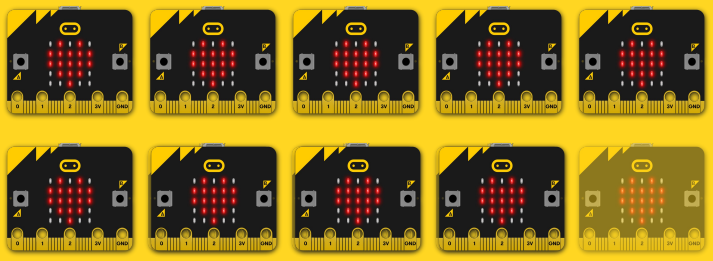
Buy the automation:bit:

go.micromag.cc/autobit

Adafruit Crickit for micro:bit



The ultimate platform for building DIY robots with micro:bit.



9/10

Maker giant Adafruit have entered the micro:bit game with their first add-on board, but is it any good? Adafruit have brought their Crickit robot platform to the micro:bit, joining their existing line up of add-ons for the Feather, Circuit Playground Express and the Raspberry Pi. This board aims to make building DIY robots as easy and simple as possible, basically the ultimate driver board.

Features

The board boasts all the things you could ask for when building a robot:

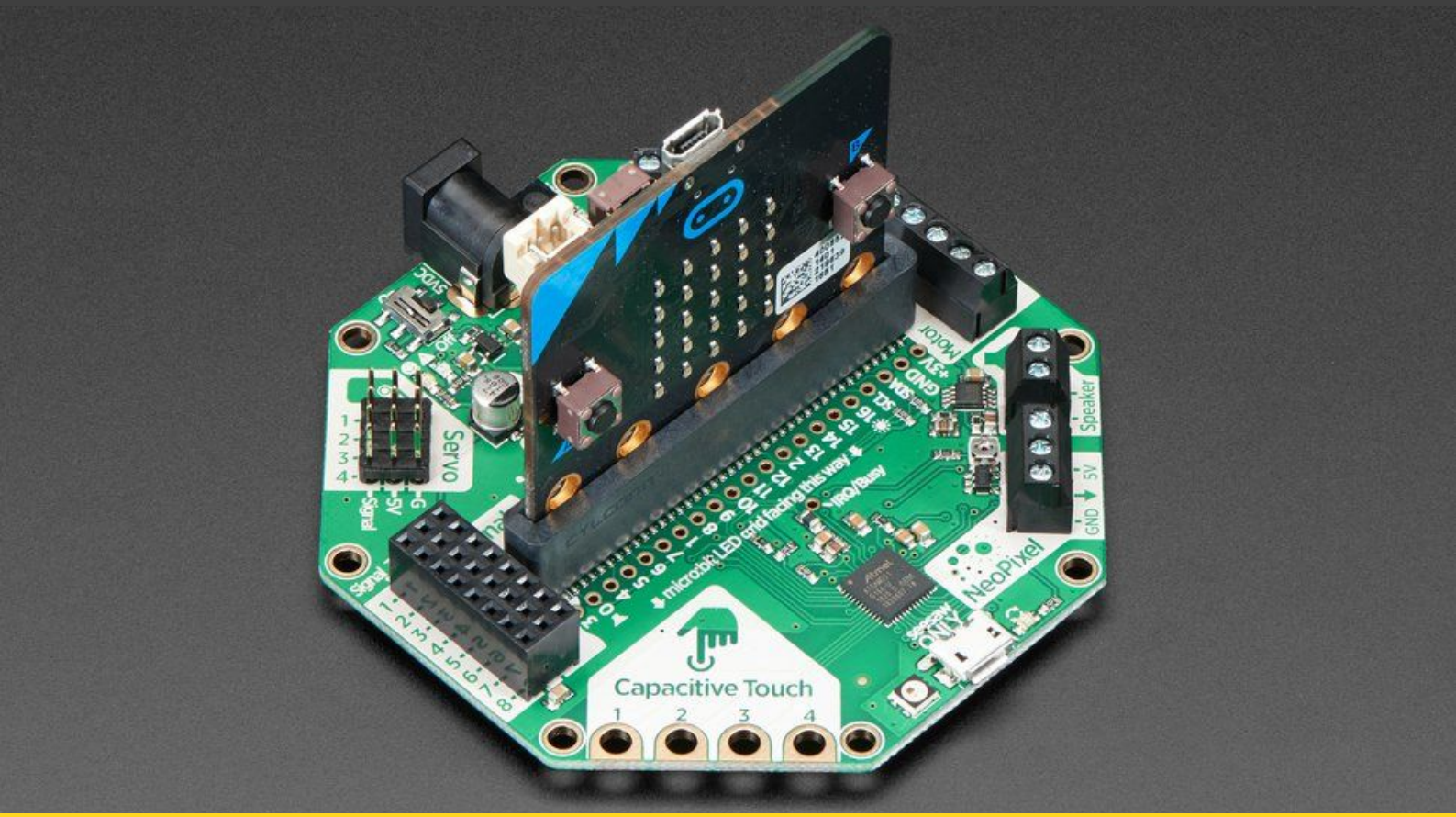
- Neopixel drivers
- Speaker output
- 2 DC motor outputs
- 4 Stepper motor outputs
- Capacitive touch
- Servo connectors
- Some I/O headers

We think that this is a pretty amazing range of functionality that gets added to the micro:bit when using Crickit, this definitely sets it apart from other add-ons on the market, it's ability to do so much. All of this is powered by a 5V 2A DC barrel jack, we find this as a weak point of this board as we'd prefer the onboard USB to be it's way of power delivery as it's a more common connector, sourcing a reputable DC power supply here in the UK can be quite expensive for some people. We have tested the Crickit with some servos, motors and neopixels and we're safe to say this board makes it so easy for beginners to start building robots with the micro:bit or even those who just want to drive things like servos and neopixels but want an easy way to wire them up to the micro:bit rather than having the hassle of croc-clips.

Programming

As with most add-on boards, you'd expect a MakeCode extension, this isn't any different for the Crickit. Adafruit have created a

Adafruit Crickit for micro:bit :review



The Crickit is easy to connect to your micro:bit, it's simply plug + play! **Photo: Adafruit**

comprehensive MakeCode Extension that allows you to control all the onboard features with easy to use blocks. We like the fact that whilst it's easy to use, it doesn't compromise on expandability for more advanced users. Adafruit also claim that the Crickit works with Arduino on the micro:bit, however we haven't tested this. One thing that did disappoint us about the Crickit is the lack of MicroPython support. This is a common occurrence with many add-on boards for the micro:bit due to the size limitations of the micro:bit's memory but we hope in the future than MicroPython is something that Adafruit will add support for and according to their website this is something that they plan on adding, we think that this would be a huge improvement for those who want to use a more advanced language like Python to use the Crickit

Resources

If you're familiar with Adafruit, you'll have no doubt heard about their amazing learning site. They've created a few guides which should be enough to get you started and there are always lots being added, you'll find it hard to run out of things to do with this board!

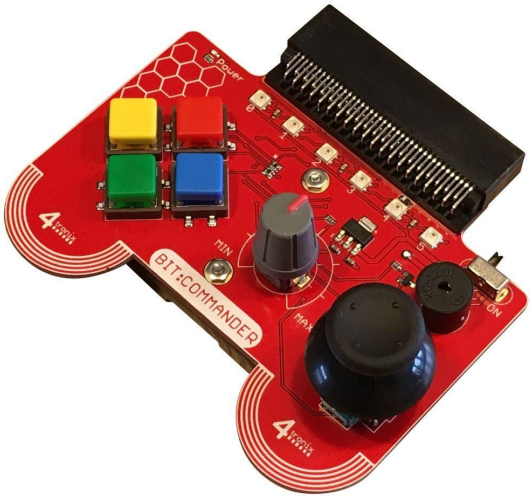
Overall thoughts:

The crickit is an amazing add-on board for those who want to tinker with robotics on the micro:bit, be that a beginner or expert. However, the lack of Python support prevents us giving this full marks.

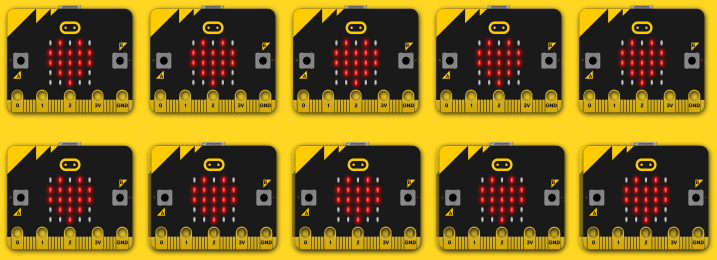
Buy the Crickit:

go.micromag.cc/crickit

4Tronix Bit:Commander



Control your next micro:bit project with Bit:Commander!



10/10

Lots of projects out there use 2 micro:bit's in their projects, one of them is typically a controller for the other micro:bit which will be the master for the main project. Most people use all sorts of ways to control the other micro:bit, the A+B buttons, different gestures and sometimes this can be hard to remember or isn't really practical.
This is where Bit:Commander comes in. This neat little add-on aims to be the perfect board for adding to your controller. It boasts a variety of different inputs as well as some programmable lights, let's take a look at some of the features.

Features

- 6 multi-colour RGB LEDs (aka neopixels)
- 4 push buttons with coloured caps (Red. Yellow, Green, Blue)
- Analog dial input
- Analog Joystick
- Powered miniature speaker

This add-on has a really impressive range of inputs to make the perfect controller for your micro:bit projects. The Bit:Commander priced at £16.90, which we think is just right for the amount of features it has. One of the things we like the most about this board is the onboard battery pack, this is connected to the PCB itself and allows you to add batteries to make the micro:bit completely portable. This saves you from having a battery pack plugged into the JST connector with no way to hold it in place or having your micro:bit powered via USB all the time, however, do note that when you are programming and testing over USB, for the features to work, you need batteries plugged in and the On/Off power switch on. Like with most add-on boards, the micro:bit slots into the Bit:Commander via an Edge Connector, this makes it easy to use because it's simply plug and play. The overall design feels really nice and works well as a handheld controller and one thing that we'd like to see in a future revision of

Bit:Commander :review

The Bit:Commander plugs into the micro:bit by it's edge connector. **Photo: Josh**

this board is something like a Vibration motor to add to the controller experience, we really liked this about another controller type board we reviewed last Issue, the :GAME Bit 64.

Programming

The Bit:Commander is really easy to program and it supports MakeCode & MicroPython. 4Tronix have built a custom MakeCode library to make it really easy to get started with coding the Bit:Commander. Whilst there is no bespoke MicroPython library, it's really easy to code the Bit:Commander with MicroPythons existing commands.

Resources

The Bit:Commander has a variety of resources that are linked to on the 4Tronix Blog, these are written by the community and there are resources for both MakeCode & MicroPython. They're great for beginners and make it really easy to get started.

Overall thoughts:

We think this is the perfect add-on board for those who want to create a controller for their micro:bit project, or even those who just want to play around with joysticks, buttons and NeoPixels!

Buy the 4Tronix Bit:Commander:
go.micromag.cc/bitcommand
Resources:
go.micromag.cc/commandres

micro:mag

micromag.cc
@micro_mag