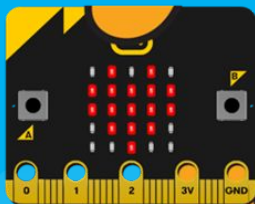


Scratch for micro:bit

Learn all about the new micro:bit features in
Scratch 3.0!

Scratch Interview

We interview 2 members of the
Scratch team!



cube:bit Review

Turn your micro:bit into a shiny
3D cube with a new board

Global Challenge

Solve a problem in your
community with the new
micro:bit challenge!

Make your own Blocks!

Make your own MakeCode blocks
with our easy to use beginners
guide.





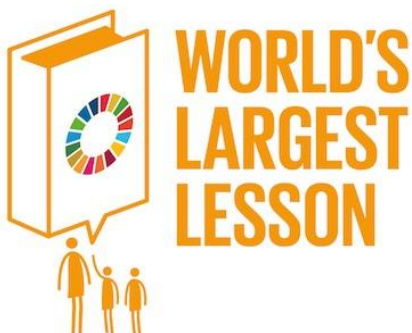
Could you help achieve the Global Goals?

We're setting you the biggest
coding challenge yet!

You could win an all expenses paid trip to London to
take part in an amazing micro:bit Global Challenge day
with winners from around the world!

The competition is open to young people across the
world.

Visit our website now for more details on how to enter:
microbit.org/global-challenge



Welcome to Issue 2

Hi

there. I'm delighted to be able to welcome you to Issue 2 of micro:mag.

basics of scratch 3, a Scratch team interview and much more.


Following the major success of Issue 1, we're happy to bring you an improved Issue 2 covering lots of news, projects, tutorials and reviews.

We're also excited to bring you an improved design for Issue 2 in which we've incorporated feedback from our readers, we hope you like it as much as we do.

This issues focus is Scratch 3.0, which is the new version of Scratch due to be released in January. The exciting thing about this new version of Scratch is it's compatibility of the micro:bit, the new extension in Scratch 3 allows you to connect up your micro:bit with your Scratch projects to bring them into the physical world. We've got articles dedicated to Scratch 3 in a brand new cover feature, you can expect articles like making a games controller, the

Issue 2 spreads across an impressive 96 pages of content from our community contributors, we really couldn't do it without them. We're now looking for contributors for Issue 3 which is due to be released in January. If you'd like to write for us, please do fill in the form over at micromag.cc/contribute.

That's it from me so grab a seat, a brew and tuck into Issue 2, we hope you enjoy it!

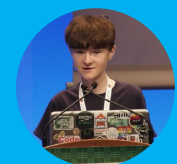


Joshua Lowe
Editorial Team

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!

Team

Editor in Chief
Kerry Kidd

Editorial team
Joshua Lowe
Archie Roques
Thomas Bass

Contributors

Giles Booth
Shenali Welkala
Michael Rimicans
Eric Roseenbaum
Kreg Hanning
Eileen King
Thomas Stratford
Alan Yorrinks
Soibheann Morgan
Martin Woolley
Nicole Parrot
Jody Carter
Sam Watson
Davit Markarian
Les Pounder
Hal Speed
Warris Candra
Rachel Lancaster
Jose Scodiero
Daniel Pers
Chris Penn
Areej Abdullah Alghamdi
John Lynch

Contact us:

Website: micromag.cc

Email: hello@micromag.cc

Twitter: [@micro_mag](https://twitter.com/micro_mag)

Cover Feature:

Scratch meets micro:bit



Page 19



Create your own MakeCode Blocks

Page 54

micro:hit - Countdown Timer

Page 82

Global Challenge



Page 9

news

- 6** **Mu V1.0 is here**
Beginners Python IDE is here
- 7** **WebUSB & Crikrit**
WebUSB beta & new Adafruit Crikrit
- 8** **Girls into Coding**
Young Avye's latest crowdfunder
- 9** **Global Challenge**
Largest micro:bit competition ever
- 12** **HackBit 2018**
A national level micro:bit hackathon
- 16** **Micro:bit in Saudi Arabia**
Transforming education with micro:bit

Cover Feature

- 20** **What's new in Scratch 3.0?**
Rundown of the new Scratch
- 21** **The micro:bit extension**
Learn about the micro:bit blocks
- 22** **Basics of Scratch + micro:bit**
Learn the basics of the new blocks
- 26** **Scratch Team interview**
Scratch answer our questions

29 **Scratch Games Controller**
Make a micro:bit Scratch controller

33 **Learn 2 Teach, Teach 2 Learn**
micro:bit SeeSaw project

:feature

37 **Micro Simon**
Hack a classic game with micro:bit

40 **Python Debugging**
Handy MicroPython debugging tips

:make

47 **micro:bit & Kodu**
Control Kodu with micro:bit

55 **MakeCode Custom Blocks**
Make your own MakeCode Blocks

63 **Formula One in Schools**
Make a micro:bit race car

70 **The micro:bit Express**
Hack a Lego Train with micro:bit

51 **Carbon Dioxide Monitoring**
Detect Carbon Dioxide with micro:bit

60 **micro:melt**
Make your own Weather Station

67 **Micro:Mate Smart Display**
Link micro:bit to Google Assistant

75 **Body Position Sensor**
Track movement with micro:bit

More Articles

82 **Countdown Timer**
Micro:hit returns for Issue 2!

84 **Meet the Foundation**
Global Engagement Team

:review

88 **4Tronix cube:bit**
3D cubes of awesomeness

90 **DFRobot Driver Board**
Low cost motor driver board

92 **Dexter GiggieBot**
An easy to control micro:bit robot

94 **Kitronik :GAME ZIP 64**
A retro handheld gaming addon

Mu Editor V1.0.0 is HERE

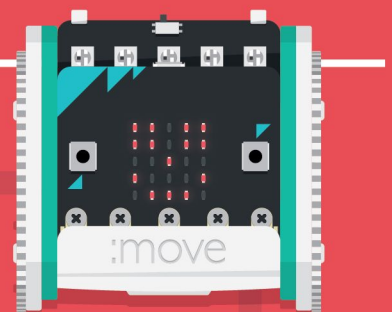
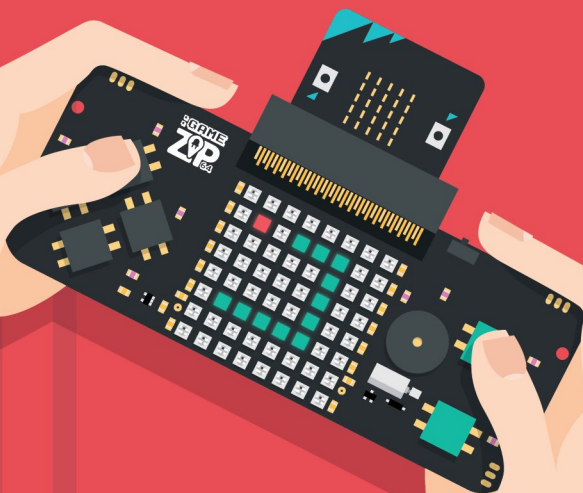
The next generation of Mu, the beginners Python IDE is here with brand new features.

What started off as a basic MicroPython editor for the micro:bit has, with the help of developer Nicholas Tollervey and community members, turned into a full Python IDE for beginners. Mu's simple but clean interface provides a basic environment to

start coding with Python compared to IDE's like IDLE. Mu now supports platforms like Adafruit's CircuitPython alongside PyGame Zero. So why not head over to codewith.mu to download Version 1.0.0 for your device today!




Kitronik



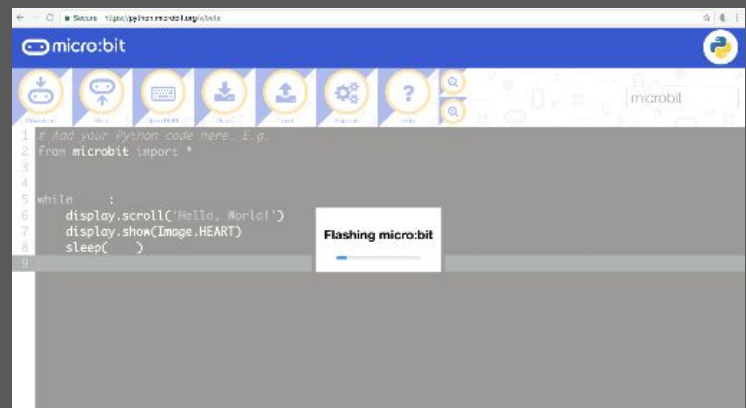
DISCOVER OUR
FULL RANGE
OF MICRO:BIT
ACCESSORIES

webUSB Testing is OPEN

A new and easier way to flash your micro:bit is in it's BETA stage and needs you!

Are you a regular user of the MakeCode and Python editors? If you run chrome and use one of these editors, you can now try out webUSB which allows you to directly flash your code onto your micro:bit without having to drag and drop the file onto the

device. This will make the process easier for beginners. The Micro:bit Educational Foundation invite you to test out the Beta of WebUSB to find any bugs that need fixing. At micro:mag we love the idea of webUSB and have been testing it for a while, so follow the link to sign up and start using webUSB.



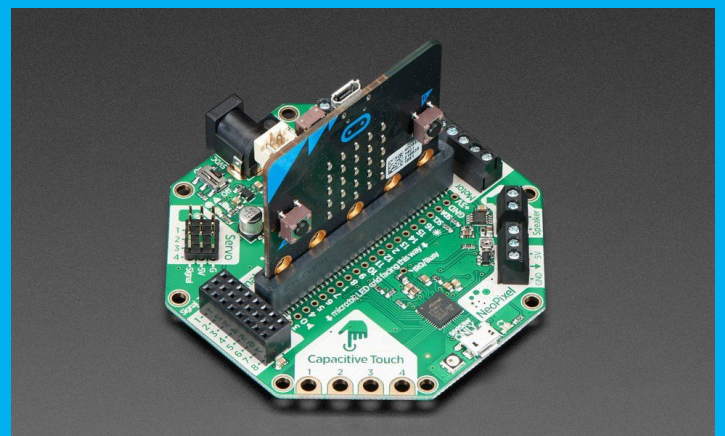
go.micromag.cc/webusb

Adafruit Crikkit for micro:bit

Adafruit's new Crikkit platform now available for micro:bit.

If your an avid fan of Adafruit, you may have seen their Crikkit series of boards. These boards which are compatible with the Circuit Playground Express and now the micro:bit allow you to easily create robot's, control servo's, make flashy lights with neopixels and much more.

Crikkit stands for Creative Robotica & Interactive Construction kit and priced at just £27.60 from Pimoroni in the UK you get lots of cool features to make your own robot that you wouldn't normally be able to with just a micro:bit. Click on the link on the right to learn more and purchase a Crikkit.



go.micromag.cc/crikkit



Avey presenting her robot at Coolest Projects this year. Photo: GoFundMe

Girls into Coding Crowdfunder

Passionate young coder Avey is planning her second event and needs your help!

11 year old coder and maker Avey after her first successful Girls into Coding event back in July is crowdfunding to run her second. The events aim is to get more girls into coding as there is a current shortage in girls that are interested, so Avey hopes to inspire more girls to be like her and she's doing this via these events.

Why crowdfunding?

Avey is crowdfunding so that at the end of the event each girl gets to take home their own tech package that will help them start their journey. Inside the package will be a BBC micro:bit starter pack and a Tinker kit, which has a set of modules to create exciting projects.

What will happen at the event?

15 girls will be able to participate in the event where they will participate in a number of coding workshops run by Avey, Firetech Camp and Think Create Learn. Alongside these workshops will be some talks by some inspiring community members.

How can I help?

Join Avey's mission to end the lack of female representation in STEM by donating to her crowdfunder below to help run this exciting event.

go.micromag.cc/girls-coding



Giles is the Educational Content Manager at the Micro:bit Educational Foundation. Previously he retrained as a teacher after a long career in BBC radio and has taught Computing from Reception to Year 11.

@blogmywiki

The micro:bit Global Challenge



Photographer : Ms Soratda Worrakittichotikorn
Ban Houy-Youkpaso School, Chiang Rai Province, Office of the Basic Education Commission (OBEC), Thailand. Courtesy of: World's Largest Lesson

Small changes can have a big impact – you can help the world promote sustainable development!

Could you help make the world a better place? Do you have ideas about how technology can help achieve the United Nations Global Goals? Are you aged between 8 and 12? Take part in the most ambitious coding challenge yet and you could win an expenses-paid trip to London to take part in an amazing micro:bit Global Challenge day with winners from around the world!

The micro:bit Global Challenge is an extraordinarily ambitious project which will see the BBC micro:bit empower children to raise

their awareness of their community and develop their computing skills. It's a collaboration between World's Largest Lesson, Arm and the Micro:bit Educational Foundation to create a worldwide competition for children aged between 8 and 12.

The competition is focused on the United Nations Global Goals for Sustainable Development (<https://www.globalgoals.org>), a series of ambitious targets to end extreme poverty, fight inequality and injustice and fix

climate change for everyone by 2030. Taking part in the micro:bit Global Challenge will enable children to explore issues that relate to them and their communities and develop their problem-solving skills. Perhaps you have ideas about using the BBC micro:bit to make your school or neighbourhood a safer place to be, or ideas about making the world a healthier place?



Your project could help make the world a healthier place

There will be six finalists from different regions: Europe, North America, the Middle East, Africa, Asia and Pacific (including Australia and New Zealand), and Latin America and the Caribbean. You can work in a team or as an individual, and one young person (plus parent or guardian) from each winning project will be brought to London will take part in the Global Challenge finale event at the end of January 2019 where the regional winners will take part in a series of group challenges and masterclasses.

The competition is free to enter. You can create your projects using MakeCode or Python, but you don't have to have a micro:bit to take part! Entries will be accepted on paper as well as

electronically. The micro:bit Global Challenge Producer Jo Inglis explained to me that the project is designed to provoke conversation and inspire creativity in the classroom and it will be accessible for non-experts, children and adults alike. The Micro:bit Educational Foundation is providing cross-curricular resources to support pupils and teachers, whether they have used a BBC micro:bit before or not.

Emma Smart of the Micro:bit Educational Foundation is very excited about the Global Challenge. She told me "This is an amazing project to be a part of, the micro:bit has huge potential for helping to solve real-world problems and we're looking forward to seeing the creative solutions from the micro:bit community. We couldn't do this on our own and want to say a massive thank you to everyone who is working with us on this incredible competition."



**Photographer : Ms Soratda Worrakittichotikorn
Ban Houy-Youkpasso School, Chiang Rai Province, Office of the
Basic Education Commission (OBEC), Thailand Courtesy of: World's
Largest Lesson**

If you want some inspiration for how the BBC micro:bit can be used in creative ways, have a look at the Ideas page <http://microbit.org/ideas/> and

find out more about how to code the device

<http://microbit.org/code/>



Jo Inglis, Producer for the micro:bit Global Challenge

Kavita Kapoor, the Micro:bit Educational Foundation's Chief Operating Officer, says this is the most ambitious micro:bit project yet. "The micro:bit already inspires young people all over the world and we see such incredible projects from such diverse places which include classrooms in Canada, refugee camps in Greece and the Amazon jungle! The Global Challenge will give an opportunity to children all over the world not just to learn more about practical uses of coding, but to connect with like-minded people in problem-solving that will make a genuine positive difference to our collective future. I am really excited about meeting participants, finding out about their communities and the different challenges they are solving with the tiny but powerful micro:bit. Best of luck"



Courtesy of: World's Largest Lesson

The closing date for submissions is in mid-November 2018. You can find more details about how to enter the competition on our website (find the link at the bottom of the page) . Shortlisted submissions will be judged by regional teams who will pick six winners who will be brought together in London, UK for the competition finale.



Small devices can have a big impact!

Find out more about the challenge over at:

go.micromag.cc/global-challenge

FINDING FITS at the Hack:Bit 2018

Shenali Welikala



**Chief Editor at
STEMUp
Educational
Foundation**

[@microbitslug](#)

[stemup.foundation](#)

The First Ever National Level micro:bit based Hackathon, Hack:bit 2018 concluded successfully.



A school team is presenting their solution in front of the judge panel

Hack:Bit 2018 made history as the first-ever National level micro:bit based hackathon. The hackathon was organized by micro:bit Sri Lanka User Group (micro:bit SLUG), an affiliate of the micro:bit Education Foundation based in the UK and a child organization of the STEMUp Educational Foundation. The Hackathon kick-started with the opening of idea submissions on January 2018. Ideation was called from both universities as well as school categories.

Around 80 submissions were received at the time submissions closed in the month of April. The ideas were then presented to a panel of judges consisting of 3 international and 3 local judges namely

1. Mr Philip Meitiner- Former Head of International Programs, micro:bit Educational Foundation, UK
2. Mr Howard Baker- Researcher, micro:bit Educational Foundation, UK
3. Mr Waris Candra- Head of Asia, micro:bit Educational Foundation, UK
4. Mr Shameera Prajapriya- Solution Architect, WebQuarters
5. Mr Pradeep Kotuwegedara- Senior Learning Solutions Specialist, Tech One Global
6. Mr Pradeep Senavirathne- Author (Embedded Technologies), Apress.

The top 25 ideas were selected to the final round of the competition based on the judges' marks.



Team SkyNet is finalizing their implementation.

Then it was time for the final round of the hackathon to be held. micro:bit SLUG partnered up with Sarvodaya Fusion and Microsoft to put forward the final round creating a sturdy platform for the students to perform well and bring their ideation into reality. The finals were held at Trace Expert City Auditorium as a 24 hour Hackathon starting on Friday 6th of July 4.00 p.m. till Saturday 7th of July at 4.00 p.m. The university category had to compete for 24 hours whereas the schoolers had to compete subjected to a time limit of 8 hours. A total of 14 school teams competed head to head at the finals. The teams include Ananda College Colombo (1 team),

Gateway College Colombo (1 team), Bandaranayake College, Gampaha (1 team), St. Anthony's College Kandy (1 team), St. Sebastian's College Moratuwa (1 team), Embilipitiya President's College (2 teams) and from Nanasala Centers (7 teams).

The universities who battled at the finals under the university category were University of Rajarata (2 teams), University of Moratuwa (3 teams), University of Colombo School of Computing (UCSC) (2 teams), Horizon Campus Malabe (1 team), Shilpa Sayura Digital Academy (1 team) and Kotelawala Defense University (1 team).



Team royal hackers is presenting their Smart Saline solution

The finalists under each category were given the aforesaid time durations to implement their ideas by using micro:bits which were provided to each team.

After an intense 24hrs of hacking the final products were evaluated by a panel of tech giants in the industry to select the winning idea.

The panel of judges of the final round who rendered their valuable contribution to select the winning products are as follows.

1. Mr Wellington Perera- CSA, Microsoft
2. Mr Thulasee Shan- TSP, Microsoft
3. Mr Isura Silva- Consultant, Sarvodaya Fusion
4. Mr Prasad Piyasena- Senior Consultant, SLIDA
5. Mr Chamira Jayasinghe- CEO, Arimac Lanka
6. Mr Calvin Hindle- Senior Business Analyst, MIT
7. Mrs Lin Gong Deutschmann- Managing Director, AOD
8. Mr Chalinda Abeykoon- CEO, Crowdisland.lk

9. Mr Shafraz Rahim- Senior Business Lead, Dialog Axiata

The two categories were evaluated separately by the aforesaid panel of judges and the final results were announced at the awards ceremony of the Hackathon.



Awards

The keynote speech was delivered by Hasitha Abeywardana- Country Manager, Microsoft Sri Lanka and the Maldives.



Hasitha Abeywardana - Country Manager, Microsoft Sri Lanka & Maldives

The winners of the school category were as follows

- Winner: Team Antonian Computer Fraternity- St. Anthony's College Kandy
- 1st Runner-Up: Team Dynamic Dudes- St. Anthony's College Kandy

Finding fits at the Hack:Bit 2018 :news

- 2nd Runner-Up: Team Royal Hackers- Nenasala Center, Udubaddawa



School category winners with their awards

The winners of the university category were as follows:

- Winner: Team Undefined- University of Moratuwa
- 1st Runner-Up: Team Imperium- University of Moratuwa
- 2nd Runner-Up: Team SkyNet- Kotelawala Defense University



University category winners with their awards

Winners and runners-up of the school category were awarded gifts worth LKR 50,000, LKR 35,000 and LKR 20,000 where the winners and runners-up of the university category were awarded cash prizes worth LKR 100,000, LKR 75,000 and LKR 50,000.

The gathering was addressed by Prabhath Mannapperuma- Executive Director, STEMUp Educational Foundation where he stated that their expectation was not only to see commercially viable products especially in the school category but to also give them the opportunity to make this competition a turning point of their lives.



Prabhath Mannapperuma - Executive Director, STEMUp Educational Foundation

With that note the first ever micro:bit based hackathon in the Nation came to a close by achieving its aim to help innovative and tech enthusiast students “Find Their Fits” through Hack:bit 2018.

Find out more about Hack:Bit:

StemUp Foundation Website:
go.micromag.cc/stemup

Micro:bit SLUG:
go.micromag.cc/mbSLUG

Hack:Bit Video:
go.micromag.cc/hbvideo

MICRO:BITS in Saudi Arabia Classrooms

Areej Abdullah Alghamdi



Areej ALGHAMDI has 14 years of experience in education. She is currently a computer science teacher in Saudi Arabia. Last year she won the best teacher award in Saudi Arabia.

A review of the change that has occurred with Computer science education in Saudi Arabia.



One of the student projects that Areej Alghamdi has helped her students create.

I would like to tell you about my experience and the qualitative leap in Saudi education that has occurred over the past two years and my aims to create a Saudi human actor in combination.

One of these changes is the change in the methods of teaching computing and linking it to the reality of the students.

Saudi teachers are very excited to have an effective impact on their students. One of these successful experiences is the adoption of the programming project through the micro:bit

Many teachers have implemented different programs with their students and many of them, despite the lack of Arabic content, help to discover many ideas and the results were impressive. Among the interactive methods that have been applied there has been the establishment of exhibits for student projects with micro:bit for the first time in the history of education in Saudi Arabia.

Many of the students were willing to learn and from the experiences that other students displayed and enjoyed.

One teacher created an interactive story to show the story to their children using microcube pieces as well as setting up an alarm system in case of an emergency or a fire. Many successful experiences have made many teachers want to learn more about the micro:bit and how it could be integrated into their lessons. This particular teacher experimented with the experience of applying the micro:bit with his students.



Some of the students projects. Just two of the many projects being carried out in class.

Using the micro:bit in class has changed a lot of my teaching strategies and also contributed to the attention of my students, especially when they are solving problems and with using the micro:bit, they are learning important skills required in the world today. I have also linked many of my students' projects with the objectives of sustainable development, for example, how to achieve environmental awareness and solve many environmental problems.

Microsoft Saudi Arabia have been great support in distributing some of the micro:bits to students in partnership with the Musk charity association. This is one of the reasons that called for the dissemination of physical programming. It is the involvement of the

community, the enthusiasm of the teachers and the qualitative shift in education that brought about this change.

Another teacher who also incorporated micro:bit into their classroom said: "Coding skills are highly relevant in today's scientific and technological careers, and they will only become more important in the future. That's why it is essential that we teach these skills" I hope that other countries can do the same that we have so their students can have experiences they need for the future.





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!

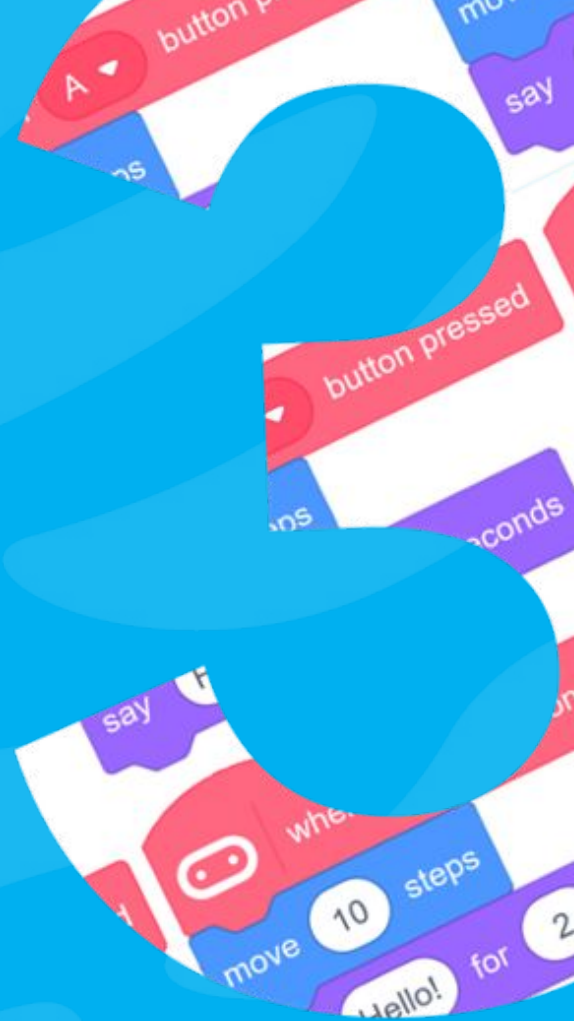
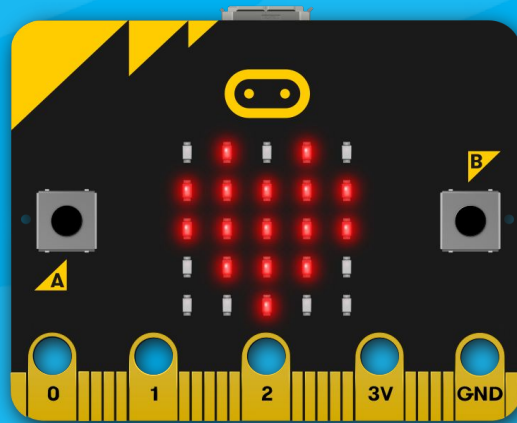
micromag.cc/contribute

hello@micromag.cc

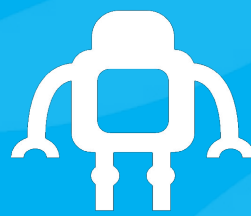
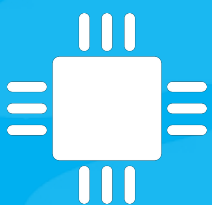
[@micro mag](#)



Scratch for



We love Scratch. It's one of the most popular ways for kids to learn coding. With 5 years olds using the basic Scratch Jr app on their tablets to 8-year-olds using the full version of Scratch. From story animations to games, those little blocks of code truly come to life with the power of Scratch. However, until very recently, there has been limited devices that you can use in conjunction with Scratch, these, for example, were things like Lego WeDo, which why great, can be very expensive for schools to buy. So, with the very latest version of Scratch 3.0, which has just been released, the Scratch team have brought micro:bit to this popular coding tool. In this exciting cover feature, we hope to cover the ins and outs of Scratch 3 and the micro:bit. Enjoy!



What's new in Scratch 3?

More extensions

The most exciting thing for us is the extensions that come with Scratch 3. This allows us to control the micro:bit and make it interact with Scratch. Making this coding tool even more exciting

Works on tablets

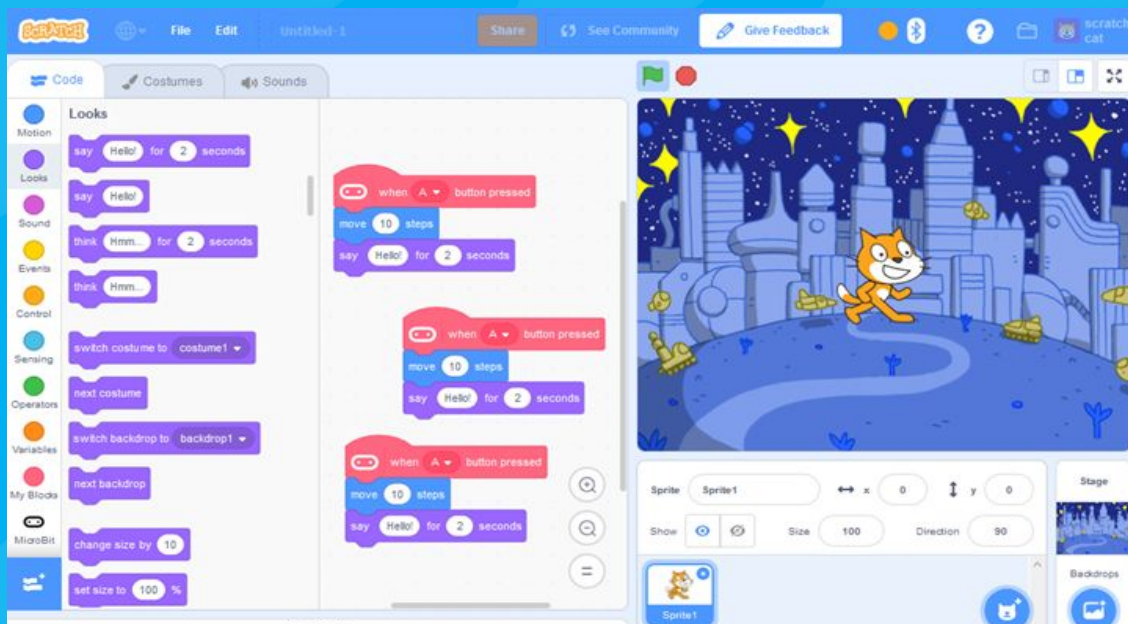
Scratch has been updated to use HTML instead of Flash, which means you can use it to program cool stuff on your tablet just like you would do on your PC!

A new look for Scratch

If you have used Scratch in the past, you'll notice the new look for scratch. Even the look of the Scratch blocks has changed.

Updated tools

The sound and costume editing tools have been updated to allow more control over sound files and the way you edit costumes. This will allow you to make your creations! sound and look better!



The sparkling new Scratch 3 UI, a familiar layout with a more modern twist



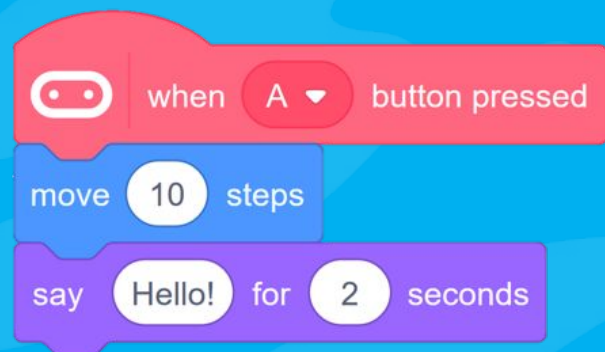
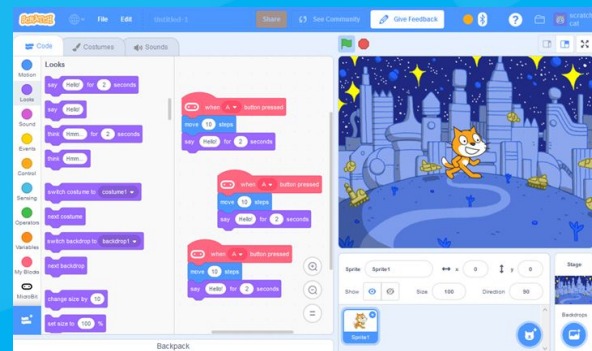
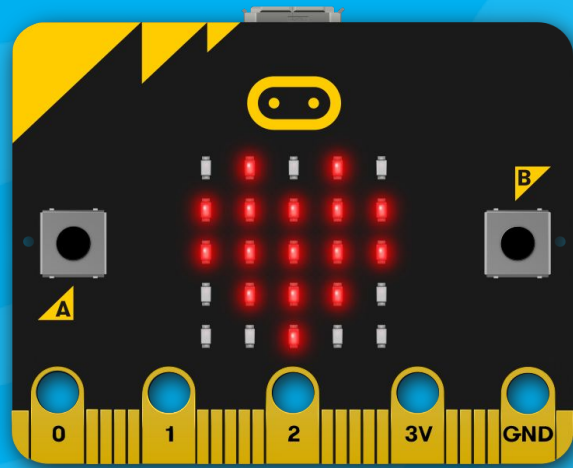
The micro:bit Extension

How does micro:bit work with Scratch 3?

You may be wondering how micro:bit actually works with Scratch. This is due to Scratch 3's brand new 'extensions' feature, this allows you to add extra blocks into Scratch. Amongst the extensions that come with Scratch is the micro:bit. There are many other extensions like Translate and Lego Mindstorms to have a play around with too. The micro:bit extension, once imported, has simple blocks that can control scratch with the micro:bit's onboard features. You can do awesome things like shake the micro:bit to make the cat draw a shape, this makes Scratch much more fun to play with. The micro:bit is connected via a program called Scratch Link, which allows you to interact with your micro:bit over Bluetooth in real time!

Fact:

Scratch is a project ran by the MIT (Massachusetts Institute of Technology) team in Cambridge, MA, US



THE BASICS of micro:bit and Scratch 3.0

Michael Rimicans



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

[@heedt](#)

[heeed.net](#)

Learn how to connect up your micro:bit to Scratch 3.0 and find your way around the brand new micro:bit extension.

If you're reading this magazine then Scratch probably needs no introduction. Its approach to teaching the basic concepts of creating code with a block-based system lowered the bar for everyone who wanted to learn the basics and have some fun doing so.

The current stable edition is Scratch 2.0, found at go.micromag.cc/scratch, is the one that most people will be familiar with. Scratch 2.0 bought some improvements in use from the first version but it still had no simple way to interact with micro:bit or similar devices.

The latest version of Scratch, Scratch 3.0, has been in development over the last few years was recently released to public Beta testing on the 1st of August 2018 with a planned full release in early 2019. This has been a complete rewrite using HTML5 with other modern web technologies and finally removes the need for

Flash. Scratch 3.0 can now run natively in any modern web browser although Internet Explorer will no longer be supported.

This new version also introduces 'Extensions' which are a framework for new functionality to be added to Scratch. One extension that has been added is the micro:bit extension which allows Scratch to communicate with the micro:bit over Bluetooth with the help of a small application called Scratch Link.

Installation

Scratch Link can be found at go.micromag.cc/scratch_3 and has the following requirements:

Windows 10+ or macOS 10.13+ Bluetooth 4.0

Whilst this article concentrates on a Windows installation the macOS version should perform the same once installed by following the instructions for macOS.

With Windows, install Scratch Link by downloading the zip file from the site, unzip it and double click on the resulting file and follow the installation process.

Once installed start Scratch Link by double-clicking on ScratchLink.exe in the c:\Program Files (x86)\Scratch Link directory. The Scratch Link installer does not appear to add the application to the Start menu, however, this can be solved by right clicking on the application file and clicking Pin to Start. Once running it should appear in the taskbar.

Setting up the micro:bit

The next step is to set up the micro:bit to work with ScratchLink. To do this download the hex file from the page, unzip it and install the hex file onto the micro:bit via the normal drag and drop over USB method. Once the hex file has been installed correctly the micro:bit screen should start to scroll a five letter phrase which will help to identify the micro:bit when connecting from Scratch. At this point the micro:bit may be switched over to battery power as all further communication with the micro:bit will be via Bluetooth.

Setting up Scratch 3.0

Make sure Scratch Link is running and open your browser and go to go.micromag.cc/scratch_beta to access the Beta version of Scratch 3 and click on the 'Try It' button.



Welcome to the Scratch 3.0 Beta

We're working on the next generation of Scratch. We're excited for you to try it!

Not Now

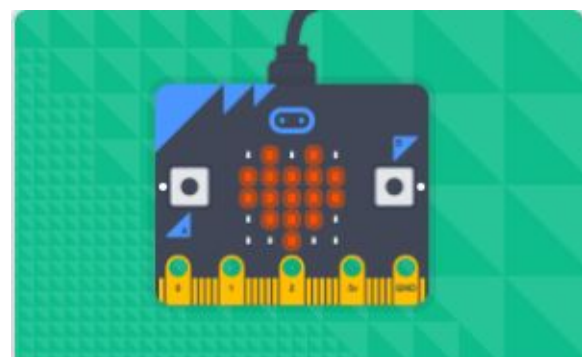
Try It! 

View 2.0 Project

To learn more, go to the [FAQ](#).

Click on the add extension button, it has two white bars and a plus symbol, at the bottom of the block palette.

Choose the micro:bit extension and wait for the extension to install.

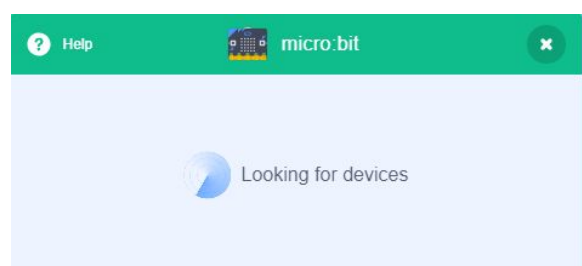


micro:bit

Connect your projects with the world.

micro:bit Extension Button

Once installed extension will then start to search for the micro:bit.



Select your device in the list above.

Refresh 

micro:bit Extension Button



Motion

Looks

Sound

Events

Control

Sensing

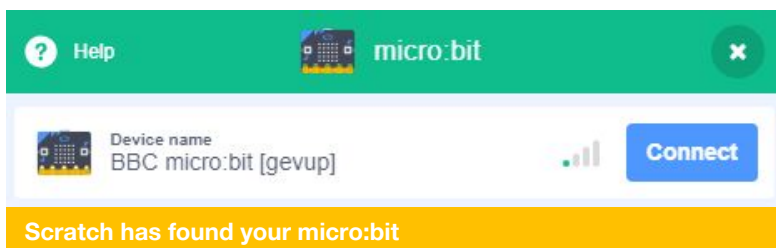
Operators

Variables

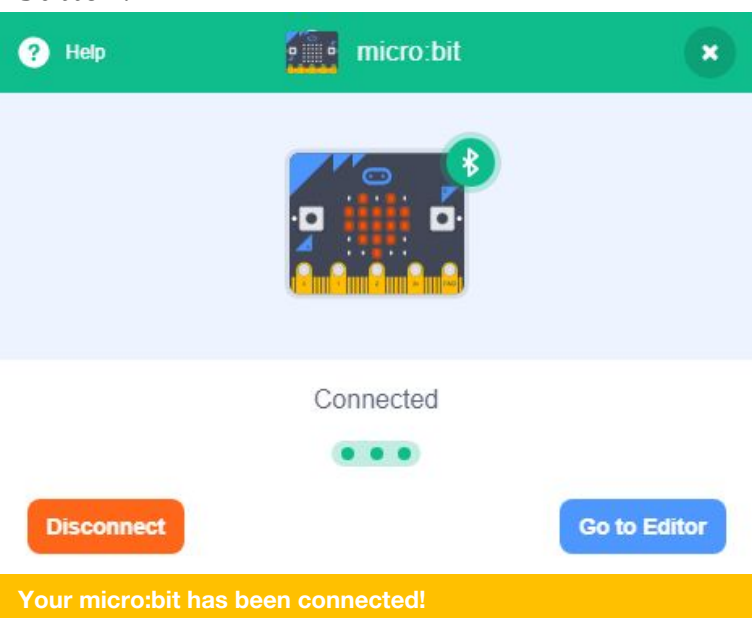
My Blocks

Block Palette

Once discovered your micro:bit will appear on the screen with the same five-letter code that's displayed on the micro:bit screen.



Click on the connect button and Scratch 3 will then connect to the micro:bit. Once connected the micro:bit will display a tick on its screen and the browser will show that's it's connected. To complete the setup click on the 'Go to Editor' button.



Troubleshooting

If the connection can not be made then check the following:

- The micro:bit is powered and has had the connection hex file installed properly.
- Bluetooth is switched on and available.
- The micro:bit extension has been installed in Scratch.

The first check will be easy. If the micro:bit is scrolling the five characters across its screen then it is powered and has the connection hex file installed correctly.

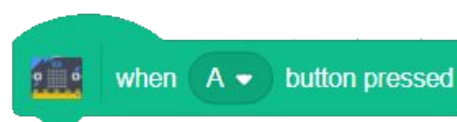
As Bluetooth is required for the micro:bit to talk to Scratch 3 then make sure that your computer has a Bluetooth dongle built-in or plugged in and configured for use. Suitable Bluetooth 4.0 adaptors are available from online stores if your computer does not have one. If the micro:bit has been set up properly and your computer's Bluetooth is active then refresh the browser page displaying Scratch 3 and reinstall the micro:bit extension.

Usage

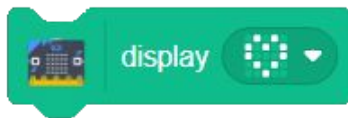
Now that the micro:bit is connected to Scratch 3 you can now start to have some fun.

Clicking on the micro:bit in the block palette will show the available micro:bit blocks. As it stands at the moment, apart from using the micro:bit screen, the extension only handles input from the micro:bit and you are unable, for example, to use it to turn a LED on and off from Scratch 3. Hopefully, using the micro:bit outputs will be added at a later date.

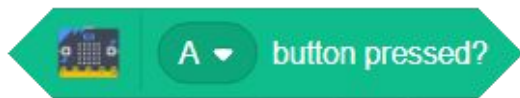
The Blocks



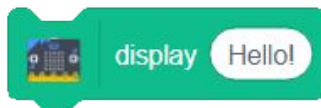
This block reads the status of the buttons. You can select A, B or Any. Pressing both A+B appears not to be supported at this stage.



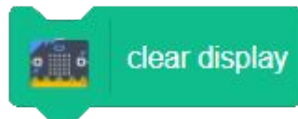
This block can be used where you need to test the the status of the buttons. Again this can sense A, B, or A+B.



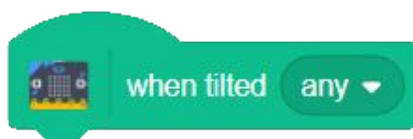
This block will scroll its contents across the micro:bit screen.



This block will clear the micro:bit display.



This block reads the micro:bit angle. It can sense the following tilt positions of the micro:bit: front, back, left, right and any.



This block can be used to test if the micro:bit is tilted.



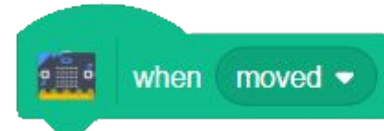
This block stores what position the micro:bit is tilted.



This block allows you to read the input status of pins 0, 1 and 2 of the micro:bit

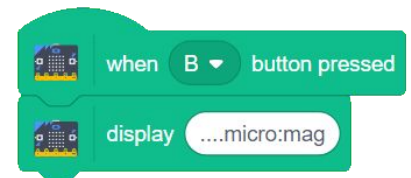


This block detects if the micro:bit has been physically moved. It can sense if it's moved, shaken or jumped.

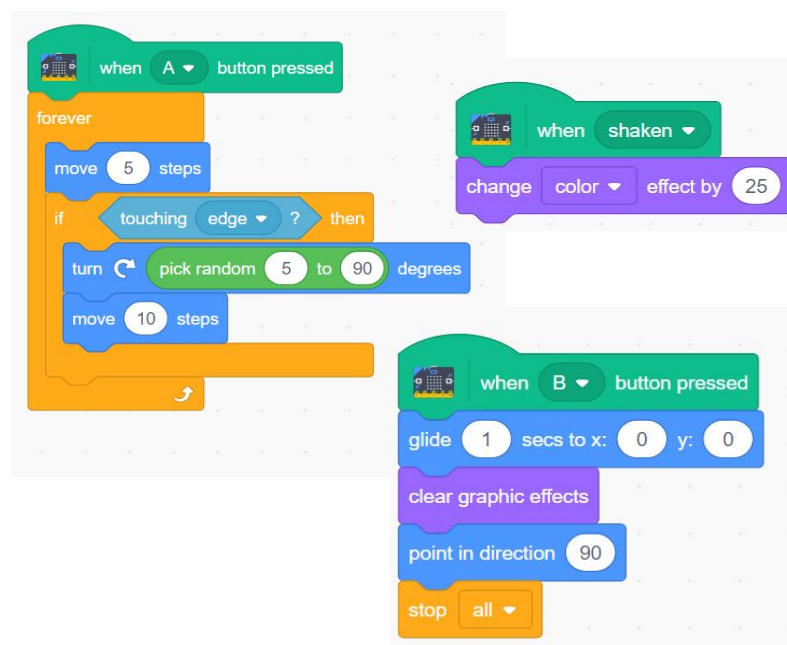


Demos

One well thought out point is that the micro:bit blocks will connect and work seamlessly with the other Scratch blocks. The example below shows a simple name badge application using the micro:bit blocks:



This script shows how the micro:bit blocks work with the existing Scratch blocks. As you can also use the micro:bit motion sensors in Scratch the example below shows how to change the Scratch Cat's colour by shaking the micro:bit whilst using the A and B buttons to stop and start the script.



So, having read this...What can you do?



SCRATCH

Team Interview

We got the chance to talk to two members of the Scratch team. Here's what they had to say...

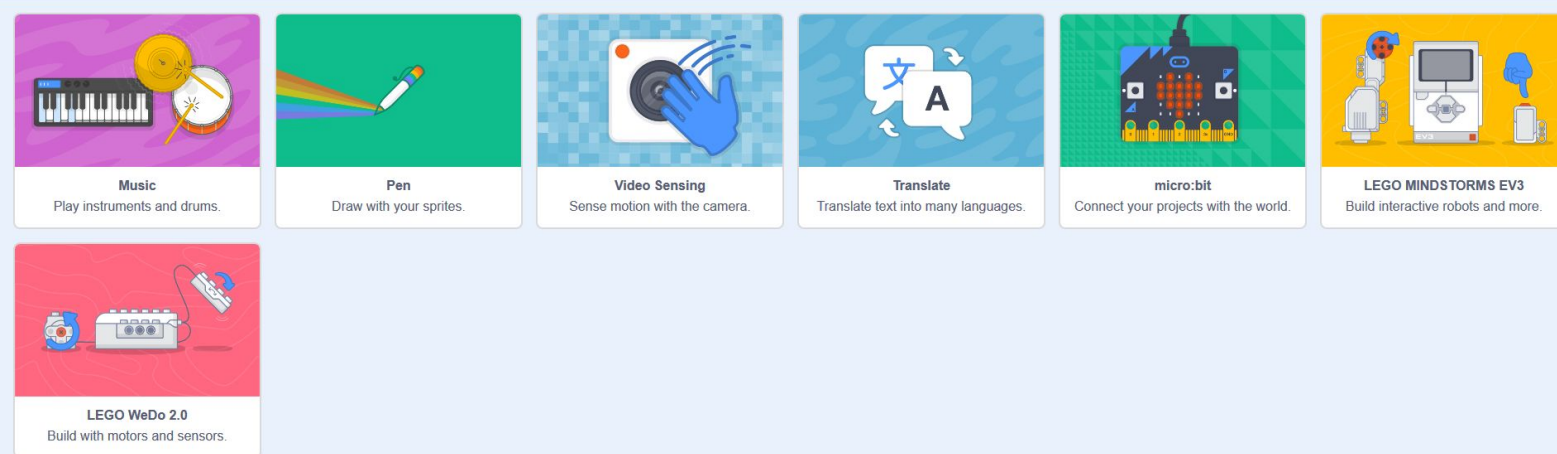
Please introduce yourselves

Kreg: Hi! My name is Kreg Hanning, I'm a researcher with the Lifelong Kindergarten research group at the MIT Media Lab.

Eric: Hello, my name's Eric Rosenbaum, a developer on the Scratch team. I also have a PhD from MIT Media Labs Lifelong Kindergarten Group.

What is your favourite part of Scratch 3 and why?

Eric: I'm particularly proud of the new "set pitch effect" block that lets you use Scratch code to change the pitch of a sound interactively. You can get all kinds of chipmunk-voice and monster-voice effects that way. I'm excited to hear what people do with it! And of course, I'm



The Extensions page in Scratch 3.0 includes support for major platforms like micro:bit, Lego Mindstorms and more as well as support for Translation and Video Sensing.

very excited about the many new extensions we're adding. But maybe my FAVORITE favourite part is the fact that we have a renewed emphasis on making Scratch even more accessible to beginners, by keeping it simple and playful.

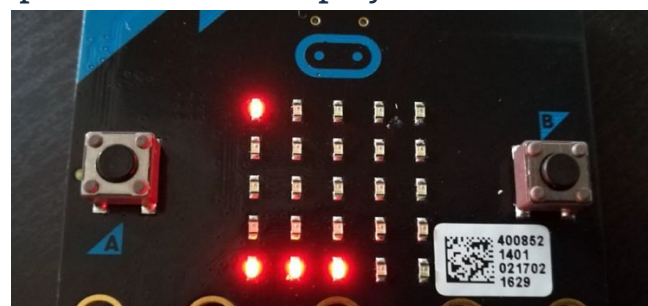
Kreg: I really love the new extension system in Scratch 3.0. It lets you play with all kinds of new Scratch extension blocks for translating text, synthesizing speech, and even controlling the micro:bit! They provide Scratchers with an easy way of interacting with the physical world and I can't wait to see what people create with them!

What inspired you to create an extension for the micro:bit?

Eric and Kreg: Scratch and micro:bit is like peanut butter and jelly- they just go together so well. We're so happy to be able to mash together the digital flexibility of Scratch, and the physical flexibility of the micro:bit. By combining them you can make all kinds of new projects that blend making in both the physical and virtual world. You can create things like new musical instruments out of household objects, game controllers out of craft materials and

recyclables, custom costumes that let you take on a role in a digital story, and a whole lot more. We really care about what we call tinkerability, which is the way you can learn about how something works by trying it out and seeing the results right away. It's the sense that you can safely try anything, and there are infinite things to try. We design Scratch with this in mind, and the micro:bit extension is the same way. It's "tethered" so that the connection is always live. For example, you can click on a block in Scratch and see the micro:bit display update right away. The coding process isn't separate from the play-it's a single continuous process.

Another thing we love about the micro:bit is the fact that it's a truly accessible physical computing platform, and it's empowering so many children around the world, which matches the spirit of the Scratch project.



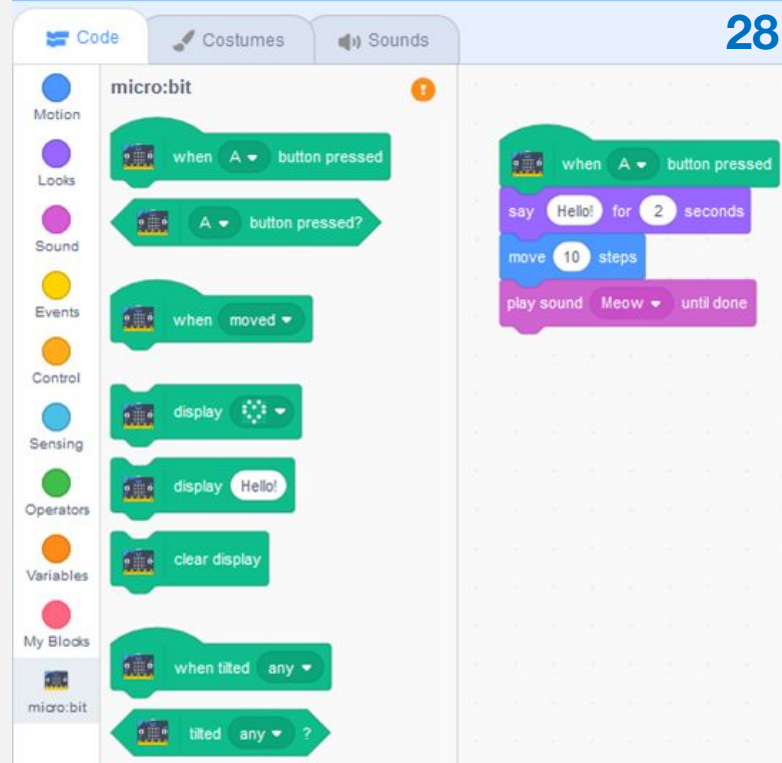
The micro:bit works with Scratch 3.0 using BLE which means you can 'live code' the micro:bit.

Do you see this as an important step in the development of Scratch and Physical Computing in schools?

Kreg: One of the main reasons we are excited to see the micro:bit used with Scratch in schools is that it can provide a playful approach to physical computing. We hope that, like Scratch, the micro:bit will provide a physical platform for children to explore their ideas regardless of what they feel passionate about. So whether a child loves art, music, or even storytelling, we hope that Scratch and the micro:bit can help promote curiosity and personal expression in physical computing.

What are the main differences between Scratch 2 and Scratch 3?

Eric: Scratch 3 is a complete rewrite of Scratch using different technology (javascript, in place of Flash), but it's backwards compatible, so generally all the features of Scratch 2 will still be there and existing projects will work as before. Many parts of the Scratch editor have been redesigned (I'm particularly a fan of our new sound editor, and our new bitmap and vector costume editors), and it comes with hundreds of new images and sounds for kids to use in their projects. Another big change is that there are a lot of new extensions to Scratch, which adds new blocks for things like the micro:bit and LEGO robotics kits, and many more that we're hoping to add in the future. We're also excited for Scratch 3 to be able to better reach kids where they are, as we work on creating great Scratch experiences on tablets and mobile devices.



The micro:bit extension loads into Scratch with a few simple clicks. It's easy! You can also integrate the blocks into standard scratch ones.

What was the hardest part of making the micro:bit extension?

Eric and Kreg: One big challenge is choosing what to include. The micro:bit has a lot of powerful features, but we wanted to keep the Scratch extension really simple. It's a balance between making sure it's understandable and fun for beginners, while still providing a huge number of creative possibilities. Mitch Resnick (founder of the Scratch project) talks about designing for a low floor and a high ceiling: making it easy to get started, but possible to do more complex projects. In the workshops with children and adults we've done so far, we've found that the compromise we've reached accomplishes this pretty well!

Scratch 3.0 gets released in January 2019 but you can try the BETA today over at:

beta.scratch.mit.edu



@eyeleanking

Eileen King teaches young people to build things with their hands and with computers in the Twin Cities area. You'll find her other tutorials on Medium ([@eileenaking](#)).

Turning your micro:bit into a Game Controller

With a few alligator clips and some imagination, you can build your own micro:bit-powered controller for a Scratch 3 game.

You Will Need:

- A computer with Scratch Link installed
- A micro:bit with batteries
- 4-8 alligator clips
- Something that can complete a circuit: buttons (or other momentary switches), conductive tape, aluminum foil, metal paper fasteners, etc.
- Optional: Crafting materials

gold pins. Tweak a game's code to use these blocks, and you can build a micro:bit-powered game controller from anything that conducts electricity!

Choosing a game

First, you'll need to make or find a game to work with. Your micro:bit has three pins that can be turned into buttons, so you'll need to start with a Scratch game that can be played with no more than three keyboard keys. Lots of games can be played with just the space key, the left and right

Scratch 3 comes with some exciting new micro:bit blocks, including ones that fire when connections are made between the micro:bits

arrow keys, or left/right/up: if you've ever tried the Pong Game or Catch Game tutorials, for example, then you've got a game you can use! If you don't already have a game made that will work, then you have two options: you can either build a game yourself (you could try one of those tutorials, or look for other options online), or you can remix someone else's game (try the Scratch Wiki's list of example platformer games, or search for games inspired by Flappy Bird, Tetris, or Space Invaders - anything that can be played with no more than three controls).

While Scratch 3 is still in beta, you won't be able to remix games made with Scratch 2. For now, if you want to work with a game you or someone else made in Scratch 2, you'll need to download it to your computer, open Scratch 3, then upload the project.

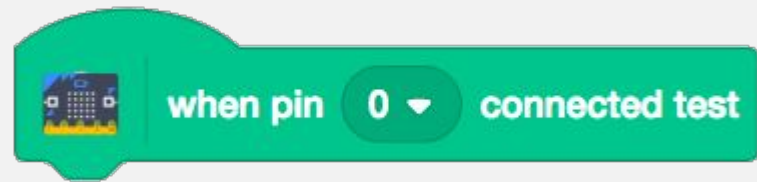
Next, spend some time playing the game! Make sure you're familiar with its controls and can confidently explain how to play - that will help you figure out how to find the parts you want to change later.

Controlling the game with a micro:bit

Once you've got a game running in Scratch 3 that uses no more than 3 different inputs as controls, you're ready to start controlling it with your micro:bit!

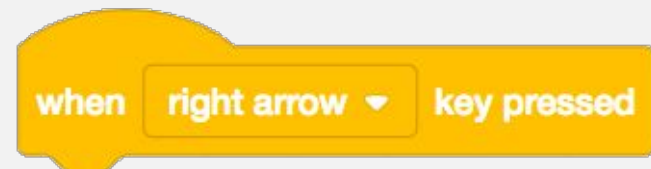
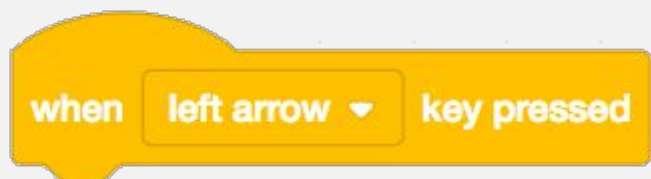
Here's what you'll need to know: your micro:bit has some gold-coloured connection on spots on its bottom edge that are called pins. There is a

Scratch block that looks like this:

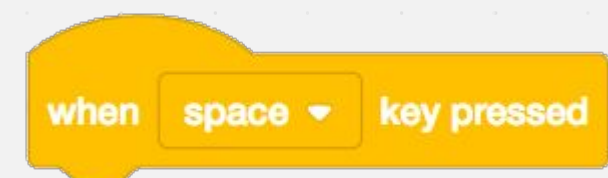


This can go on top of a script, and it makes that script run whenever a connection is made between the pin that says GND (for ground) and the pin that says 0, 1, or 2 (whichever one you've chosen on the block). This connection can be made by your fingers, by two pieces of metal touching, or by anything else that can conduct at least a little bit of electricity!

To use these pin connections to control your game, you'll need to identify the parts of the code that currently handle input, then change them to use micro:bit pin connection blocks instead. For example, if you're working on a game controlled by the left/right and right arrows, then you'd look for scripts that start:



If the game uses the spacebar, look for



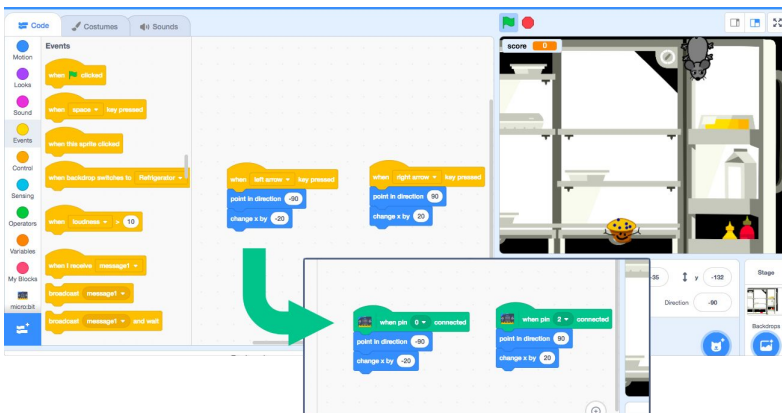
If you're working with a remix of someone else's project, it's totally okay to not understand every single block in the project - just focus on finding

the scripts whose Events blocks match the game's controls.

When you find one of these controller scripts, detach the code from the Events block it's attached to, and put in this block instead:

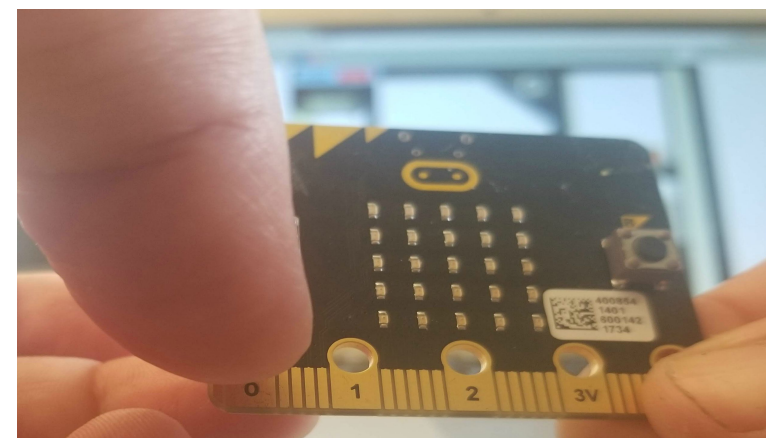


Be sure to choose a different pin for each control!



On pin connected blocks

After you've found and replaced the keyboard controls with pin controls, give it a test run: can you now play the game by making connections between ground and the numbered pins? Touch and release the pins to ensure that the new controls work the same way as the original ones!



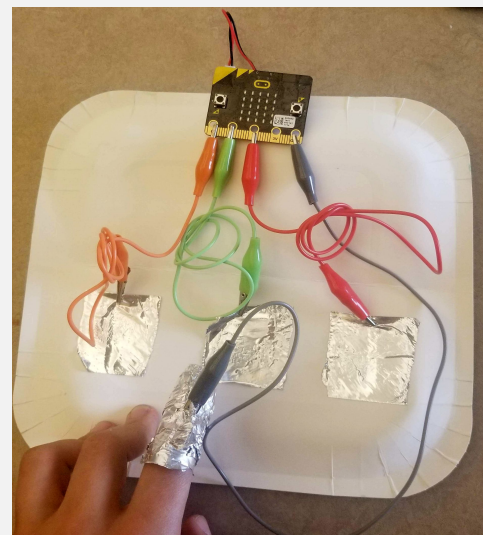
Test your game by holding ground, then touching and releasing another pin.

Creating a controller

Now that the game can be played by making pin connections, you can get creative with how you'll make those connections! Start by clipping an alligator clip onto each pin, and test your game by touching the other ends of the alligator clips together to make pin connections. It should still work the same as it did when you were putting your fingers directly on the pins!

Now, to turn it into a custom controller, think about what other materials you could work with: anything that will conduct electricity can be clipped onto the alligator clips and touched together to control the game. What do you want the player to do to make those connections?

There are lots of possibilities! For example, try creating a conductive surface to clip to each of the numbered pins, then tapping those surfaces with something clipped to ground.



An example game controller made of foil and conductive tape.

Here, we clipped each pin to a piece of conductive tape, then clipped the ground wire to a piece of foil. We wrapped the foil around a finger, but you could also wear it as a bracelet, sculpt it into a

magic wand, or come up with your own creative idea.

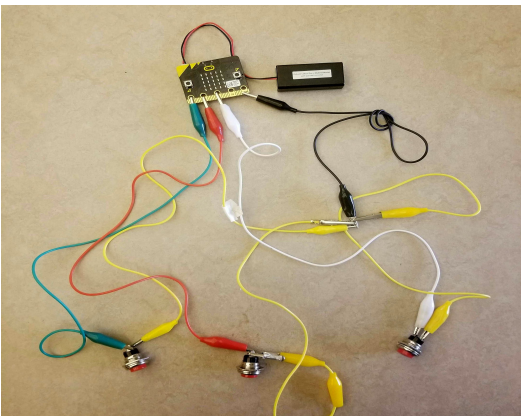
Don't have conductive tape? Foil alone can be fun, especially if you're willing to get a little active, like we did with this foot controller:



One foot remains stationary on a foil pad clipped to ground; the other foot taps the left, right, and jump "buttons."

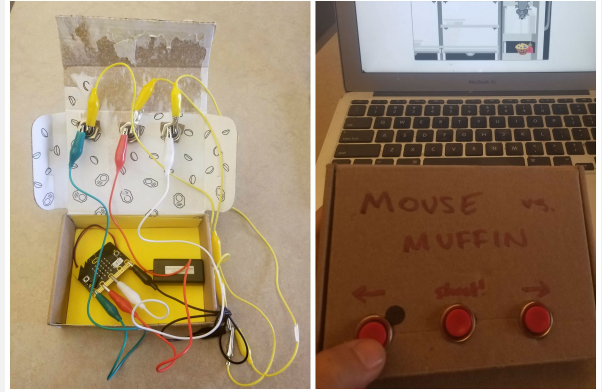
For an additional challenge, try it as a team game: three people stand with one foot on the grounded pad, then each person has one of the other controls to tap with their other foot. It's tougher to coordinate than you might think!

If you have access to buttons or another type of momentary switch, those can be fun to use! Each switch will need to be clipped to a numbered pin and to ground - one wire on each leg of the switch. Be careful that the alligator clips aren't in contact on the base of the switch, though!



If you're using more than one button, it's easier to clip alligator clips to each other than to try to cram more than one clip directly onto the ground pin.

Finally, a little bit of crafting can turn a tangle of wires into a polished custom controller:



Whatever you decide to build, make sure to test it thoroughly. Then find someone to share it with!

Troubleshooting:

If your controller suddenly stops working, here are some good things to check:

- Are the micro:bit and computer still communicating? If the micro:bit restarted or moved out of range, it may need to be paired again.
- Are the alligator clips askew? If they're not squarely clipped on the pins, perpendicular to the bottom of the micro:bit, that can cause problems.
- Are there alligator clips touching each other that shouldn't be? Especially if you choose to put wires inside a small container, it's possible for clips to make unwanted connections with each other.

John Lynch



John is a professional teacher, half-decent carpenter, and amateur game designer. Building game controllers with the Micro:Bit connects all his pursuits together.

@Mittensbrother

mittensbrother.com



Explode the Controller

@ Learn 2 Teach, Teach 2 Learn

Connecting physical play and video games with see-saws, Micro:Bit, and Scratch 3.0

This summer, the youth teachers at the Learn 2 Teach, Teach 2 Learn program are blowing up video games with new, homemade controller designs. Instead of sitting down and pressing buttons to play, a player uses these devices by running, jumping, and balancing to interact with the game. We call it the 'Explode the Controller' project.

For example, we created a simple Avoider game in Scratch 3.0 where the player must dodge incoming asteroids by balancing on the see-sawing '♥ Rocks Board' and tilting it to move the ship. A micro:bit attached to the controller sends tilt angle data from its accelerometer to the game via Bluetooth, and Scratch uses this information to change the

position of the ship sprite. Players of all ages love how the game challenges both their coordination and reaction time.



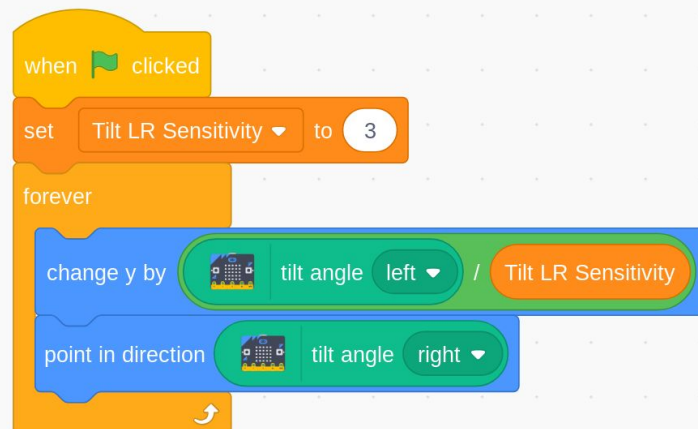
Trinity avoids asteroids on the ♥ Rocks Board



Closeup of tilting ♥ Rocks Board



Detail of the Scratch 3.0 Avider game



Micro:Bit blocks are included in Scratch 3.0

Why do we want to add physical activity to our games? Well, it's loads of noisy fun to play using your whole body, but we also design games like this to get young people excited about making code themselves. Boston's Learn 2 Teach, Teach 2 Learn program empowers the most underrepresented youth in our community to move from being consumers of technology to producers and creators. Each spring, 30 high school students gain skills in 6 maker technologies, including microcontroller coding with micro:bits. In the summer they build projects, then spend the last three weeks teaching summer STEAM Camps in the Boston neighborhoods most in need of educational resources. Many of these children have never experimented with code before, but game controllers like the ♥ Rocks Board help to inspire these beginners because even basic game projects can be made fun immediately when connected to physical activity.

According to program director Susan Klimczak [@zackboston](#), Learn 2 Teach, Teach 2 Learn brought activities like 'Explode the Controller' to more than 600 children at 25 community organizations.

We hope to inspire micro:mag readers to try out their own homemade controller ideas, too! What kinds of motions, dances, or other challenges can you imagine for players in your next video game project? What will your controller look like, and how can tech like the micro:bit connect it to your code?

To learn more about the fantastic making and education happening at the Learn 2 Teach, Teach 2 Learn program, follow them on twitter [@Learn2TeachSETC](#).

Want to try out the ♥ Rocks Board? It features 3D printed parts and simple materials to be quickly and easily reproduced. Build instructions for this and other 'Explode the Controller' devices are available at

go.micromag.cc/explodethecontroller

micro:**mag**

Contribute to micro:mag

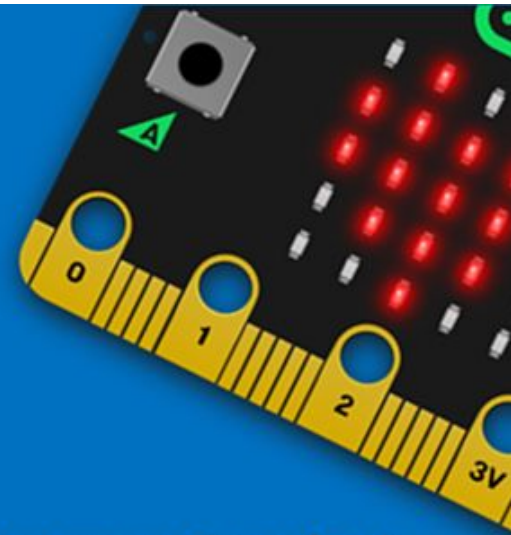
If you've made a cool micro:bit powered project,
why not write for us?

Our magazine is written by the community for the
community and we even offer help and support if
you'd like help writing your article.

Fill in the form:

micromag.cc/contribute

To get in touch, email us at: hello@micromag.cc



Get fresh copies of micro:mag delivered to your inbox

Two black Micro:bit boards are shown. The top one is angled and has several red LEDs lit up. The bottom one is partially visible at the bottom right, showing its numbered pins (0, 1, 2, 3V, GND).

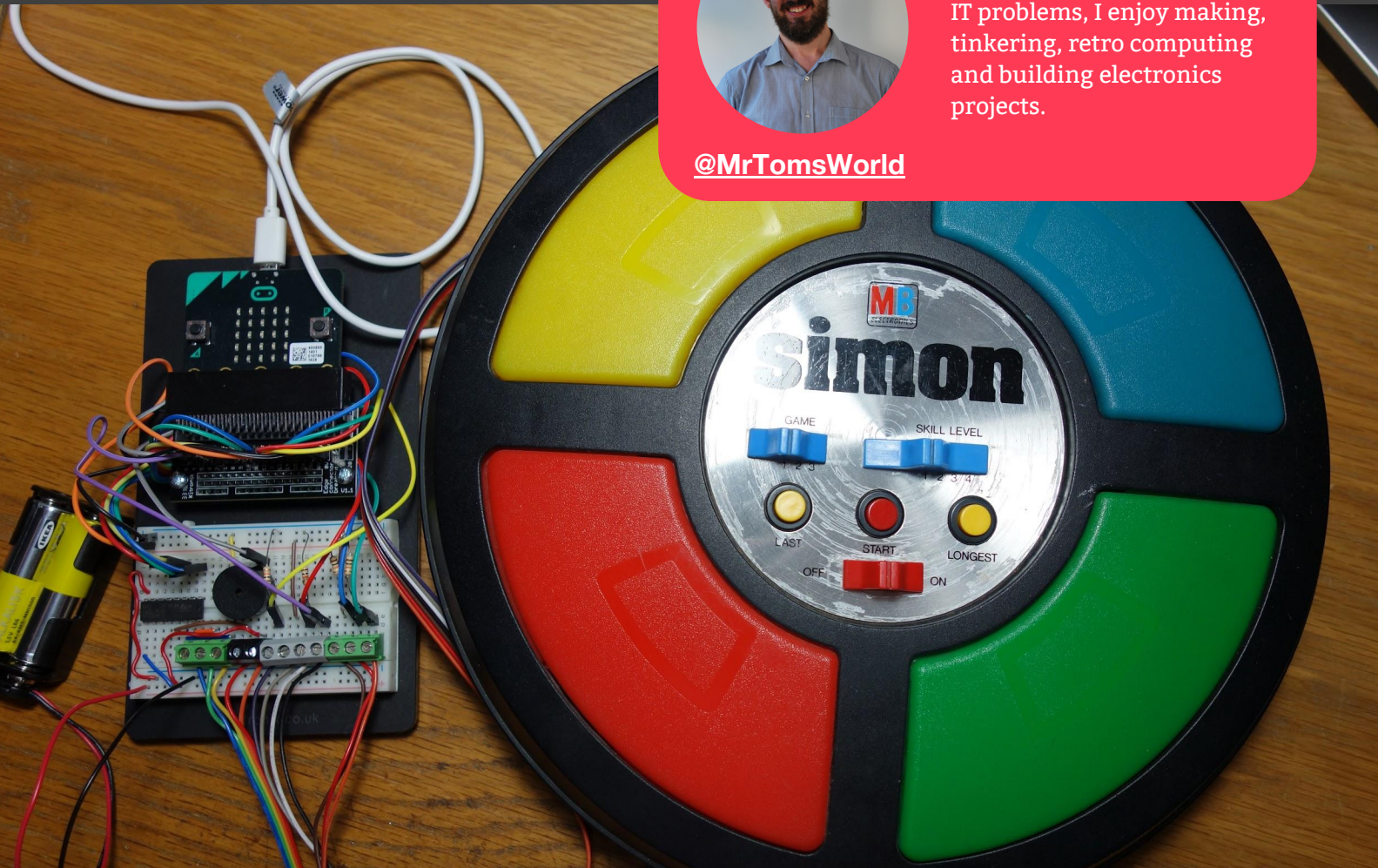
Make sure you never miss an issue of the unofficial community magazine for micro:bit lovers - we'll email each issue as soon as it's released (and we won't ever spam you or give your details to anyone else)

To sign up, go to
micromag.cc/email



Thomas is an ICT Support Technician. When not fixing IT problems, I enjoy making, tinkering, retro computing and building electronics projects.

@MrTomsWorld



The completed Micro Simon game, complete with external micro:bit and the associated circuitry.

Micro Simon

Take a classic MB Simon game and combine it with a micro:bit.

This project is based on an excellent micro:bit MicroPython Simon Game example I found on the MultiWingSpan website go.micromag.cc/multiwingspan This gave me an idea, could a vintage MB Simon game be controlled by a micro:bit? The Milton Bradley Company was an American board game manufacturer established by Milton Bradley, in Springfield, Massachusetts, in 1860. He enjoyed

early success when he packaged a series of games, including The Checkered Game of Life, in a pocket-sized game pack (the country's first "travel" game) designed for soldiers during the Civil War.

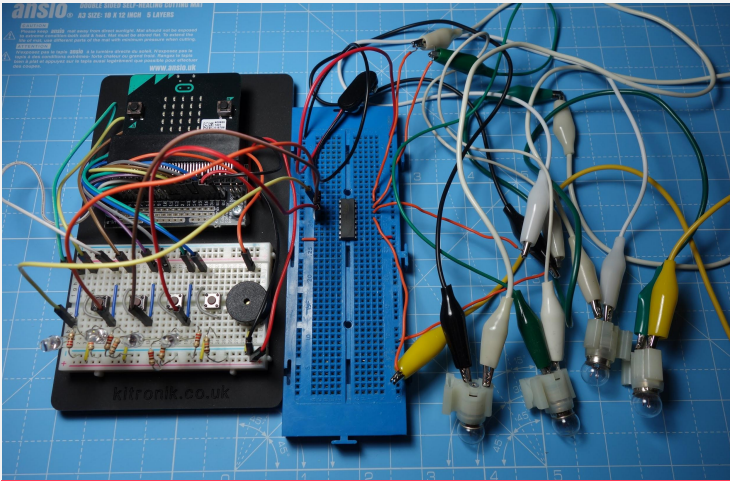
Simon is an electronic game of memory skill. The device creates a series of tones and lights and requires a user to repeat the sequence. If the

user succeeds, the series becomes progressively longer and more complex. Once the user fails or the time limit runs out, the game is over.

The original version was manufactured and distributed by Milton Bradley, launched in 1978.

Testing

With this idea in mind, I had a quick hunt on eBay for a broken Simon game, after a while one came up so I duly purchased it. While I waited for the game to arrive, I knocked up a quick circuit with some low voltage light bulbs and a ULN2003A Darlington transistor array (An integrated circuit containing several transistors) to drive the bulbs.

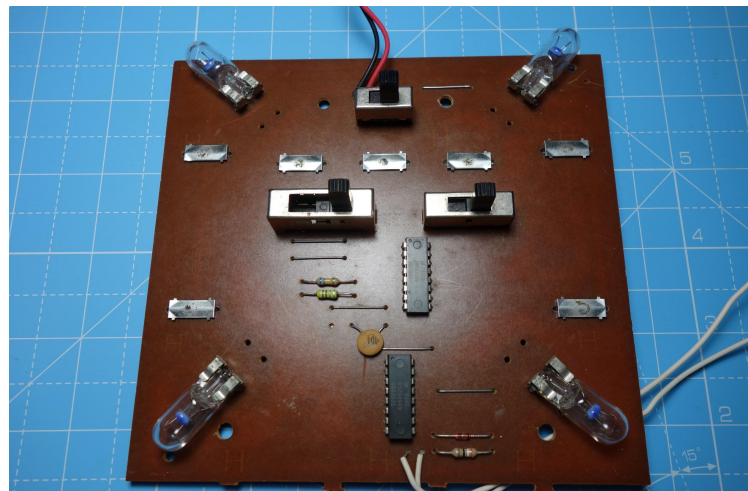


Here I have connected the micro:bit Simon game I build on a breadboard to another breadboard with a ULN2003A on and this is connected to the bulbs.

When the game arrived I duly took it apart, well it sadly didn't work and was pretty beaten up. Oh well!. I found some useful information about these classic Simon games here:

go.micromag.cc/classicsimon included on the site was a schematic which helped to trace out the existing Printed Circuit Board (PCB) layout so I could tap into the existing bulb and switch circuits.

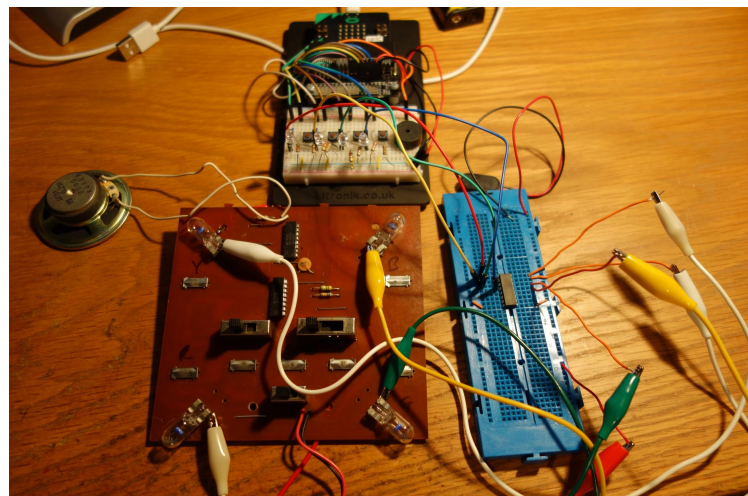
The original full-sized game uses a custom MB Electronics processor which performs nearly all of the functions for the game. The only other significant piece of electronics is an additional integrated circuit which interfaces the processor to the light-bulbs and the built-in speaker.



I removed the main PCB from the Simon game enclosure so I could trace out the connections.

Next Steps

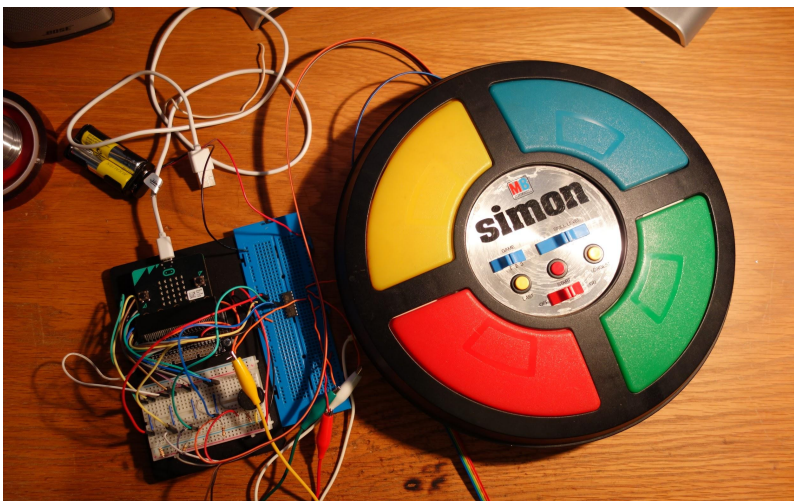
I removed the main PCB from the enclosure and connected some wires temporarily to the existing PCB to continue testing.



Here I have connected the two breadboards to the main PCB to check that everything works.

Once I was happy with this, I removed the existing custom processor additional integrated circuit and soldered some new wires to the existing colour switches and bulbs. I then connected these to my existing breadboard layout and gave it a quick whirl, it worked brilliantly.

Completed Project



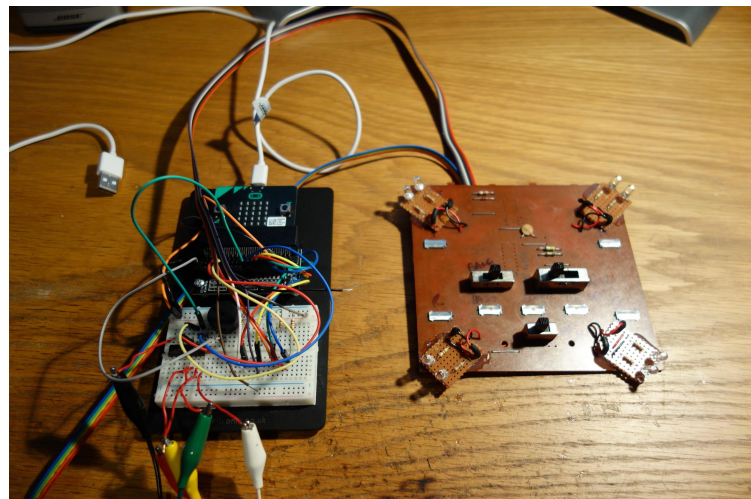
Showing the completed project with everything back together.

I took my Micro Simon game out on the road and it performed well but the bulbs weren't very bright. Despite some experimenting to see if I could get the bulbs any brighter, I couldn't so I decided to build some Light Emitting Diode (LED) modules to replace the existing bulbs. I brought some high-intensity white 30000mcd [1] LEDs and set about building the LED modules on a small piece of stripboard.

The light output of LEDs is rated in either candela (also candle) or lux, with the candela being the more common. Both are a measure of luminous intensity. The higher the candelas/lux, the brighter the light. Standard LEDs emit only a modest amount of light, not even 1 candela.

They're rated in millicandelas or thousandths of a candela. Millicandela is typically abbreviated as mcd.

I removed the existing bulb holders from the MB Simon game PCB and glued the new LED modules onto the existing PCB.



Showing the new LED modules attached

micro:mag

Contribute to micro:mag

Have a cool project to share?
Write about it for the next
issue of micro:mag.
We'd love to hear about it!

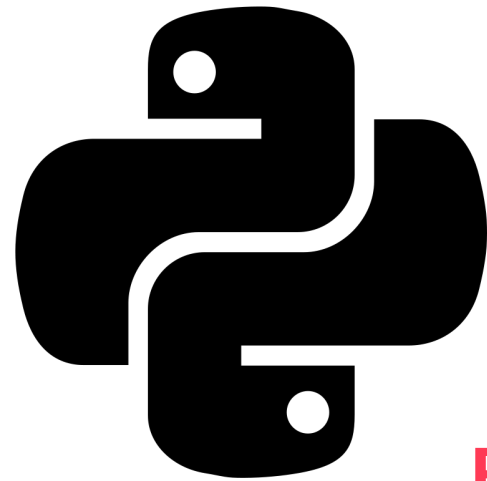
Fill in the form at:
micromag.cc/contribute

Any questions? Email us at:
hello@micromag.cc



I am an avid open source programmer. Checkout s2m, (go.micromag.cc/s2m) a program that allows you to control your micro:bit using Scratch 2.0 on Windows, Mac and the Raspberry Pi.

@BrassFigLigee



Python Debugging: Ninja Tips And Tricks

Python Debugging Techniques For The micro:bit

You Will Need:

- A micro:bit and USB cable
- mu-editor installed on your PC (<https://codewith.mu/>)

You've just finished writing your latest Python creation for the micro:bit. You're trying out some new Python features you've just read about, and now you wait with joyful anticipation as the LED on the back of the micro:bit continues to blink as your program loads. Finally, the blinking stops, your program is loaded, and then – what is this? Is it not working? In an instant, you go from joyful anticipation to a feeling of confusion, disappointment, and maybe even frustration. No worries, with this article as your guide, you will be well on your way to becoming a confident Ninja Debugger, able to confront the most stubborn of bugs.

Ready to begin? Take a deep breath, inhale the

confidence and exhale those buggy negative feelings. Now you are ready to walk the path of a Ninja Debugger.

So What Exactly Is A Bug Anyway?

A bug is simply a programming error, inadvertently introduced into a program by the programmer. Essentially there are 2 types of bugs, syntax errors, and run-time errors (sometimes known as logic errors).

Syntax Errors

A mistyped Python keyword or a misspelt variable name are examples of syntax errors. Forgetting to indent your code properly is another example of a syntax error. There are many types of syntax errors, but the good news is that there are tools available to locate and identify syntax errors for us. In this article, we will use the mu editor to find syntax errors. Correcting a syntax error found by mu is as easy as going to the line that mu identified for

us, modifying the code and then rechecking that our changes are correct. Easy-Peasy – minimum stress.

Run-time Errors

Run-time errors occur while the programming is running. An example of a run-time error is one that executes an illegal operation. Attempting to divide an integer by zero is an instance of an illegal operation. With this type of error, the interpreter “throws an exception” that identifies the cause of the error and the line number on which it occurred. In the case of the micro:bit, the exception information is scrolled across the display. This type of bug is usually as easy to correct as a syntax error since the interpreter informs us of the offending line number and the cause of the error. The only difference is, that this bug is not found until the program is run. We will demonstrate this very bug in just a bit.

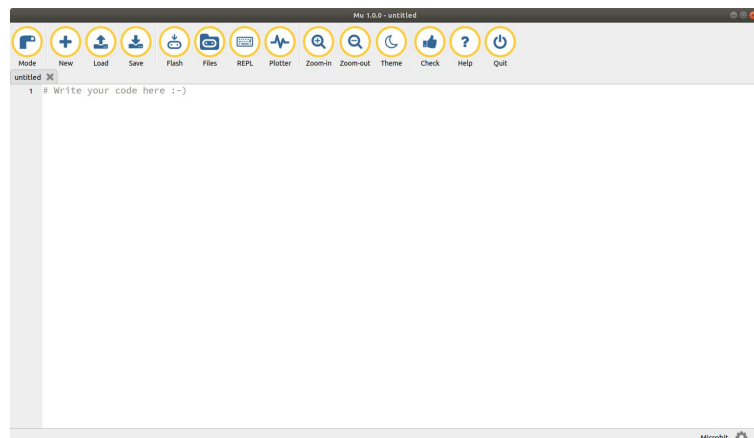
Another type of run-time error is a code design error. For example, let’s say, we’ve written a program that accepts a value expressed in grams and we want the program to convert that value to ounces. When we test the program, we notice that the result returned is not the expected value. With this type of error, we need to dig into the code. Perhaps we selected the wrong conversion formula to use, or perhaps we are using the correct formula but when we converted the formula to Python code, we introduced a bug. Run-time design errors can be challenging to fix because we need to become

“code detectives “ to figure out their cause. This takes practice, and with experience, becomes easier and easier to do.

Finding And Fixing Syntax Errors Using The Mu Editor

Now that we know what bugs are, let’s use the mu editor to find and squash some syntax errors!

If you haven’t already done so, install the mu editor on your computer and then start it.



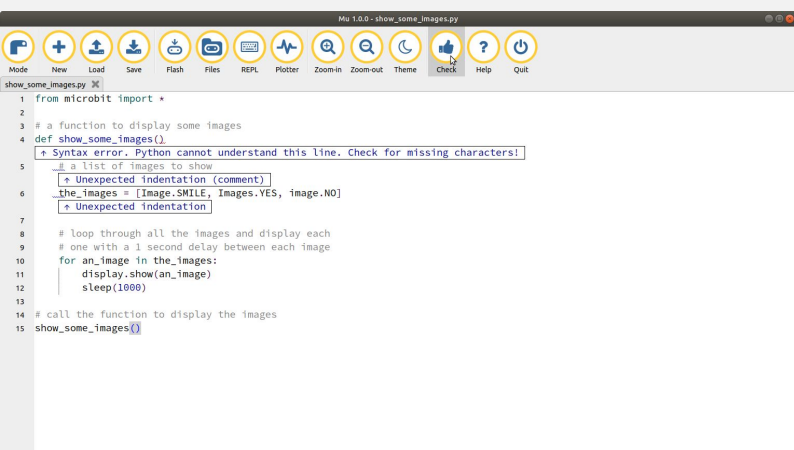
Next, copy and paste the following code into mu.

```
from microbit import *

# a function to display some images
def show_some_images()
    # a list of images to show
    the_images = [Image.SMILE, Images.YES,
image.NO]
    # loop through all the images and display each
    # one with a 1 second delay between each
image
    for an_image in the_images:
        display.show(an_image)
        sleep(1000)
    # call the function to display the images
show_some_images()
```

:feature Python Debugging

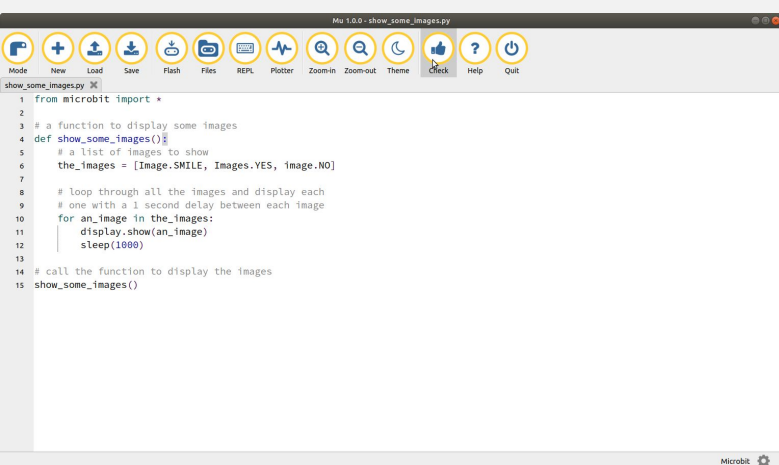
This code contains several syntax errors. Let's use the mu editor to find the errors for us. Click on the "Check" button at the top of the mu editor. Mu will show all of the syntax errors it finds.



```

1 from microbit import *
2
3 # a function to display some images
4 def show_some_images():
5     # a list of images to show
6     the_images = [Image.SMILE, Images.YES, image.NO]
7
8     # loop through all the images and display each
9     # one with a 1 second delay between each image
10    for an_image in the_images:
11        display.show(an_image)
12        sleep(1000)
13
14 # call the function to display the images
15 show_some_images()
  
```

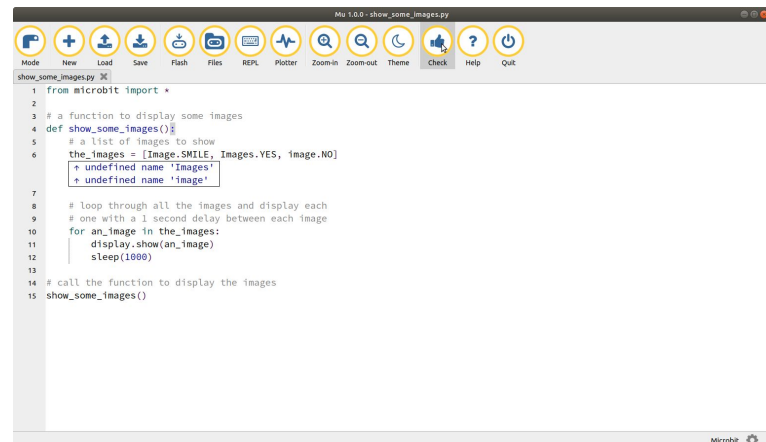
Let's fix the first error which occurs on line 4 by adding a colon at the end of the line. After fixing this error, click on "Check" again.



```

1 from microbit import *
2
3 # a function to display some images
4 def show_some_images():
5     # a list of images to show
6     the_images = [Image.SMILE, Images.YES, image.NO]
7
8     # loop through all the images and display each
9     # one with a 1 second delay between each image
10    for an_image in the_images:
11        display.show(an_image)
12        sleep(1000)
13
14 # call the function to display the images
15 show_some_images()
  
```

The syntax error on line 4 has been fixed, but notice that now all the other errors identified when we first clicked "Check" seem to be gone. When we fix a syntax error, and click on "Check", the mu editor will limit the check to the lines that we modified. We need to click "Check" again to check for any additional syntax errors.



```

1 from microbit import *
2
3 # a function to display some images
4 def show_some_images():
5     # a list of images to show
6     the_images = [Image.SMILE, Images.YES, image.NO]
7
8     # loop through all the images and display each
9     # one with a 1 second delay between each image
10    for an_image in the_images:
11        display.show(an_image)
12        sleep(1000)
13
14 # call the function to display the images
15 show_some_images()
  
```

The indentation errors identified on the lines 5 and 6 that were identified when we first clicked "Check" are no longer there, and now some new errors for line 6 have been identified. What is going on? Here's the story - sometimes a single syntax error will have a cascading effect on lines below the error. Once we fix the initial error, the cascading errors may disappear, and when we recheck the code the next true error will be identified.

Now it's your turn. Fix the errors on line 6. Trying fixing each one separately and then check after each fix to make sure you've correctly fixed the error. Then click "Check" a second time to continue checking the file for any additional errors that may still be lurking.

When you have fixed all the syntax errors, click on mu's load button to install and run the program on the micro:bit.

You should see the micro:bit display the 3 images on the display.

Finding And Fixing Runtime Errors Run-time Exception Errors

Let's purposefully create a program that will

Let's purposefully create a program that will generate the run-time error that occurs when we divide an integer by zero.

```
from microbit import *

# divide a number by zero
dividend = 5
divisor = 0

# an illegal operation - divide by zero
# this will "throw" a ZeroDivision
if dividend/divisor == 0:
    display.show(Image.YES)
else:
    display.show(Image.NO)
```

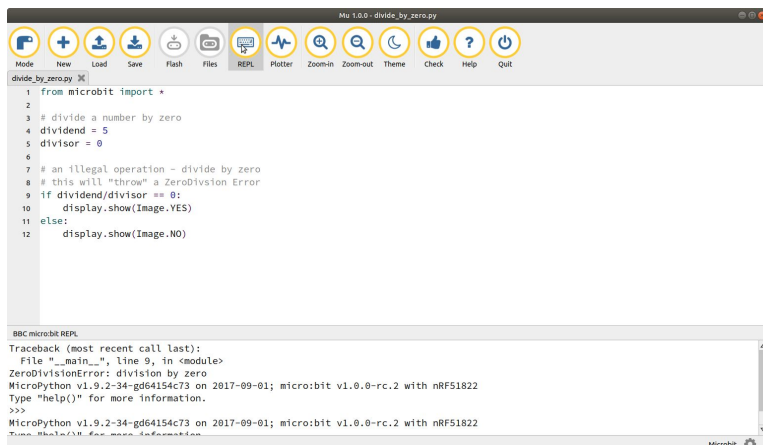
Check the file for syntax errors and then flash it to the micro:bit by clicking the Flash button.

Python recognizes that dividing by zero is an illegal operation and will "throw" an exception. The exception and its information are scrolled across the micro:bit display. Because it is sometimes difficult to read the scrolling exception information, there is an easier way to view the exception information.

On the mu editor, click the REPL button. This will open a window at the bottom of the editor that displays the exception information. We see that on line 9 a ZeroDivisionError occurred. We would then first modify our code so that this illegal operation did not occur, then check the changes for syntax errors and if there are none,

finally we would re-flash the micro:bit.

AN IMPORTANT NOTE: Before flashing the micro:bit, make sure to click the REPL button to hide the REPL window. If the REPL window is open, mu will not allow your code to be flashed. That is because both the REPL and the flash utility use the serial port. Only one of these utilities can be active at a time.



```
Mo 1.0.0 - divide_by_zero.py
Mode New Load Save Flash Files REPL Plotter Zoom-in Zoom-out Theme Check Help Quit
divide_by_zero.py X
1 from microbit import *
2
3 # divide a number by zero
4 dividend = 5
5 divisor = 0
6
7 # an illegal operation - divide by zero
8 # this will "throw" a ZeroDivision Error
9 if dividend/divisor == 0:
10     display.show(Image.YES)
11 else:
12     display.show(Image.NO)

BBC micro:bit REPL
Traceback (most recent call last):
  File "__main__", line 9, in <module>
ZeroDivisionError: division by zero
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; micro:bit v1.0.0-rc.2 with nRF51822
Type "help()" for more information.
>>>
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; micro:bit v1.0.0-rc.2 with nRF51822
Type "help()" for more information.
```

Run-time Design Errors

We discussed this type of error earlier using the gram to ounce converter example. This was an example of a pure software bug. For pure software bugs, it is sometimes easier to copy the code to your desktop computer and use a debugging program such as Thonny or PyCharm to isolate the problem. When fixed, you can copy the corrected code into mu and try out your changes. On the rare occasion, you may encounter a bug that is due to an interaction with the hardware on the micro:bit. The hardware is not the problem, but the software interaction with the hardware causes the bug.

Here's an actual case of this type of run-time design error that I encountered. When I released s2m, a program that allows one to control a

micro:bit using Scratch 2.0 (on Windows, Mac or Raspberry Pi), I accidentally introduced an interesting bug. s2m: go.micromag.cc/s2mdocs

One of the features of the s2m program is to allow one to connect a digital output device, such as an LED, to pins 0, 1 or 2 of the micro:bit and then control the LED from a Scratch script.

When I did so and turned the LED on, it would turn itself off shortly thereafter. This was a bug – the LED was supposed to stay on until it was commanded to turn off.

In the case of this bug, the micro:bit hardware was clearly involved, and so some analysis was needed. It was curious that I could turn the LED on, but it would quickly extinguish itself. The question was, what in my code was “automagically” turning off the LED? If I turned the LED back on, it would turn itself off again. So I started to think and realized that the problem seemed to be periodic. The LED would extinguish itself in the same amount of time each time it was enabled. So, I checked all of the code that called `sleep()` and I did not immediately see any problems. Giving the problem some further thought, I remembered that s2m allows one to connect pins 0, 1 and 2 as inputs to things like switches, and it monitors those pins and reports back to Scratch. The monitoring is done within the main loop of the program which contains a short `sleep()` instruction. So I deduced (not so elementary my dear Watson) that the problem must be somewhere in the loop section that was reading the digital pins. I still did not

understand the exact cause, but to help isolate the cause, I commented out the code that was reading pin 0 in the loop. When I did this, I could fully control the LED, both on and off through Scratch. So somehow reading pin 0 shortly after setting pin 0 high would turn it off. It took a little digging and reading some hardware documentation, but the reason for the bug was, that even though as I was setting the pin to be an output pin, when the read of the pin came along, unbeknownst to me, it changed the pin mode back to an input pin which turned off the LED. A pin cannot be both input and output at the same time.

So what was the fix? I added some logic that when a pin was set as an output pin, an entry was marked in a table to prevent that pin from being scanned for input data. The fix was easy, the determination of the problem was complicated.

The lesson learned here is that sometimes, by disabling code by (just add a leading # character to “comment out” the code), can help to isolate the cause of the problem.

How To Determine If A Section Or Line Of Code Was Executed

Sometimes, it is helpful to know if a section of your code is being executed. For example, you may wish to know which branch of an if/else statement is being executed. A common technique to accomplish this is to add a print statement to each branch. This technique works well when you are debugging on your desktop

computer, but the micro:bit does not have a “print” statement to use. To get around this limitation, instead of printing, you can display a different image on the micro:bit for each branch. This is an easy workaround to help you get some useful clues as to how your code is operating.

The Take-Away

Now that you understand some of the types of bugs you may encounter in your programming adventures, and how to deal with them, let’s summarize what you have learned.

1. Always check for syntax errors using a tool like the mu editor before flashing code to the micro:bit.
2. When a syntax error is found, fix each one from the first to the last, one by one, and then rechecking as you go along.
3. For run-time errors, use the REPL to display any possible exceptions thrown by Python.
4. For design type run-time errors, try to isolate the area of code where the error might be occurring.
 1. Use the display to identify if a line of code is being executed.
 2. Comment out the code to see if a specific line of code is the cause of the issue.
5. If you are having problems figuring out how to fix a bug, avoid frustration:
 1. Take a break – go outside and play for a while. At the very least, step away from the computer.
 2. Ask a friend, parent or teacher for help. Sometimes we are so close to the problem, we can’t see its solution. Fresh eyes can help.
6. Remember, debugging is a skill that takes practice and determination to master. Each time you successfully squash a bug, your debugging skills get stronger. Stay on the path, and you will become a Debugging Ninja.



Help us cover the costs of micro:mag

A close-up photograph of a black Micro:bit board. A small square component is visible on the board, with a blue triangle and the letter 'A' printed below it. A yellow ribbon cable with two circular loops is connected to the board. The loops are labeled with the numbers '0' and '1'.

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.

A close-up photograph of a black Micro:bit board. A yellow ribbon cable with two circular loops is connected to the board. The loops are labeled with the numbers '2', '3V', and 'GND'.

Donate on

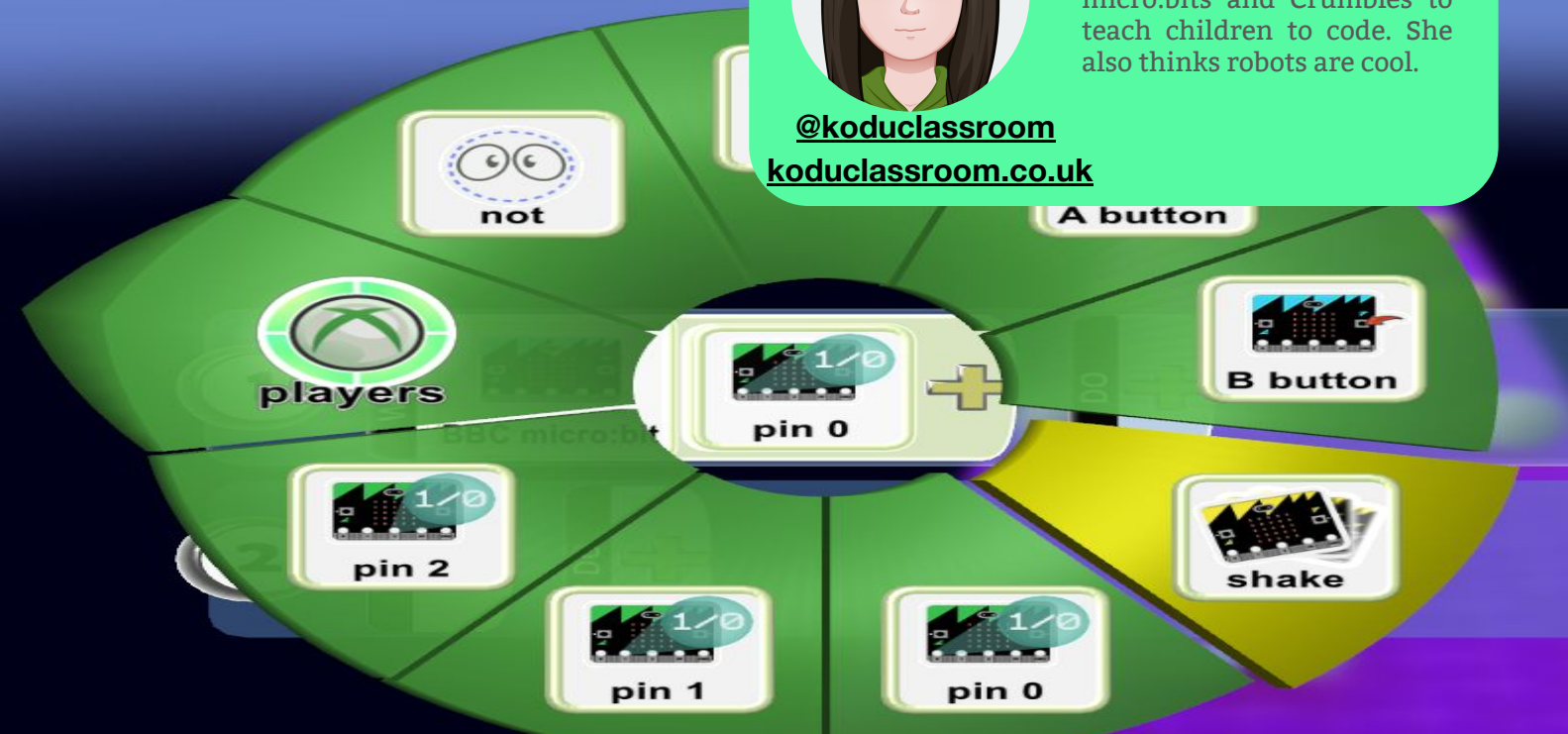
 **PayPal:**

micromag.cc/donate



Siobhán loves using Kodu, micro:bits and Crumbles to teach children to code. She also thinks robots are cool.

[@koduclassroom](https://twitter.com/koduclassroom)
koduclassroom.co.uk



Use your Micro:bit with Kodu Game Lab to transform your Micro:bit into a game controller

Using the micro:bit as a game controller with Microsoft Kodu Game Lab

How to program the computer games you build in Kodu to use a micro:bit as a game controller.

You Will Need:

- A micro:bit and USB cable
- Kodu Game Lab installed on your PC (go.micromag.cc/kodudwnld)

In this tutorial, you will learn to program your Kodu games to respond to the micro:bit being used as a game controller.

You will program the micro:bit to display outputs as well as use its accelerometer to control the player in your game. This tutorial assumes that you already have some experience of building games in Kodu.

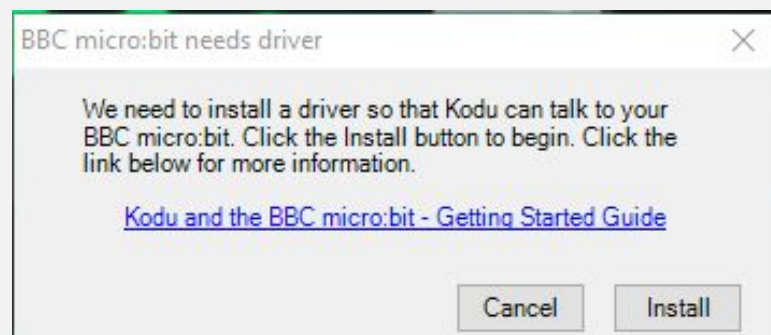
A bit about Kodu and the micro:bit

If you haven't come across Kodu Game Lab before, it is a great piece of software from Microsoft and even better – it's free! In Kodu,

you can build your own 3D computer games on your PC. It uses a visual programming language involving writing lines of 'code'. I have taught pupils from 7 upwards to use this software with a micro:bit and it is amazing how quickly they get the hang of the software. The 'code' is like a sentence and as a result, it has to make sense in order to work! The code is written in the form of When... Do... so an action is triggered to take place when an event occurs.

Setting up your micro:bit to work with Kodu

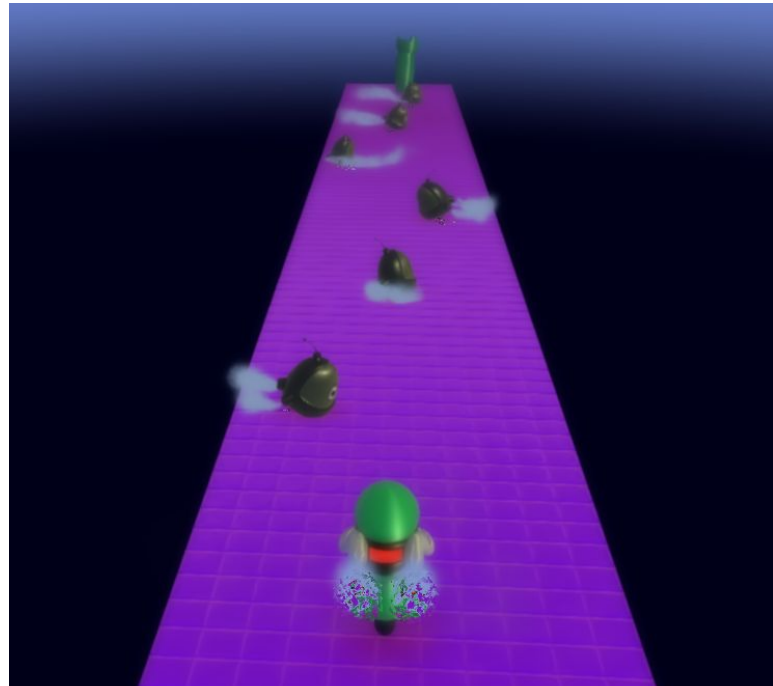
When you attach your micro:bit to your computer using the USB cable, you should be prompted to install an additional driver. If you are planning to do this using networked computers, you may need to ask your administrator to do this for you. Once the driver is installed, Kodu is ready to use the micro:bit as a controller.



This dialog box will appear when you attach your Micro:bit prompting you to install the driver

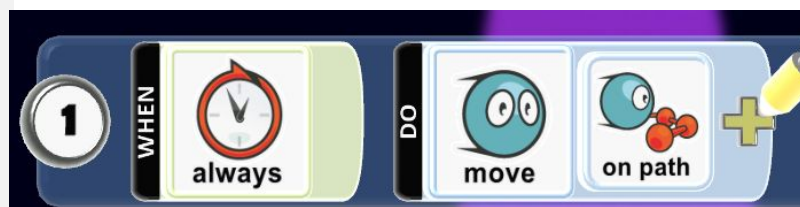
If your having trouble using Kodu, Siobhán has written some tutorials over on her website. These should help you get used to the Kodu interface: go.micromag.cc/kodubasic

Build a game



Build a simple game in Kodu with a clear objective and some obstacles for your character to avoid.

First, you will need to build a basic game in Kodu. I made this simple game where the player is the green cycle and the objective is to reach the castle at the end of the purple path. All of the black Kodu characters have been programmed to always move on a path. This means they will act as an obstacle for my green cycle to overcome.

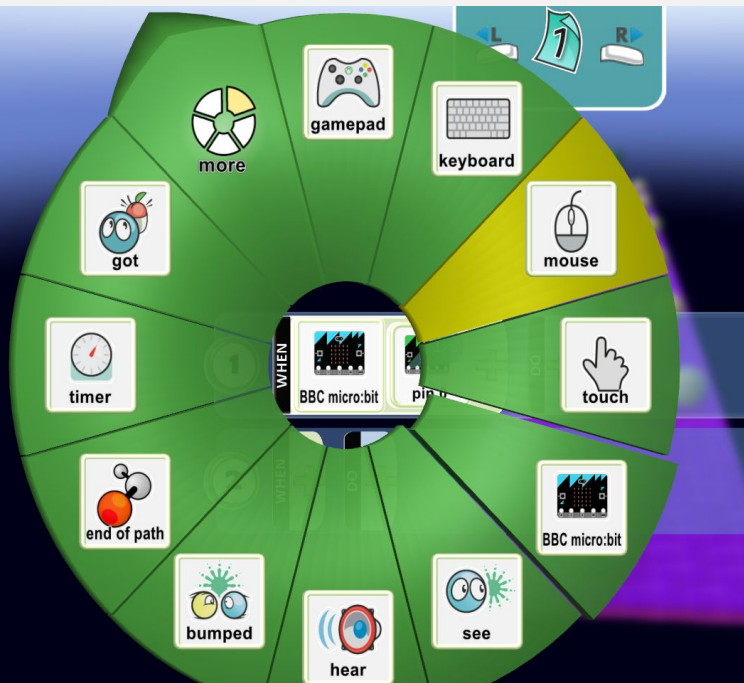


Program your enemies to move by themselves on a path.

You can build a much more exciting game using many more objects and characters so have a great time exploring the software and creating amazing worlds.

Programming the cycle to respond to the micro:bit

First, you will want your player to move when you move the micro:bit. This will use the accelerometer so that it can sense which direction you want your character to move in. When you go to program your cycle, you should see that the micro:bit image now appears. If it doesn't, it shows that the driver hasn't been installed.



If you can see the Micro:bit symbol, your driver has installed correctly

see that the micro:bit image now appears. If it doesn't, it shows that the driver hasn't been installed.

Once you click on the micro:bit symbol, you should see that lots of options for possible micro:bit inputs appear:



Some of the ways you can use the micro:bits inputs to control your player in the game

To program the cycle to move when you move the micro:bit, here is the code that you need:



Program your cycle to respond to the movement of the micro:bit.

Now if you press the Escape key twice or press the play mode, you can enter 'game mode' rather than the editing mode you have been using to create your game. Sometimes, when you go to do this, the micro:bit may start flashing on the back but that just means that it is transferring your program to the micro:bit. Now you can test your game.

Once you have programmed the cycle to move using the tilt function, you can also program it to respond to other inputs, including the buttons, the micro:bit recognising that it is being shaken, or the pins being used. You can also take

advantage of the micro:bits output functions by programming it to display an image or a scrolling message on the LED display. To program the cycle to move when you move the micro:bit, here is the code you need:

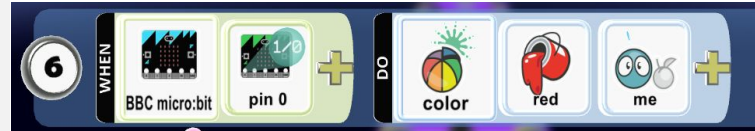


A basic program to control the cycle so that it responds to the micro:bits inputs.

Programming the cycle with this code means that it does the following:

1. WHEN the Micro:bit is tilted, the cycle will move
2. WHEN the Micro:bit is shaken, the cycle will jump
3. WHEN the B button is pressed, the cycle will shoot multi-coloured blips that will cause the target to explode.
4. WHEN the A button is pressed, the Micro:bit displays a helpful message to the player: 'go to the castle'
5. WHEN the cycle bumps a black Kodu, the Micro:bit displays 'oh no!' on its LED display

Become a human circuit!



Become a human circuit by using the input/output pins on the micro:bit

Another way of using the micro:bit is by programming something to happen when you hold one of the input/output pins at the same time as holding the ground pin (GND). In the code shown here, if you hold Pin 0 at the same time as holding the GND pin, your cycle will change to red instead of green.

Now that you know the basics, it's over to you to create interactive Kodu games using the micro:bits input and output capabilities.

Let's take the project even further, see what you can do with the information you've learnt.

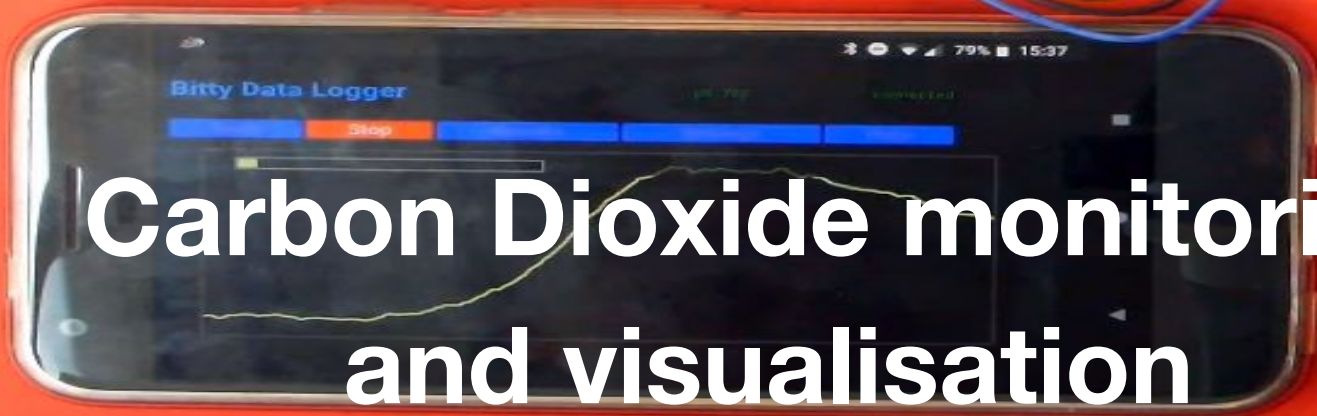
Taking the project further:

There are all sorts of ways that you can use your micro:bit to interact with Kodu. This tutorial from the Kodu Team (go.micromag.cc/koduadvanced) even shows you how to add an LDR (Light Dependant Resistor) to control the light level in your game world!



[@bittysoftware](#)
bittysoftware.com

Martin loves 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.



Carbon Dioxide monitoring and visualisation

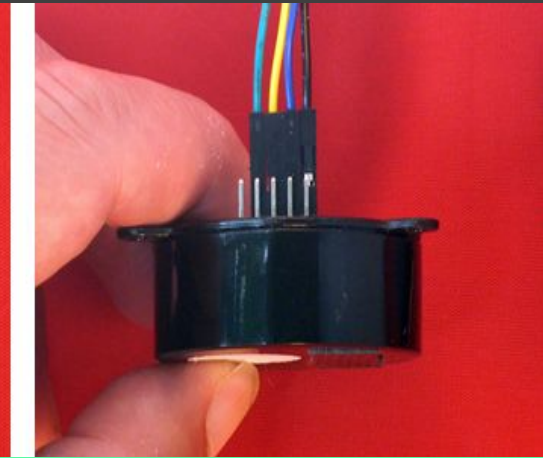
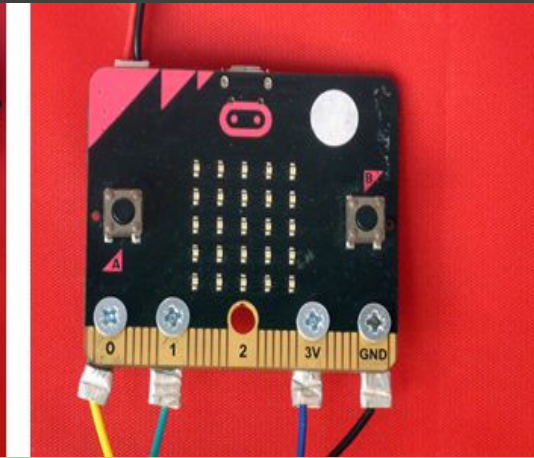
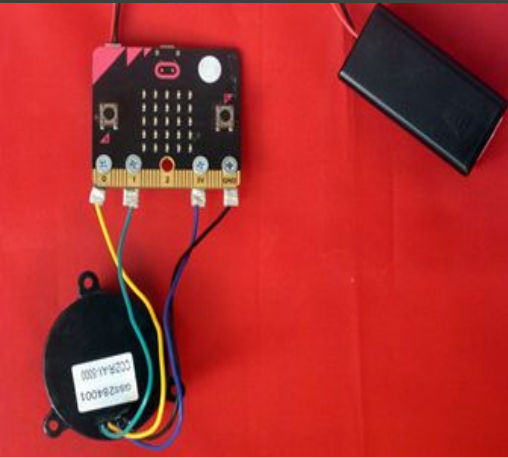
Bitty Data Logger charting CO2 readings from the sensor connected to the micro:bit in real-time.

Learn how to use a BBC micro:bit, a sensor and smartphone app to monitor and visualise carbon dioxide (CO₂) levels in the environment.

You Will Need:

- 1 x BBC micro:bit
- 1 x CozIR®-A sensor from GSS – see go.micromag.cc/buysensor
- 1 x Android or iOS smartphone or tablet with Bluetooth 4.0 (minimum)
- The Bitty Data Logger application from Bitty Software running on the smartphone see go.micromag.cc/bittydatalogger

CO₂ monitoring has all sorts of important, real-world applications. It's used in food production, healthcare and even in places like space stations! A micro:bit can be connected to a CO₂ sensor and with the right code, collect CO₂ levels and report them over Bluetooth to a smartphone app for visualisation. In this tutorial, I'll describe the general approach involved, highlighting how the components talk to each other. If you'd like to have a go at



How to connect the CozIR®-A sensor to a micro:bit

making this project, full details, including source code can be found at go.micromag.cc/fullCO2project

About Bitty Data Logger

Bitty Data Logger is an application for iOS and Android devices which can capture, chart and share data communicated from a micro:bit using Bluetooth. The data could be from an internal sensor like the micro:bits accelerometer or from an external device connected to the edge connector like a capacitor or a CO2 sensor. Bitty Data Logger makes the invisible visible!

Connecting the CO2 sensor

The CozIR®-A sensor has a number of pins on its underside. The important ones are the GND, 3.3V power, receive (RX) and transmit (TX) pins. The TX and RX pins allow data to be transferred, one bit at a time between the micro:bit and the sensor. Communication which takes place one binary bit at a time is called serial communications.

To connect the CozIR®-A to a micro:bit, you make connections like this:

CozIR	Micro:bit
GND	GND
3V3	3V
TX	RX (pin 1)
RX	TX (pin 0)

Communicating with the CozIR®-A Sensor

The GSS CozIR®-A sensor has a set of commands which a device like our micro:bit can use to talk to it. Talking involves sending commands to the sensor's RX pin and listening for the result involves receiving data from the sensor's TX pin.

For example, sending the command `G\r\n` to the sensor tells the sensor to calibrate itself for more accurate readings. `\r\n` represent the special characters carriage return and line feed which all the CozIR®-A commands must end with.

Communicating with Bitty Data Logger

Bitty Data Logger uses the micro:bit's event system. The event system is what makes it possible for things happening (i.e. events) like a

button being pressed, to trigger code you've written. The event system lets different parts of the micro:bit system communicate with each other. You can think of this as being like someone shouting whenever something of interest happens:

Hey! Someone just pressed Button A!

One of the nice things about the event system is that software components that generate or respond to events do not have to be inside the micro:bit! They can be connected to the micro:bit over Bluetooth using something called the Event Service. All MakeCode applications which use the Bluetooth package, automatically have the event service built into them, meaning that events can be used for bidirectional communication between the micro:bit and the other device, connected over Bluetooth.

Various event types are used by Bitty Data Logger. These are the ones which are used in communicating CozIR®-A sensor data:

Event ID	Event Name	Direction of Communication	Purpose
9020	Pin Selection	bitty data logger to micro:bit	Let's bitty data logger tell the micro:bit which pins on its edge connector to read data from before transmitting it over Bluetooth.
9030	Data	micro:bit to bitty data logger	Each 9030 event has a value which combines a pin number with a value. This is how up to three different types/sources of data from an external device, connected to the micro:bit, can be communicated to bitty data logger.

See go.micromag.cc/binarynums

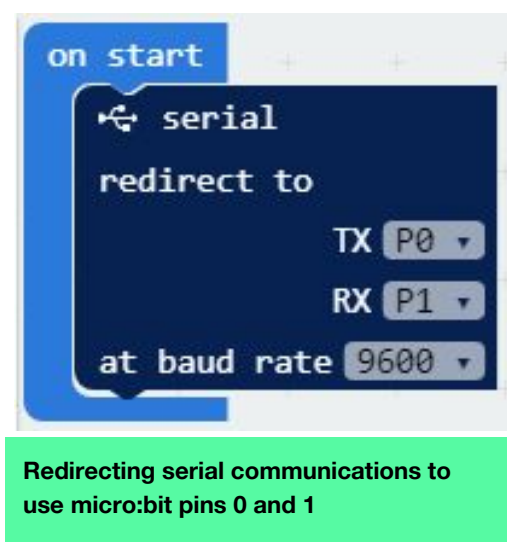
to learn more about binary numbers and bits!

Code

We can think of the code needed as falling into two parts; we need some code that will send commands to the sensor and which will receive data back using *serial communications*. We also need code which will put the CO2 data into the right format for Bitty Data Logger and send it over Bluetooth as an *event*.

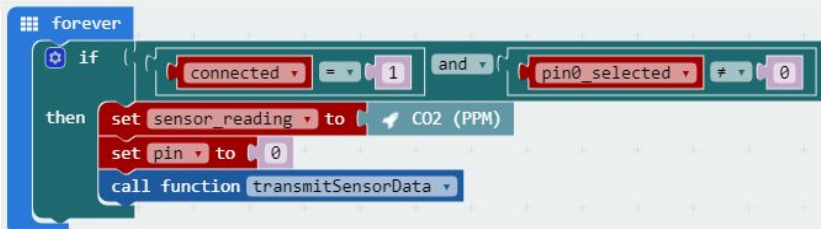
1) Obtaining Data From the Sensor

Usually, if we're performing serial communications with a micro:bit, data passes into and out of the micro:bit via the USB socket. In our case, we want to read and write to the TX and RX pins that connect the sensor to the micro:bit. This means we have to *redirect* serial communications to the micro:bit pins we've connected the sensor to. In our case, that will be pins 0 and 1. This MakeCode block will do what we need:



We then need to send the appropriate commands to the sensor. Luckily, we can avoid knowing all the details, because Simon Monk (go.micromag.cc/simonmonk) wrote some custom MakeCode blocks that take care of all the

nitty-gritty. Our code needs to loop inside the usual forever block, requesting CO2 readings and then sending the data over Bluetooth to Bitty Data Logger, like this:

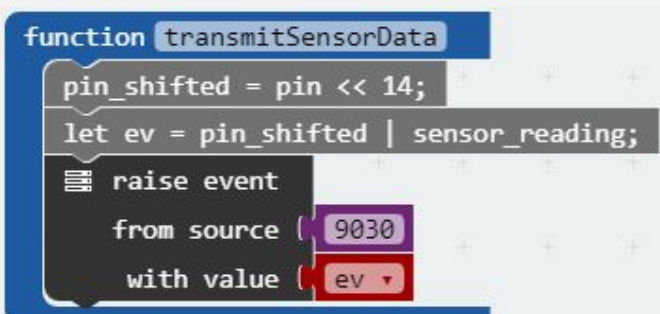


Repeatedly reading CO2 levels from the sensor and then transmitting the value over Bluetooth

The CO2 (PPM) block returns a CO2 level reading in *parts per million*.

2) Sending Data to Bitty Data Logger

After obtaining a CO2 reading, our code calls a *function* `transmitSensorData`. This function is where the data is put into the right format and sent over Bluetooth. This might not be obvious from the code but all that's involved is creating a micro:bit event with the right values and by magic, it will be sent to Bitty Data Logger!



Combining a pin number and sensor reading and creating an event which will be sent over Bluetooth.

The Full Project

Hopefully, this article has given you a sense of what's involved in using sensors and Bluetooth to capture and visualise data on a smartphone. There is, of course, a bit more involved than has been covered here. Find out more in the box on the right

Taking the project further:

Your next step should be to try the full project, which you will find at:

go.micromag.cc/fullCO2project

micro:mag

Contribute
to micro:mag

Have a cool project to share?
Write about it for the next
issue of micro:mag.
We'd love to hear about it!

Fill in the form at:

micromag.cc/contribute

Any questions? Email us at:
hello@micromag.cc

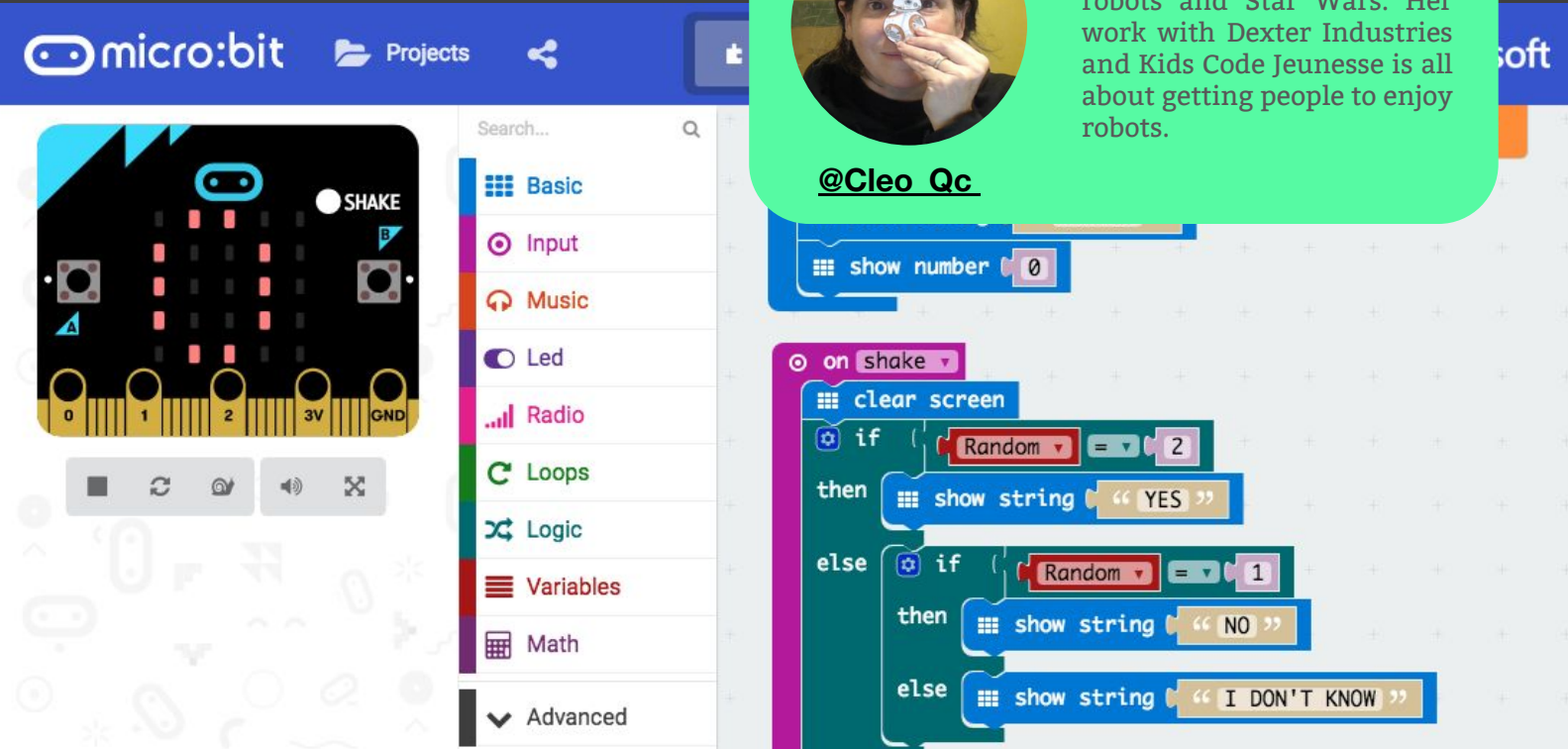
:make

Nicole Parrot



Nicole hails from Hoth also known as Canada. She enjoys robots and Star Wars. Her work with Dexter Industries and Kids Code Jeunesse is all about getting people to enjoy robots.

[@Cleo_Qc](#)



MakeCode is the main editor for people programming the micro:bit

Make your own blocks with Microsoft MakeCode

Makecode is a great environment which can be expanded upon when you know how to create your own blocks

You Will Need:

- A micro:bit and USB cable
- Makecode (makecode.microbit.org)

In this tutorial, you will learn how to create your own blocks using two different approaches. The first one takes a drag-and-drop approach using the Functions category, and the other one uses textual programming which will require a little bit of Javascript. Or a lot, depending on your

needs. We will be developing a library of facial emotions for the micro:bit in order to cover the concepts and keep the Javascript to a minimum.

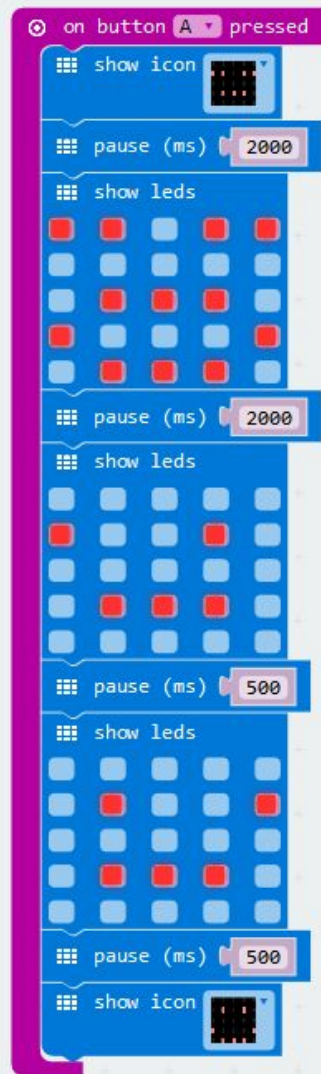
A bit about MakeCode

MakeCode offers a great environment for learning how to code with the micro:bit. It is simple enough for beginners to get started and flexible enough that you can add to it once you push your code a little further and suddenly feel

the need for special blocks. There are two simple methods to add your own blocks to the MakeCode editor:

1. The Functions category
2. The “custom.ts” file

Let’s create a wake-up animation. At first the micro:bit is asleep, then it yawns, checks left and right to see what’s happening, then it’s happy to see you. In the example here, it’s linked to Button A so we can test it at will but it would also make sense to have it run with the on start block. It’s a bit long and clumsy to keep in your code, especially if we want to use it in two (or more) different places! The code reference is on the left.



Create a Function

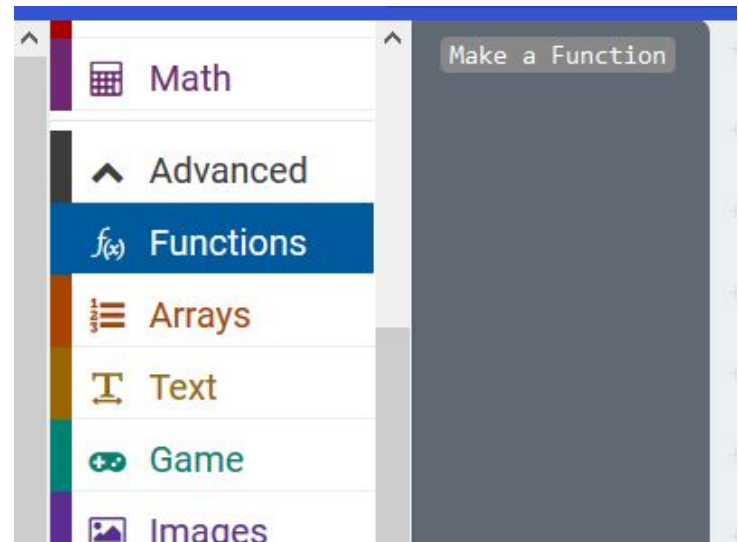
Creating a function is the easiest method of adding a block to MakeCode.

Textual programming is not required, only blocks. This is conceptually the same approach as creating a new block in Scratch. A couple of easy steps will get you into a place where you

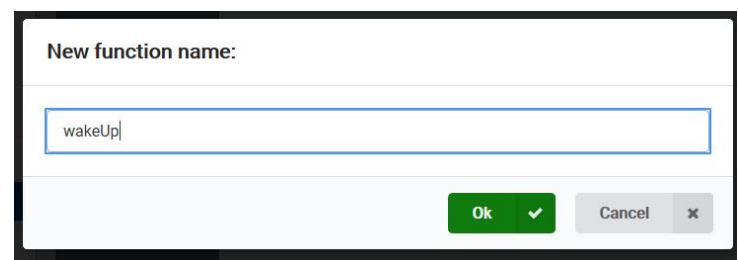
can start your function:

1. Click on Advanced
2. Click on Functions

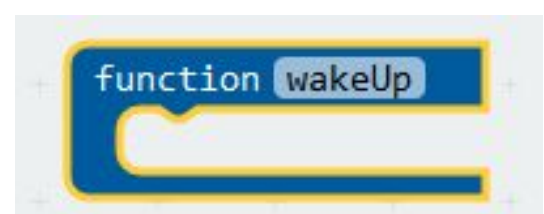
This opens up a flyout containing a single grey block:



Click on that single block, called “Make a Function”, you will get a dialogue window:

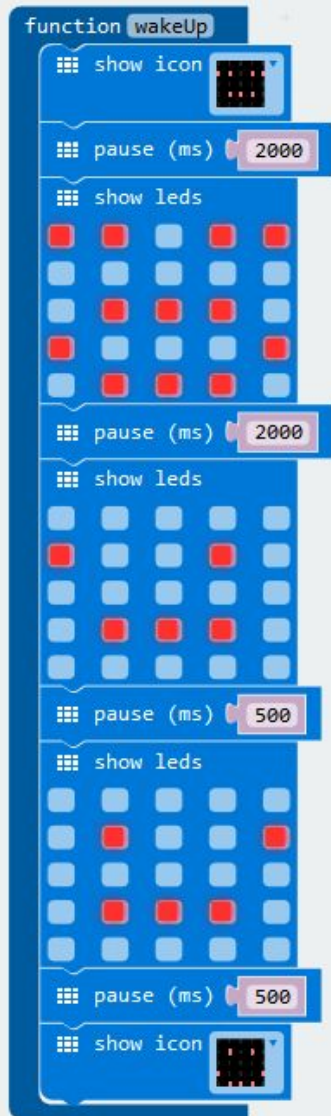


Enter the name you want for your function. Here, I called it “wakeUp”. Try to be as descriptive as possible, without getting too long a name. You can use spaces but, out of habit, I do not. Click on the Ok button once you’re happy with your name. A definition block has appeared in your programming editor:



Create Makecode Blocks **:make**

You can fill in this block with the animation we created at the start. Either drag the blocks you already did into this function block or, if you haven't created the animation yet, create it now.

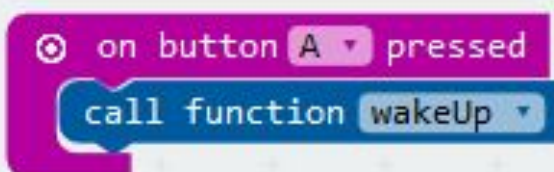
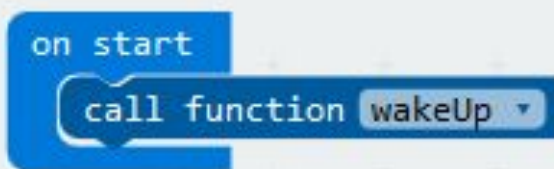


Once you think you have everything, it's time to try it out!

Go back to the "Advanced" section and click on "Functions" again.

Now the flyout has a new block, with the name you have chosen, your own very special block!

Drag it in the On Start event to test it out. I've also used it in the Button A block now that it no longer takes all that space!



This gives us code that is

- Much easier to understand than our long

block sequence!

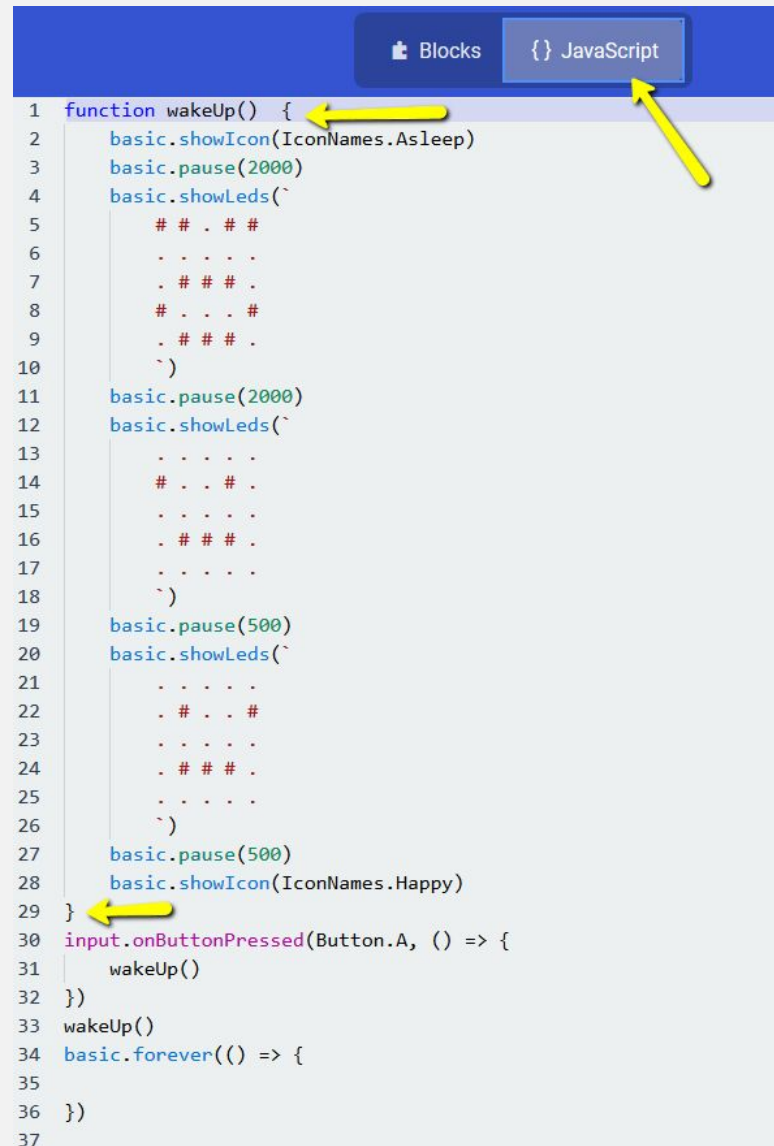
- Much easier to follow what's going on. "Wake up" is more descriptive of the intent of the animation so you'll remember what you meant to do even a few months down the road.
- Much easier to add extra code without getting lost in what we're trying to achieve.
- Much easier to use the same animation in more than one place. Maybe you want your Micro:bit to also wake up when shaken.
- Much easier to keep your programming area clean. You can move the function defining block outside of the visible area (lower down or to the right) so your application stays uncluttered.

Second approach: the "customs.ts" file

This second approach will let us hide the details of our function code even more. (less clutter is always good). But also, it will let us re-use code easily from one project to the next. However, it comes at a price, we need to go to textual programming, at least a little bit.

Let's grab our current code first. With the above project still on your screen, switch to "Javascript" and highlight the textual code for our function. Start from the first curly bracket up to the corresponding end curly bracket (which will be on a line all by itself). Do not include the curly brackets themselves (the { and } characters).

They are indicated by arrows in the following image:

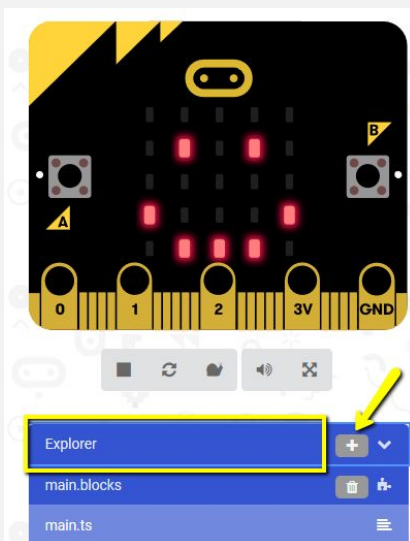


```

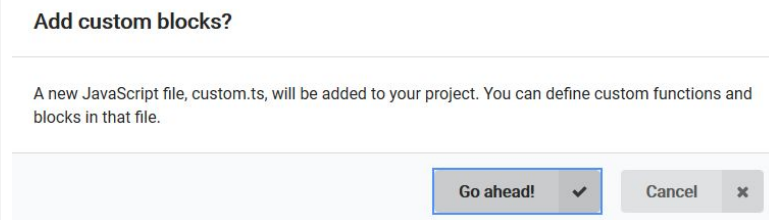
1 function wakeUp() {
2   basic.showIcon(IconNames.Asleep)
3   basic.pause(2000)
4   basic.showLeds(`
5     . . . . .
6     . . . . .
7     . # # # .
8     # . . . #
9     . # # # .
10    `)
11  basic.pause(2000)
12  basic.showLeds(`
13    . . . . .
14    # . . # .
15    . . . . .
16    . # # # .
17    . . . . .
18    `)
19  basic.pause(500)
20  basic.showLeds(`
21    . . . . .
22    . # . . #
23    . . . . .
24    . # # # .
25    . . . . .
26    `)
27  basic.pause(500)
28  basic.showIcon(IconNames.Happy)
29 }
30 input.onButtonPressed(Button.A, () => {
31   wakeUp()
32 })
33 wakeUp()
34 basic.forever(() => {
35
36 })
37

```

Click on “Explorer”, and the + sign next to it, as indicated here (make sure you’re still in Javascript mode):



Accept the dialogue that pops up:



A new file containing Typescript code has just been created for you. It’s called the “custom.ts” file and we will be adding our code in there. You can also notice too that a new category has been created! It’s called “Custom” and shows up between the “Basic” category and the “Input” Category.

In the “custom.ts” file, there’s a lot of code that we won’t need but it doesn’t hurt to keep it for now. Look for “// Add code here” on line 27. You will be pasting your code right after this line, before the curly bracket } on the following line.

Let’s change the name of the default function from foo to wakeUp (on line 26, just before the // Add code here line. We can remove what’s in the parentheses too as our code is simple.

```
export function wakeUp(): void {
```

And you’re done! This is it, you’ve done it!

If you go back to Blocks mode, you will see your new block under the Custom category! You can get rid of the cumbersome function definition we created earlier. We no longer need it, and it’s taking room for nothing.

Attention:

Your `custom.ts` file is part of your current project but you may want to keep a copy of it for re-use in other projects, or sharing with a friend, or simply as a backup, should something happen to your project. Your best approach is to copy/paste it into a file on your computer and keep it safe. You can also share it with your friends, as they can now copy/paste it into their own `custom.ts` file.

What Next?

But what about the fib value block?

There's an extra block in the custom category. You can get rid of it if you want.

1. Go back to Javascript
2. Click on Explorer
3. Click on `custom.js`
4. Scroll to the end of the file
5. You do need to keep the very last line, which has a closing curly bracket: `}`. Make sure you leave that one in.
6. You can delete the three lines above it, that start with `export function fib` (remember that `fib` was the name of the block we want to get rid of!)

When you go back to Blocks mode, there will only be one block now in the Custom category.

I hate green

Then you're in luck as you can change the colour of the custom category.

Look for a line that reads

```
namespace custom {
```

The line just above it has `colour =#0fbc11` where

`#0fbc11` is the green colour. You can replace it with any colour you like based on its colour code. See <https://html-color.codes/> for ideas. You must use the American spelling for colour, and keep the `#` sign in. You can also change the name if you edit the word "custom" for "animations" for example although this will break any program you may have on your screen. (you can simply swap out the blocks).

How do I create more blocks?

You can create as many blocks as you want.

Just after the line that says "namespace custom {" (on line 18) you can add this code:

```
///  
export function myownblock(): void {  
  // add code here  
}
```

How can I share my blocks?

Here's a simple way of sharing your custom blocks. Empty any blocks you may have in the makecode editor but do keep your `custom.ts` file! Click the Share button and copy the link it's giving you. Give that link to your friends.

Is that an extension? ?

You've done a lot of the work required to create an extension! However, for a complete extension you need GitHub. In the meantime, you've already gained a lot of knowledge! Now you have enough knowledge to handle long programs in Makecode, by hiding some complexity behind your own blocks. The sky is the limit!

:make

Jody Carter



@codeyjody

Jody is a primary school teacher and computing leader for GLF Schools, CAS Master Teacher, Raspberry Pi certified educator, coding enthusiast; trying to make computing enjoyable and engaging.



No matter what the weather, always keep up to date with the conditions with this micro:bit weather station.

Micro:melt - just exactly how hot is it!?!

How to make a weather station in your garden with two micro:bits using the radio functionality.

You Will Need:

- 2 x micro:bits
- Battery pack with JTS connection
- Micro USB cable
- Sparkfun weather:bit, <https://www.sparkfun.com/products/14214>

Now that summer's officially here it's time to dig out the flip-flops, hanky on the head and get the factor 50 on. Once you've done that why not make a nifty little weather station relay - that

way you won't have to stand in the boiling sunshine to find out how hot it is!

First, connect your micro:bit to the weather:bit, this is the Weather Station. The other micro:bit will be the Receiver.

Next, go to microbit.org and use the Javascript Blocks editor; we're going to work on the Weather Station first.

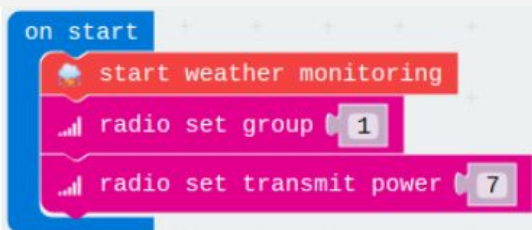
Step 1: Add Weather:bit package

Click on Add Package and type in “Weather bit” to search. Click on the weather:bit and it’s added to the main menu. You will find Weatherbit at the bottom of the main menu.

Step 2: Add the weather blocks

The Weather Station will be activated when it receives the activation message (in the form of a string sent to it). It will take 3 readings: temperature, air pressure and humidity, and send these back to the receiver using the micro:bit’s radio feature.

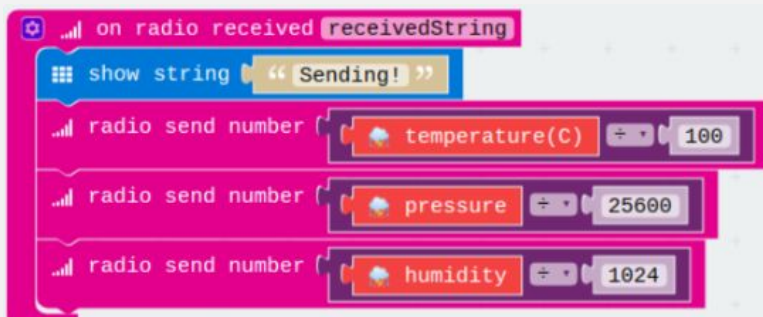
Let’s have a look at the code...



Start the weather station monitoring the weather.

First, tell the weather:bit to start monitoring the weather. This is followed by setting up the radio. The same group must be set by both sending and receiving micro:bits. Secondly, we want maximum power so that the weather station can be read over long distances.

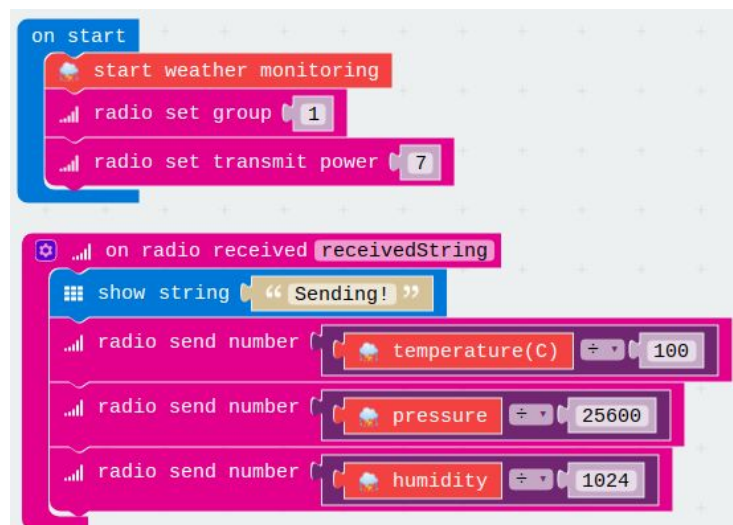
Step 3: Add radio blocks



Read the measurements and send back.

When the receiver sends the activation message the Weather Station will display a message then send the three data readings. The readings are sent as numbers so make sure you use the radio send number blocks. Note that the raw readings on the weather:bit need to be converted into something more meaningful.

Your Weather Station Code Blocks should look like this:



The complete code for the Weather Station.

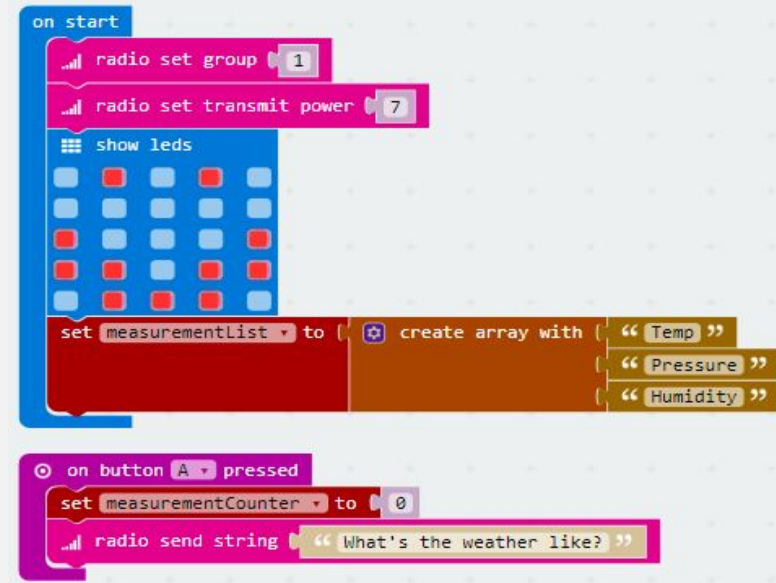
Save your project then drag the .hex file onto the micro:bit icon in your file system. Now start a new project and name it Receiver.

Step 4: Create Receiver project

The Receiver has 2 jobs: send an activation message to the Weather Station to tell it to send the data across and to display the data it receives in return.

To begin with, set the radio group and strength to the same as the Weather Station. This ensures that they can talk to each other. Also, show a little smiley face to say that everything’s good.

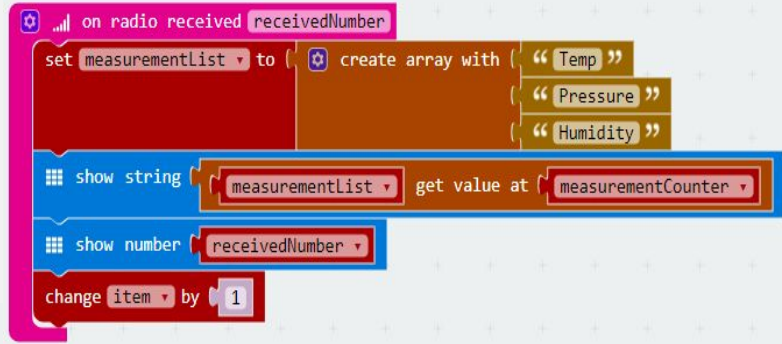
The next block is where to send the activation code. This just needs to be a string so could say anything at this point. The set measurementCounter to 0 is a counter we use in the next code block which we will look at now.



Send the activation message to the Weather Station.

In the same way that we used on radio received on the Weather Station to start reading the data, it's used here to display the data once it's been received. Let's look at them in more detail.

When the Weather Station sends through the data it makes 3 separate calls, temperature, humidity and pressure. The data sent through will always be in that order so set a counter - measurementCounter - so we know how many readings have been sent. An array with the 3 reading names stored in it is looped through and kept track of using measurementCounter. This makes sure that temp is displayed for the first number received, pressure for the second and humidity for the third. If another request is sent through to the weather station button A is pressed and sets the counter back to 0.



Upon receiving the measurements display them to the user.

Save the project, download the .hex file and drag it on to the Receiver micro:bit. You should see a smiley face once completed. Make sure both micro:bits are powered up and press button A on Receiver. The Weather Station should say 'Sending!' and then the Receiver will scroll through the 3 different measurements.

micro:mag

ADVERTISE
with micro:mag

If you make products for the micro:bit. micro:mag is the perfect place to get noticed!

For more details, email us at:
hello@micromag.cc

:make

Sam Watson



My name is Sam and I am 10 years old. I live in Yorkshire and love to code! I wrote some code that was run on the ISS for the Astro Pi challenge.

The 4tronix Bit:Bot And Bit:Commander at 'Formula One in Schools'

The Bit:Bots on race day as our pit stop entertainment! Each one was controlled with a Bit:Commander.

Pit Stop entertainment!

In this tutorial you will learn how I coded the Bit:Bot and Bit:Commander to race two Bit:Bots.

You Will Need:

- micro:bit
- 4tronix Bit:Bot
- 4tronix Bit:Commander
- Batteries
- Micro USB cable
- Access to the python HTML Editor

Learn how I coded the 4tronix Bit:Commander as a controller for the 4tronix Bit:Bot for a race. It was part of the Formula One in schools

competition. I used Python to code the Bit:Bots and Bit:Commanders after having been asked if I could do something with a Bit:Bot during the pit stop.

Introduction

Each class in Year 5 were asked to design and make a 3D printed Formula One car. Using SolidWorks each team designed a car and gave it a name. The designs were then 3D printed and

painted, axles were fitted and wheels added.



The 3D printed F1 car 'The Dark Star' designed using SolidWorks.

We also had to plan merchandise, entertainment for a pit stop, and create a portfolio as part of the competition.



Flags and bookmarks with our 'Dark Star' logo on to promote our team, and the stickers that our guests and visitors were given.

One of the team members thought that I could code Bit:bots for our pit stop. I agreed and started to work on the code during each team meeting. Before long I suggested using the Bit:Commander as a controller instead of the micro:bit alone. I then began to work on the code at home as well as in the meetings at school.



One of the posters to promote The Dark Star.

The Bit:Bot

First of all, I had to programme the micro:bit for the Bit:Bot. Here is the code I used:

```
#Bit:Bot Code

from microbit import *
import radio

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

def Drive(lft,rgt):
    pin8.write_digital(0)
    pin12.write_digital(0)
    if lft<0:
        pin8.write_digital(1)
        lft = 1023 + lft
    if rgt<0:
```

```

rgt = 1023 + rgt
pin12.write_digital(1)
pin0.write_analog(lft)
pin1.write_analog(rgt)

```

```

while True:
    s = radio.receive()
    if s is not None:
        if s=="N":
            Drive(800,800)
        elif s=="S":
            Drive(-800,-800)
        elif s=="NE":
            Drive(800,200)
        elif s=="NW":
            Drive(200,800)
        elif s=="SE":
            Drive(-800,-200)
        elif s=="SW":
            Drive(-200,-800)
    else:
        Drive(0,0)
    sleep(20)

```

The Bit:Commander

Next, I had to code a micro:bit for the bit:commander to send radio signals to the bit:bot. Eg. North, South, North East, North West, South East, South West. Here is the code I used:

```

# Bit:Bot controller code.

from microbit import *
import radio

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

while True:
    a = pin12.read_digital()
    b = pin14.read_digital()
    dx = pin1.read_analog()

```

```

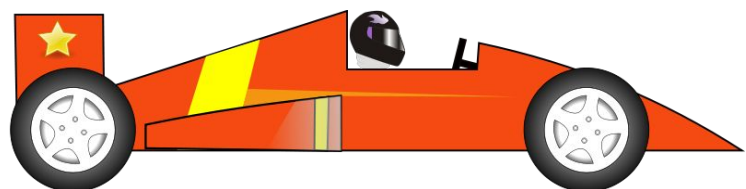
if a and dx<150:
    # forwards left
    display.show(Image.ARROW_NW)
    radio.send("NW")
elif a and dx>850:
    # forwards right
    display.show(Image.ARROW_NE)
    radio.send("NE")

elif b and dx<150:
    # backwards left
    display.show(Image.ARROW_SW)
    radio.send("SW")
elif b and dx>850:
    # backwards right
    display.show(Image.ARROW_SE)
    radio.send("SE")
elif b:
    #backwards
    display.show(Image.ARROW_S)
    radio.send("S")
elif a:
    # forwards
    display.show(Image.ARROW_N)
    radio.send("N")
    sleep(20)

```

Preparation

The day before race day the team manager and I realised we needed a track to race the Bit:Bots on. We started to make it from some cardboard boxes. Unfortunately, we ran out of time in school! So I brought all the pieces of track home to finish making it. I spray painted the track black, put it all together and added masking tape road markings to it.





Making the track from cardboard boxes! Sprayed and put together at home late the night before the race!

I had a last-minute hitch on the eve of the race when I discovered we were a Bit:Commander short! I then got in touch with a friend who could lend me one!

Race Day

On the day of the competition I went into school early with the track, Bit:Bots and Bit:Commanders and along with the rest of the team we prepared our track, merchandise, portfolio and the Bit:Bots and Bit:Commanders.

As soon as people were coming into the hall they were spotting our pit stop and were eager to give it a go. Judges and visitors were all impressed and were curious to find out more about the

Bit:Bots and Bit:Commanders! People had lots of questions about the Bit:Bots such as: 'Are these Bit:Bots powered by BBC Micro:Bits?' and 'How long did it take to code these Bit:Bots?' I explained how I had programmed them and that I had done lots of work on it at home too. Our 3D printed F1 car didn't win the time trials on the day but we had lots of fun!



The 'Formula One in Schools' track used for the time trials for the 3D printed F1 cars.

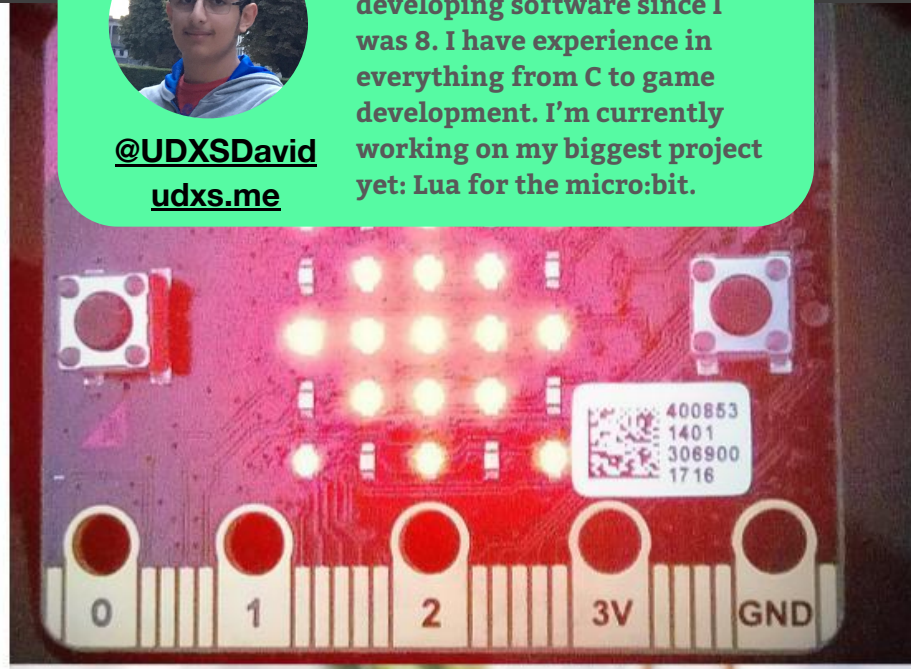
:make

Davit Markarian



@UDXSDavid
udxs.me

Davit is a 15-year-old Armenian teen that has been developing software since I was 8. I have experience in everything from C to game development. I'm currently working on my biggest project yet: Lua for the micro:bit.



Micro:mate: Your Very Own Smart Display

Who needs a \$200 Smart Display when you have a micro:bit, Pi, and phone already lying around.

You Will Need:

- micro:bit
- Raspberry Pi 3 or Zero W
- A Google Account
- A Phone with the Google Assistant

In the search for a summer project, I built a cool “Smart Display” that can show you the weather, stocks, messages, date, and time. What makes it cool? Well, it’s powered by the Google Assistant, meaning you can ask any Google Assistant compatible device (i.e a phone, Wear OS smartwatch, or Google Home) what to show

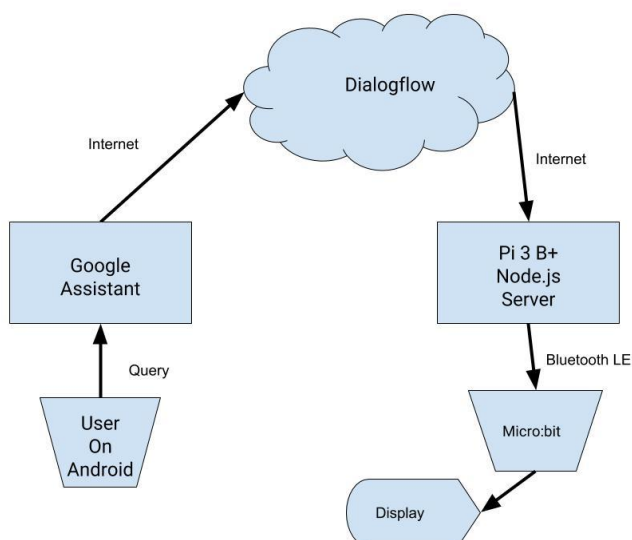
on the display. Use it to remind you to do something, show you the current weather, or keep up with stocks, if that’s your cup of tea. The best part? It’s very low cost of only \$25. go.micromag.cc/installNodeJs

What I used to build it?

Micro:mate is built with Node.js, which lets you use JavaScript, the language that powers the web, to write console applications, mainly servers. It just so happens that Micro:mate isn’t just one server, but two. One side interfaces with Google and the other side talks to the micro:bit through Bluetooth LE. I picked Node not only

because I'm familiar with JavaScript but because of NPM, the Node Package Manager. NPM is known for having a package to accomplish anything. NPM has everything we need to not only talk to the micro:bit, but also to run a web server that can communicate with Google as well as retrieve information like stocks (From Yahoo Stocks) and weather (From OpenWeatherMap).

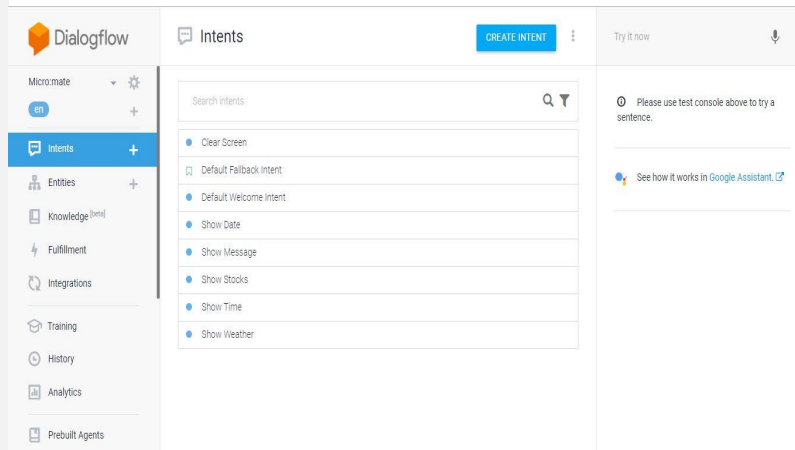
The other half of this project happens not on the Pi, not on the micro:bit, but on Google's cloud. Specifically, we use Dialogflow, a service from Google that lets you build AI agents that can take queries, such as "Show me the weather" or "Show me the stocks for MSFT", and process them, extracting parameters, such as "MSFT" or "Brussels", depending on the query. Because it's a Google product, it's super simple to connect it to the Google Assistant. On top of that, they also have Cortana and Alexa support which requires a bit more effort.



My plan for how everything would connect and interface.

What can you ask it?

I wanted the display to show information that is continually useful. I ended up adding Time, Date, Reminder messages, Weather (which I'm most proud of because of its fancy icons), and Stocks. On top of that, you can also ask it to clear the screen. You can browse the Dialogflow intents to see what you can say.



The Dialogflow console is where agents are made and configured.

How do you set it up?

First, Go ahead and download the files from GitHub go.micromag.cc/setup. Here, you can find the code, the micro:bit program, as well as detailed instructions. In an attempt to keep this tutorial short and sweet, the code and detailed instructions themselves are actually in the README. Here, you can take a look at the general list of steps:

1. On the Pi, Install Node
 - a. Install NVM
 - b. Install Node 8 through NVM
 - c. Update NPM
 - d. Enable sudo usage for Node/NPM
2. Install Prerequisites for using Bluetooth LE

Micro:mate - Smart Display :make

3. Install all the required packages in the micromate folder
4. Set up Dataplicity
 - a. Install it on the Pi
 - b. Set up tmux
 - c. Set up the Wormhole
5. Put the hex program on the micro:bit
6. Set up the Dialogflow agent
 - a. Import the zip file of the agent
 - b. Set the fulfillment URL
 - c. Configure Google Assistant Integration
7. Set up the Assistant Action
 - a. Configure the name and details
 - b. Deploy it as an alpha and open the link on your phone
8. Run the program
9. Ask the assistant whatever you want!
Look through the Dialogflow intents to see what you can do.

That's it! Now, you can play around and extend it as you wish.

Please check out my website udxs.me



micro:mag

Reviews in micro:mag

If you make cool add-ons for the micro:bit, why not get it reviewed in micro:mag?

This is a great opportunity to get your product noticed amongst our thousands of readers.

To get in touch, email us at: hello@micromag.cc

```

dataplicity®
pi@udxs-pi-3bp:~/Documents/micromate $ sudo node micromate.js
Looking for Micro:bit...
Discovered a Micro:bit. Connecting...
Connected. Starting Server...
Server started on port 80.
Test message sent.
Request recieved for intent: Show Weather
Query: show the weather
Request recieved for intent: Show Weather
Query: show the weather
Request recieved for intent: Show Time
Query: show the time
Request recieved for intent: Clear Screen
Query: clear the screen
  
```

What the server shows on the Pi when we get a request.

Chris Penn



Chris is a CS teacher from Warwickshire, everything that I code either involves Minecraft or something childish ☺

@ChrisPenn84

The micro:bit Express

Take your Lego train and control it with Python code. (Old and newer motors)

You Will Need:

- micro:bit compatible motor controller by Kitronik: To find out more: ([link](#))
To buy the controller: ([link](#))
- I recycled the battery pack from the Cam Jam Edukit 3 buy here [link](#) any generic 4.5 volt battery pack will be fine.
- A Lego motor / Lego Train Motor

In this tutorial, we will be going through step by step how to create a Remote-controlled Lego train using 2 micro:bits and MicroPython. There are two separate programs the remote code and the motor controller code. The first section we will look at creating the controller then we

will create the code to control the motor.

About the motors

We will be using old 1980's motors like below:



These motors are easy to pick up from site like Ebay, Take a look online to see if you can pick some up for a cheap price.

With the older motors they come with a cable that connects the train motor to the battery carriage, I unscrewed one end of the cable so that I could feed the two wires into the motor control board.

On the older motor to connect the motor to the motor controller board, you will need to unscrew the Lego metal pins to expose the two wires then connect them to the motor one sockets on the motor controller.

Newer power functions motor from 2005 like below:

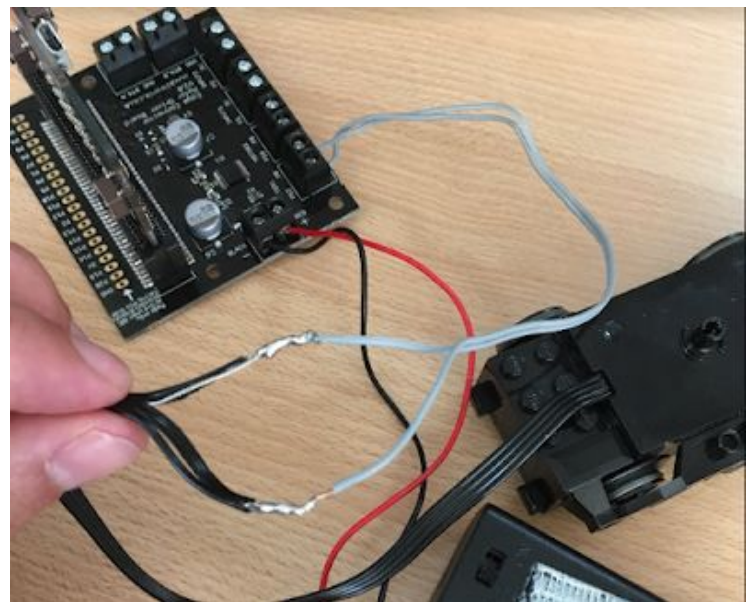


with this newer style motor, which is what you will find in most modern lego trains the wire that comes from the motor has 4 wires that are stuck together which then attach to a 1x1 lego plate which attaches to the Lego Power Functions battery pack. I wasn't interested in this so I cut the cable just before this and stripped the wire back so you could see each of

the 4 wires. The two left ones were soldered together and the two right ones were. This was then soldered onto the existing Lego cable that I had. This wasn't necessary. The sellotape was to protect the wires from touching.



The cable looked like this after it had been soldered together and then connected to the motor controller board:



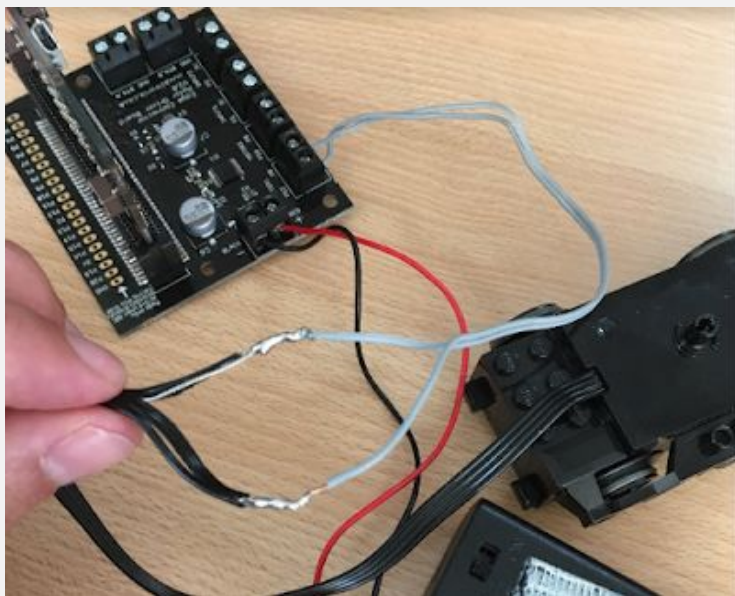
On the newer style motor The wires were then soldered together as above.

Stage 1:

Take the one of the two micro:bits and the motor controller mentioned previously. plug in the micro:bit into the motor controller as shown on the next page:



Then find the 4.5-volt battery pack and connect the wires from it into the motor controller board. You can see below the red wire goes into positive and black negative. Tighten the screw to secure wires. Then get the wires from the Lego train motor and connect them into the two motor 1 slots and tighten the screws to secure them in place. Again you can see below:



Next, you will need to add the batteries to the battery box. Please make sure that it is turned off so you don't waste your battery. Congratulations the wiring is sorted. Next up stage 2 the code.

Stage 2: Coding the remote control

Next up, we need to create the code to get this to work. Firstly we will code the remote, I have used micro python and there are three methods you can use to achieve this. Method one is installing the 'Mu' editor which can be installed from here: <https://codewith.mu/>.

Method two is by using the web based editor which can be accessed here:

<https://python.microbit.org/v/1> or you can use Edublocks, this uses/creates the same code but it is drag and drop similar to Scratch, you may prefer this option you can access this from:

<https://microbit.edublocks.org/>.

I used option 2 because I found it to be quicker for what I needed. I will leave it for you to make up your own mind.

Using option 2 I created the following code:

```

1 from microbit import *
2 import radio
3 radio.on()
4
5 option = 0
6
7 while True:
8     sleep(250)
9     option = option + button_a.get_presses()
10    display.scroll(str(option))
11
12    if button_b.was_pressed():
13
14        if option == 1:#fwd
15            display.scroll("F")
16            radio.send("Forward")
17        elif option == 2:#bck
18            display.scroll("R")
19            radio.send("Back")
20        elif option == 3:#stp
21            radio.send("S")
22            display.scroll("S")
23            radio.send("Stop")
24            option = 0
25        elif option == 4:#stp
26            radio.send("Coded")
27            display.scroll("Co")
28            radio.send("Coded")
29            option = 0

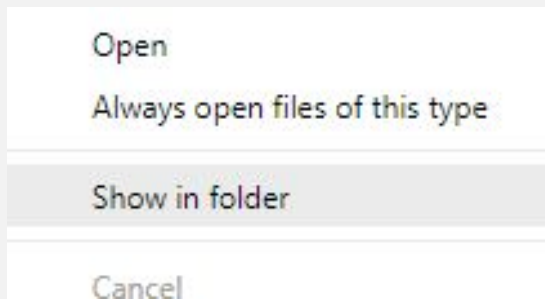
```


The micro:bit Express **:make**

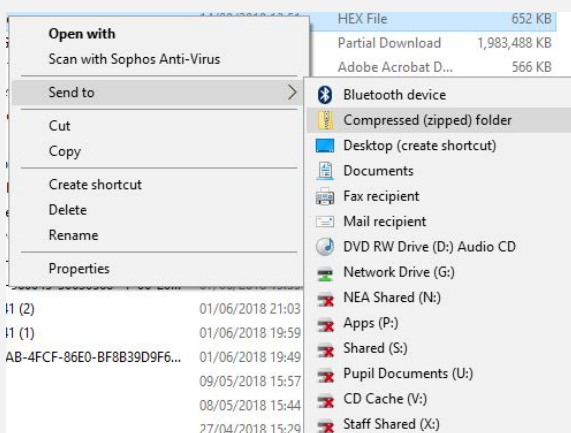
Next step is to load the code onto the micro:bit that you will use for your remote, ensure that you have plugged your micro:bit into your machine. Then on the website select download to download the hex file onto your pc like below, it should appear like so:



This is using the Chrome browser. Then left click and click on show in folder:



Next up to get the hex file onto the micro:bit, right-click onto the hex file and then select 'send to' then select 'micro:bit' on the right-hand side. This will then flash for 10- 15 seconds, you can see this by looking at the back of the micro:bit you will see an orange flashing light.



Congratulations you have now created your remote controller.

Stage 3 The code to control the motor. As we did in stage 2 you have three options to create your code I chose option 2.

The code I created looks like this:

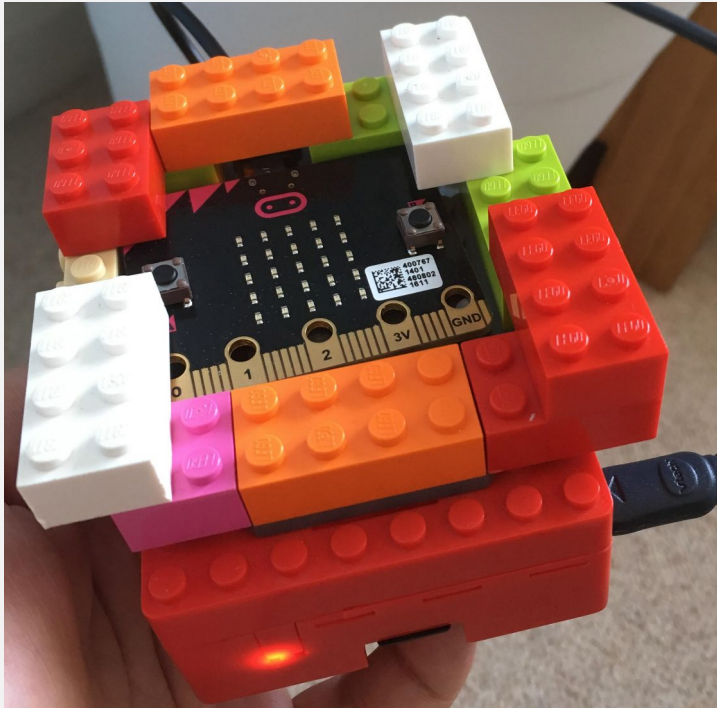
```

1 from microbit import *
2 import radio
3
4 radio.on()
5
6 def fwd():
7     pin8.write_digital(1)
8     pin12.write_digital(0)
9     display.scroll("F")
10
11 def stp():
12     pin8.write_digital(0)
13     pin12.write_digital(0)
14     display.scroll("S")
15
16
17 def rev():
18     pin8.write_digital(0)
19     pin12.write_digital(1)
20     display.scroll("R")
21
22 while True:
23     incoming=radio.receive()
24     laps = 0
25     if incoming == "Forward":
26         fwd()
27     if incoming == "Back":
28         rev()
29     if incoming == "Stop":
30         stp()
31     if incoming == "Coded":
32         while laps <= 8:
33             fwd()
34             sleep(500)
35             stp()
36             sleep(2000)
37             rev()
38             sleep(500)
39             stp()
40             laps = laps + 1

```

Now you will need to repeat the steps from stage 2 to upload your hex file onto the micro:bit.

Congratulations you should now be ready to test your micro:bit Express. You will need to slide this micro:bit into the motor controller slot like below:



Now you will need to power the remote by using plugging it into your pc. Then turn on the battery pack connected to the motor controller. If you click the on the 'a' button on the remote and press 'b' to confirm the train should move 1 should be forward, 2 is back and 3 stop and 4 is a coded path. When you get 3 and 4 it will reset to 0.

Congratulations you have now finished your coded Lego train.



Taking the project further

1. Why not try using a Raspberry Pi to power the remote
2. Try creating your own coded path.
3. Have a go at some more projects on Chris' blog:
go.micromag.cc/jammy

Pradeeka Seneviratne



Pradeeka is a technological Writer, avid maker, and loves to design wearable tech project, just like this body position sensor.

[@pradeeka7](#)



Image Credits: "Designed by yanalya / Freepik"

Body position sensors provide a signal that indicates to the system the patient's sleeping posture.

Building a Body Position Sensor

Easy to build wearable body position sensor and an Android app to monitor five different patient positions (standing/sitting, supine, prone, left and right.)

You Will Need:

- A micro:bit
- 2xAAA battery holder and batteries
- micro-USB cable
- Computer with Internet connection, WiFi connected
- Smartphone or tablet running Android OS, Connected to the same WiFi network
- Piece of thick cardboard
- Pair of scissors and a paper cutter
- Double sided tape
- Belt (clothing) or flexible band or strap

Body position sensors provide a signal that indicates to the system the patient's sleeping posture. In this project, you build a wearable body position (posture) sensor with a micro:bit. You also develop a simple Android app with MIT App Inventor to monitor patient's posture.

You will be using Bluetooth UART service to send data from micro:bit to the Android app.

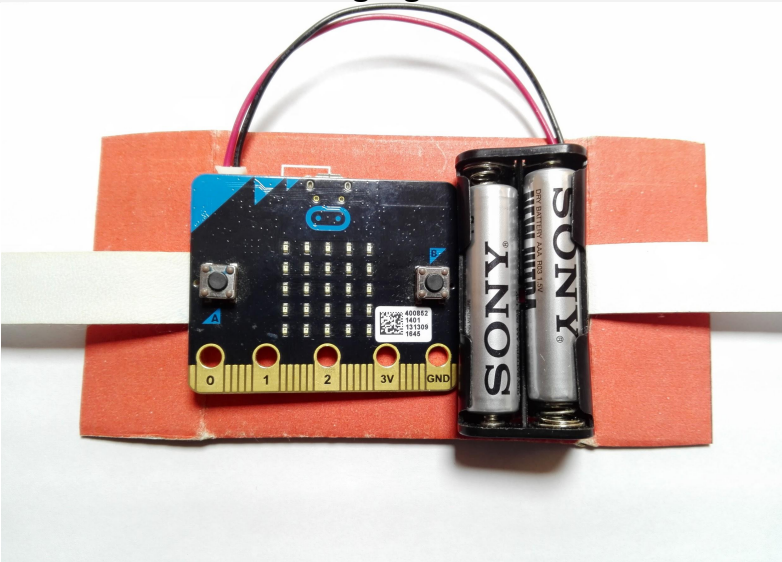
The body position (posture) sensor uses the triple axis accelerometer of the micro:bit to obtain the patient's body position. The sensor sends real-time body position data to an Android app through Bluetooth.

Here is the list of micro:bit accelerometer gestures can be used for each body gesture.

Body Position	Accelerometer Gesture
Standing/Sitting	logo up
Supine	screen up
Prone	screen down
Left	tilt left
Right	tilt right

Assembling Hardware

Let's build a prototype version of the wearable. Using a piece of double-sided tape, stick the micro:bit on the cardboard. Then take another piece of double sided tape and stick the 2XAAA battery holder next to the micro:bit. Plug the battery connector and insert 2XAAA batteries. Insert the belt through the cardboard holder as shown in the following figure.



Assembled body position sensor.

Building the Code for micro:bit

First, build a program for micro:bit to detect different body positions using MakeCode for micro:bit

(<https://makecode.microbit.org/>).

When micro:bit detect a new gesture, the name of the gesture will send to the app using the Bluetooth UART service.

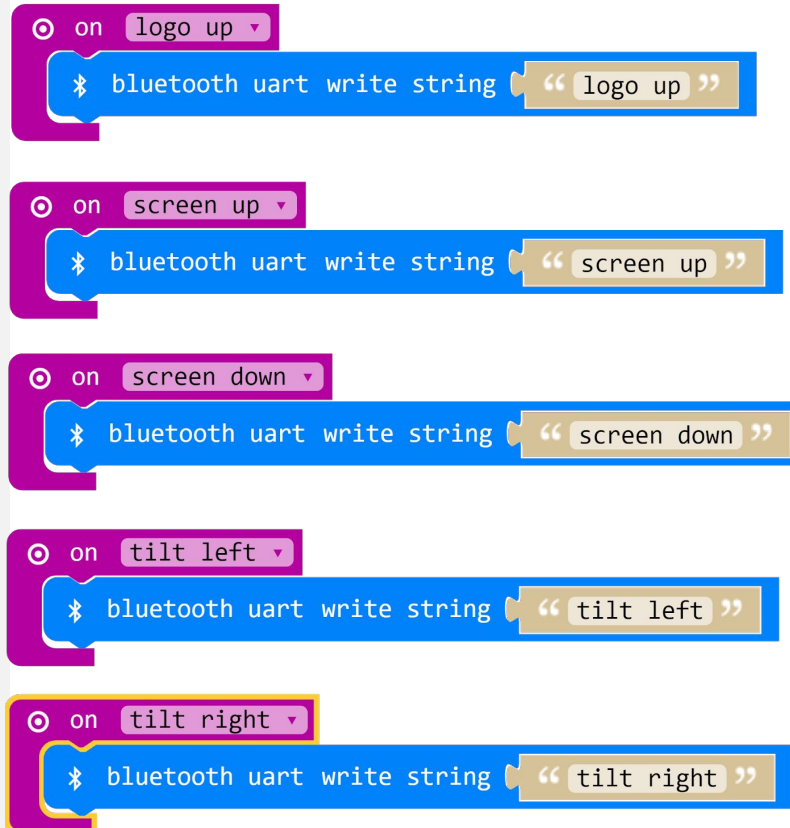
1. First, you should install 'Bluetooth' package for micro:bit. In the toolbox, click 'add package'. Then, in the 'Add Package' dialog box, click 'bluetooth'.
2. Click, 'Remove package(s) and add

'bluetooth' button to remove the existing 'Radio' package and add the new 'bluetooth' package.

3. You have now installed the Bluetooth package and are ready to use bluetooth UART service for the micro:bit.
4. Add an event to indicate when the micro:bit is connected to over Bluetooth. In the toolbox click 'Bluetooth' followed by 'on bluetooth connected'. Then, in the toolbox click 'Basic' followed by 'show icon' block. Drag the 'show icon' block into the 'on bluetooth connected' block. In the 'show icon' block, click on the drop-down list and choose 'YES' icon.
5. Add another event to indicate when the Bluetooth connection to the micro:bit is lost. In the toolbox click 'Bluetooth' followed 'on bluetooth disconnected'. Then, in the toolbox click 'Basic' followed by 'show icon' block. Drag the 'show icon' block into the 'on bluetooth disconnected' block. In the 'show icon' block, click on the drop-down list and choose 'NO' icon.
6. In the toolbox, click 'Bluetooth' followed by '...More'. Then click 'bluetooth uart service' block. The block will add onto the coding environment. Now, drag the 'bluetooth uart service block' into the 'on start' block.
7. In the toolbox, click 'Input'. Then click 'on gesture' block ('on shake' is the default

selection) to add it onto the coding environment. Click on the drop-down list of the 'on gesture' block and select 'logo up'. This gesture is corresponded to the body position 'standing/sitting'. In the toolbox, click 'Bluetooth' followed by '...More'. Then click 'bluetooth uart write string' block. The block will add onto the coding environment. Now drag the 'bluetooth uart write string' block into the 'on logo up' block. In the text box, type in the text 'logo up'.

8. Now duplicate the 'on logo up' event four (4) times. Modify them for the gestures 'screen up', 'screen down', 'tilt left', and 'tilt right'. Type in the text boxes for uart write, 'screen up', 'screen down', 'tilt left', and 'tilt right' respectively.
9. Following figure shows the complete code built with the MakeCode for micro:bit.



Code for micro:bit.

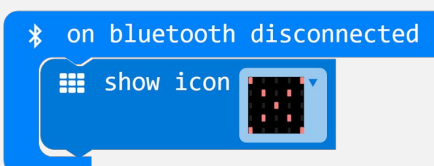
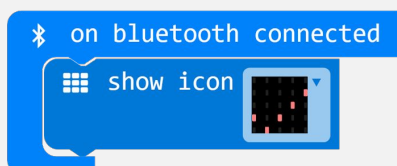
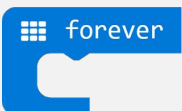
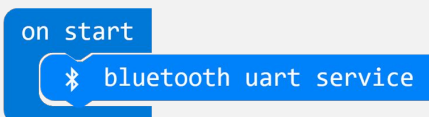
10. Download the hex file and flash it on to the micro:bit.

Building the App with MIT App Inventor

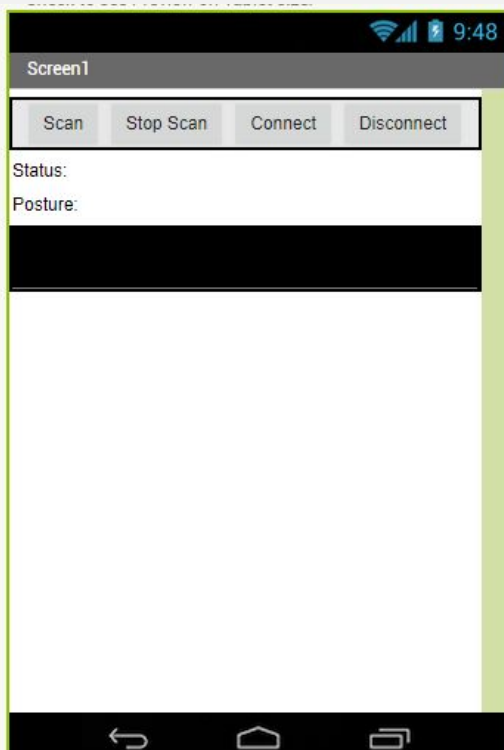
The MIT App Inventor can be used to quickly develop Android apps for micro:bit. The steps below will guide you how to develop a simple app to communicate over Bluetooth with micro:bit.

1. Go to MIT App Inventor (<http://ai2.appinventor.mit.edu/>) and sign in with your Google account.
2. Start a new project and type in 'body_posture_monitor' in the 'Project name' box.
3. Drag a 'HorizontalArrangement' form the 'Layout'.

MakeCode Blocks:

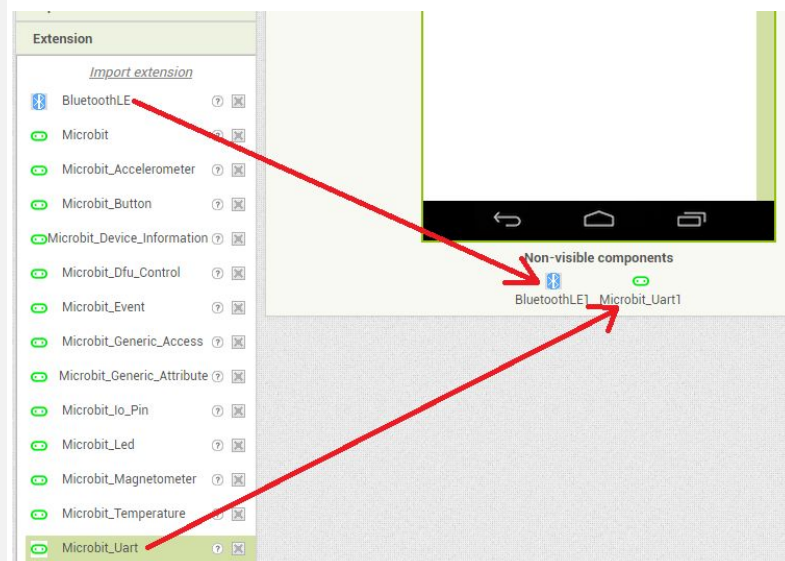


4. Then add four buttons from the 'User Interface'. Rename them as 'ButtonScan', 'ButtonStopScan', 'ButtonConnect', and 'ButtonDisconnect'.
5. Change their text to 'Scan', 'Stop Scan', 'Connect', and 'Disconnect'.
6. Below the 'HorizontalArrangement' add a Label from the 'User Interface'. Rename it as 'LabelStatus' and change its text to 'Status'.
7. Add another Label below to the 'Status' and rename it as 'LabelData'. Change its text as 'Posture'.
8. Drag a 'ListView' from the 'User Interface' and place it below the 'LabelStatus'. Rename it as 'ListBLE'.
9. The following figure shows the completed UI.



Completed UI for the app.

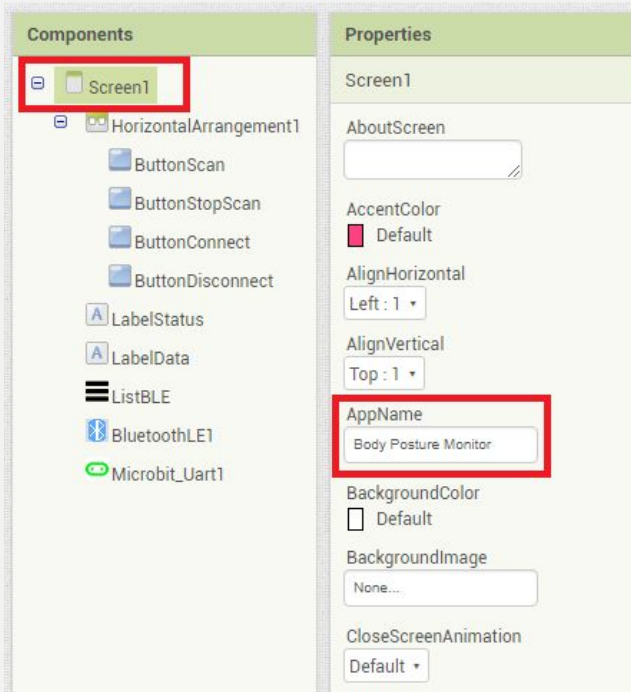
10. Download the BluetoothLE extension from <http://iot.appinventor.mit.edu/assets/edu.mit.appinventor.ble.aix>.
11. In the 'Palette', click on 'Extension' at the bottom and then on 'Import extension' and then 'Choose File'. Browse the downloaded extension on your computer and 'Import' it.
12. Drag the 'BluetoothLE' extension onto the viewer.
13. Click 'Import extension' again, paste in the URL **'<http://iot.appinventor.mit.edu/assets/com.bbc.microbit.profile.aix>'** and click the 'Import' button.
14. Drag the Microbit_Uart extension onto the viewer.
15. The following figure shows the viewer with above two extensions.



Adding extensions.

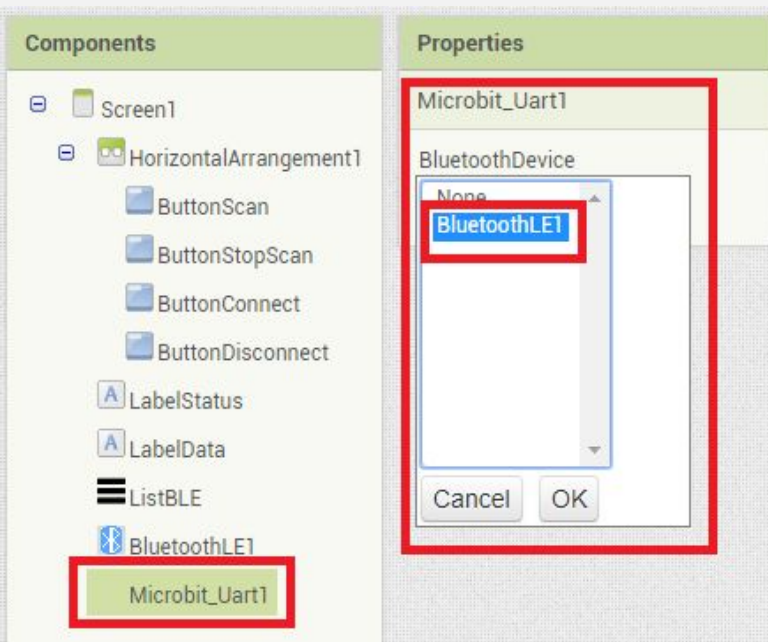
Body Position Sensor :make

16. In the 'Components', click 'Screen1' and type in 'Body Posture Monitor' in the 'AppName' text box as shown in the following figure.



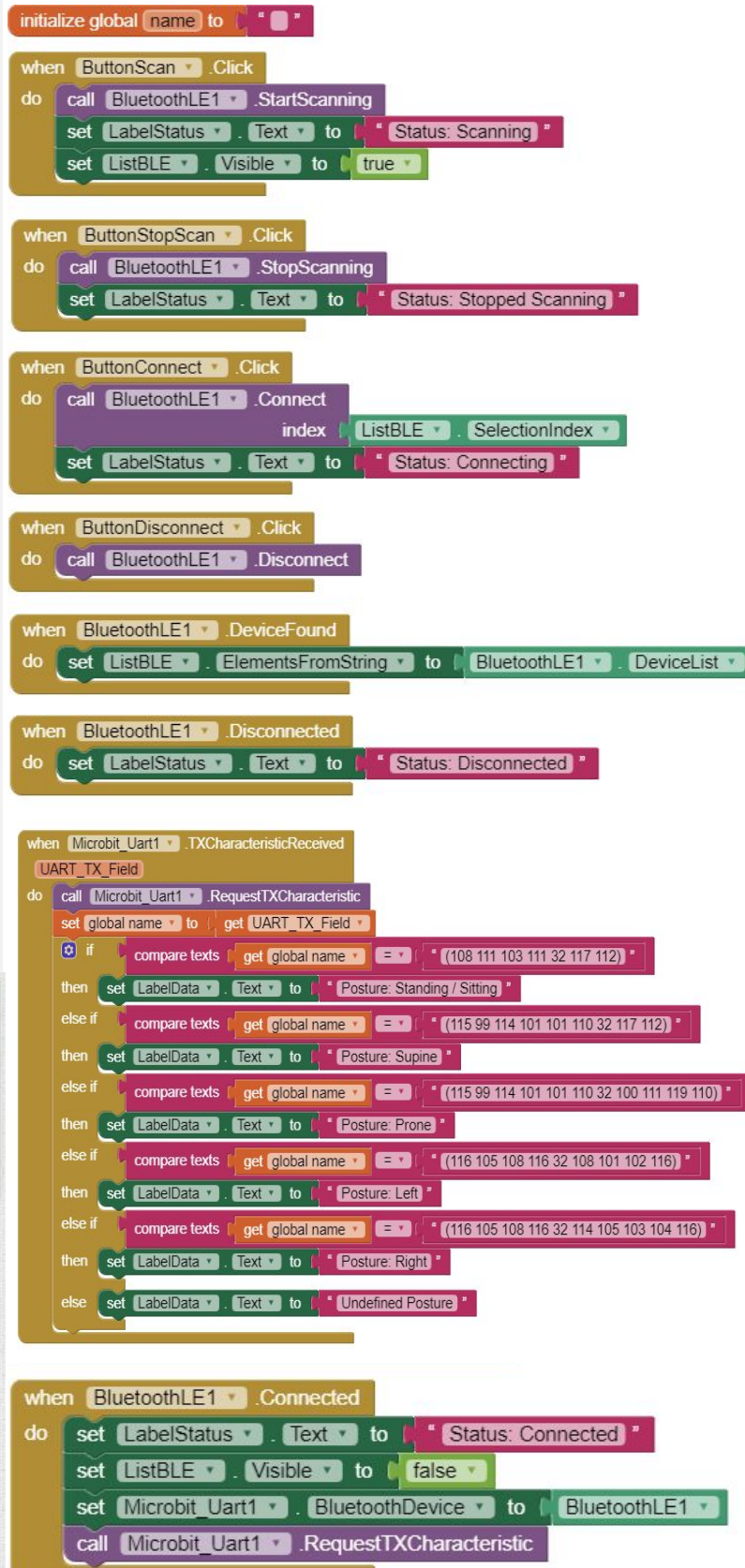
Editing the app name

17. In the 'Components' click 'Microbit_Uart1' and from the 'Properties' choose 'BluetoothLE1' as shown in the following figure. Then click OK.



Configure UART with Bluetooth.

18. Click on the 'Blocks' to switch to build the program with blocks.
19. Build the program for Android app as shown in the following figure.



Code for Android app.

When you've finished your code, make sure that you have not missed a block as this may cause your code not to work properly and throw errors. It's good practise to go back and check your code before you continue.

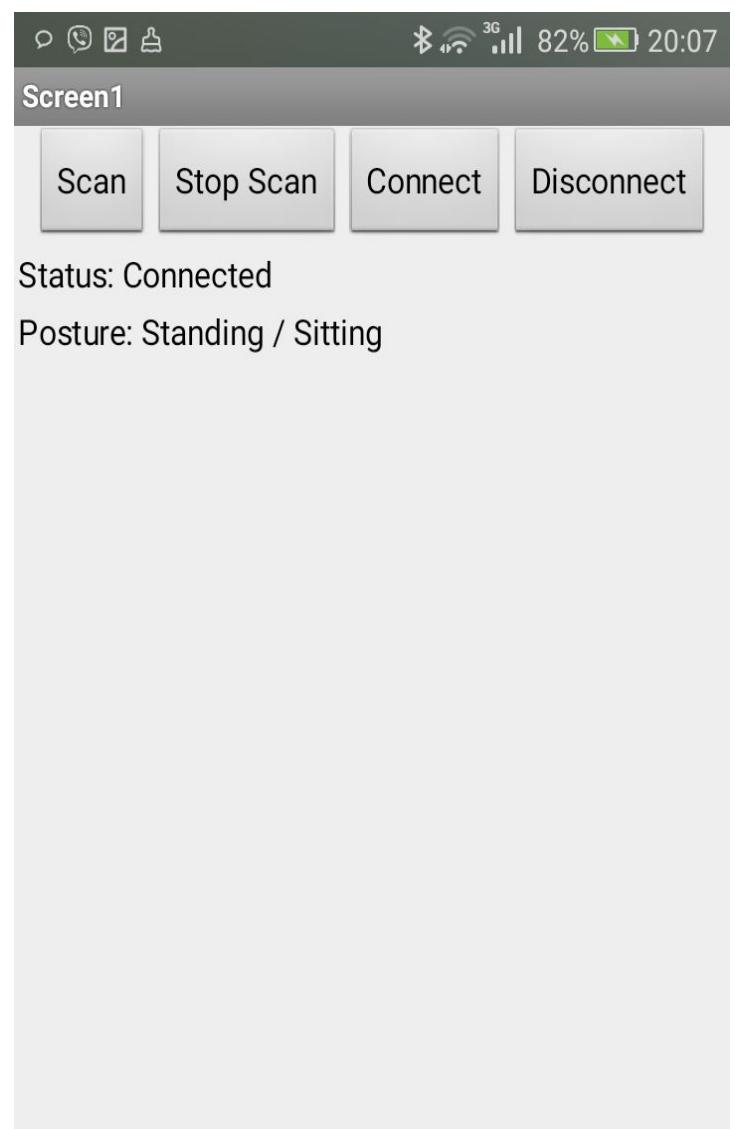
Wear It

Wear the body position sensor using the belt just above the waistline.

Use It

1. First, you will need to pair your Android smartphone or tablet to the micro:bit.
2. Install 'MIT AI2 Companion' app on your Android device from the Google Play.
3. In the MIT app inventor, click Connect | AI Companion. You will get a pairing code something like this: 'biuosv'
4. Then open the 'MIT AI2 Companion' app.
5. Type in the pairing code (six-character code) in the text box. Then tap 'connect with code' button.
6. A runtime version of your app will start on the smartphone.
7. Tap 'Scan' button to scan all the available micro:bit boards.
8. Once found, tap the 'Stop Scan' button.
9. Then, tap on the name of your micro:bit.
10. Finally, tap the 'Connect' button

11. The status will change to 'Connected'. Also, a 'tick/YES' will display on the micro:bit display.
12. Once connected, change your body to different positions. The app will display the real-time position of your body on the Android app.



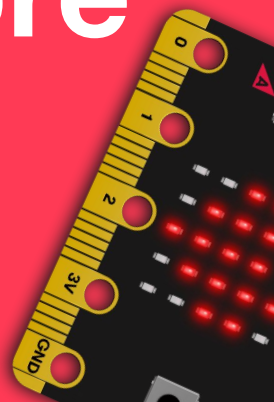
Taking the project further

You can save body position data to Google Fusion tables for further analyze.

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.

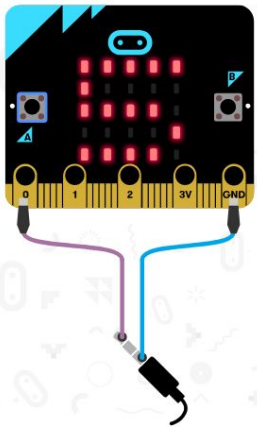
Get in touch for more
info - email us on
hello@micromag.cc



Countdown Timer

Make your very own micro:bit countdown timer in 9 simple steps

Timers are handy, they can time a boiled egg to perfection, help us take group photos, and be used to help rockets to launch. We shall make our own timer using a micro:bit, some crocodile clips and a speaker / headphones. You will need to connect the headphones to the micro:bit as per the image below..



Follow this wiring guide to connect your speaker.

Step 1: Working with input



From the Input menu drag the “on button A pressed” block into the coding area. Any code inside the block will be run when button A is pressed.

Les Pounder

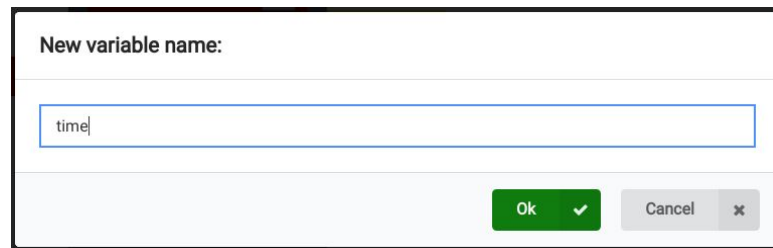


Les is a maker and trainer who has worked with the Raspberry Pi Foundation and the BBC to deliver computing training.

@biglesp

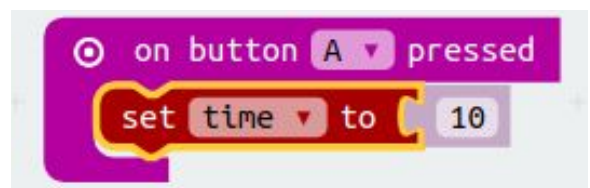
bigl.es

Step 2: Create a variable



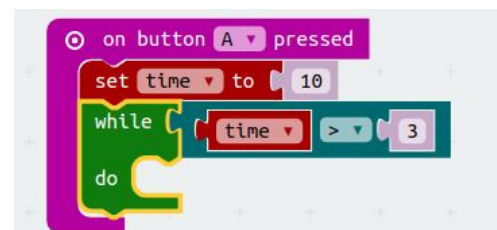
In the Variables menu click on “Make a Variable” and call it “time” We shall use this variable to store the duration of our countdown.

Step 3: Using the variable



From the Variables menu drag “set item to 0” block to the coding area. Click on the arrow next to “item” and change to “time” then change 0 to 10.

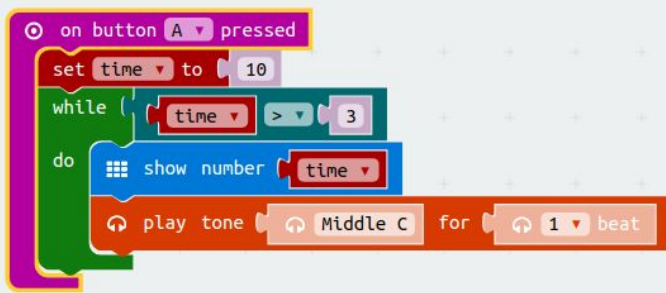
Step 4: Creating a loop and a test



We use a while true loop from loop and from

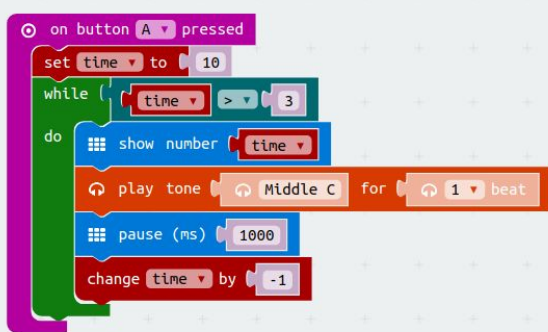
logic we connect a `_ > _` block. Then from Variables we drag time and place it in the left space and type 3 in the right space.

Step 5: Showing and hearing the countdown



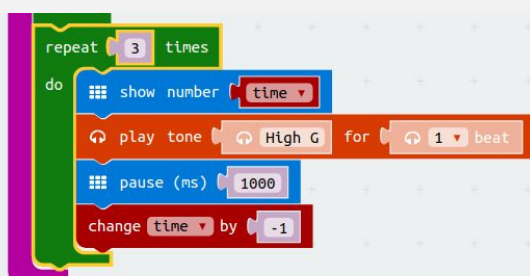
From the Basic menu, we use Show Number and use our time variable as the number. Then from Music, we use Play Tone and choose middle C to indicate a countdown is running.

Step 6: Countdown and pause



Each time the countdown runs, it needs to pause for 1 second (pause (ms) from Basic) and then change the value of our time variable by -1.

Step 7: Final 3!



Once the countdown reaches 3, we use another loop, and for 3 loops we change the tone to a

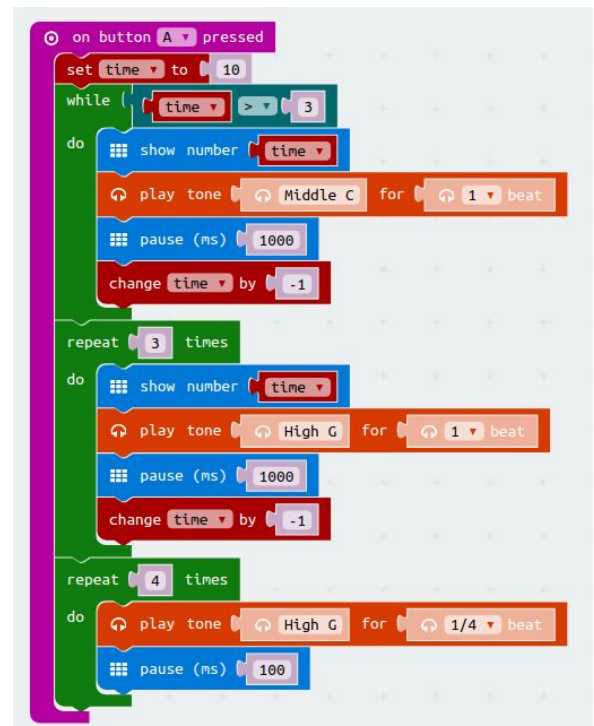
higher note, to indicate that there is not much time left.

Step 8: End on a high note!



With the countdown reaching 0 we trigger a loop that will repeat 4 times. Each time it goes around, it will play a high G note really quickly. Just like an alarm clock!

Step 9: Putting it together

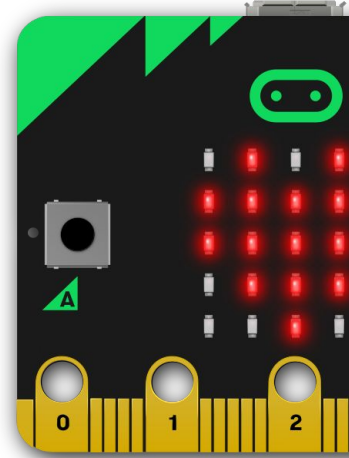


Here is all of the code for this project. We have learnt how to use loops, conditional tests, input, and make music with the micro:bit!

That's it, now download the code to your micro:bit and when it has finished, press the A button to start the countdown!

Well done you have made your own countdown timer! Can you change the timer to time a boiled egg?

Meet the foundation: Global Engagement Team



Meet the mighty team of 4 people that are responsible for making the micro:bit available across the world..

Have you ever wondered how the micro:bit gets into the hands of so many young people in so many countries? Meet the Global Engagement Team! They keep themselves busy talking to governments and ministries of education around the world to encourage the use of micro:bit in schools and to make sure that all their education materials are as accessible as possible. They also attend as many trade and education events as they can, talking to their ecosystem and helping accessory makers and resellers to get the information they need to get their products onto the market. Ready to meet them? Come and say hello...

Hal - Chief of Global Engagement



Hal heads up their global engagement team and is in charge of all things micro:bit in North America. He has a long history in the hi-tech industry and sees digital skills – particularly computational thinking and coding – as the key to socioeconomic mobility. He believes the micro:bit to be, “a magical device that enables children to connect the virtual world of software with the physical world of making”. His ambition is to give every child, the world over, the opportunity to express their digital creativity with a micro:bit. Go, Hal!

Waris - Head of Asia

Waris oversees micro:bit development activities in the Asia Pacific region. Based in Hong Kong, he works tirelessly with educators, schools and universities to build a sustainable micro:bit ecosystem ... and partners with local policymakers, accessories makers, developers, and resellers to create even more micro:bit magic! Even at weekends, Waris can be found at Hour of Code events and introducing the micro:bit to students around Asia.



Rachel - Head of Product and Channels

Rachel looks after the Foundations relationship with Premier Farnell – the people who make and distribute all those lovely micro:bits. (In fact, she used to oversee UK distribution for Raspberry Pi and the micro:bit, too!) Based in the UK, Rachel also looks after resellers, accessory makers and curriculum providers, as well as leading on activities across Europe. Phew! She's been in the electronics industry for many years, and absolutely loves helping children learn new skills with the micro:bit.



Jose - Head of Latin America (Brazil)

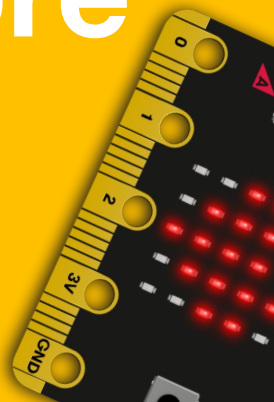
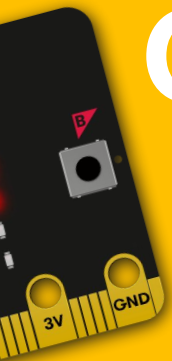
Jose is responsible for micro:bit activities in Latin America. Based in Brazil, he's a veteran of the hi-tech world and has been involved with key education initiatives in the region for quite some time. He sees technology as, "a powerful tool to promote social inclusion for children and underprivileged adults" and has been involved in fantastic international projects such as One [Laptop Per Child](#) and the Brazilian Ministry of Education's drive to distribute tablets to state school teachers.



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**





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

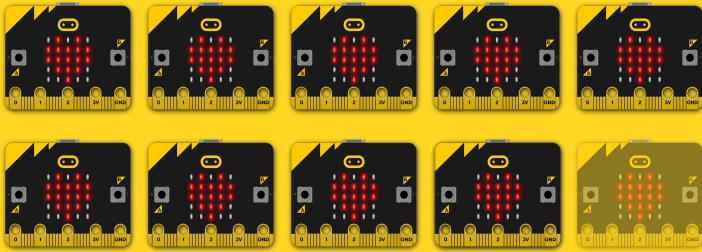
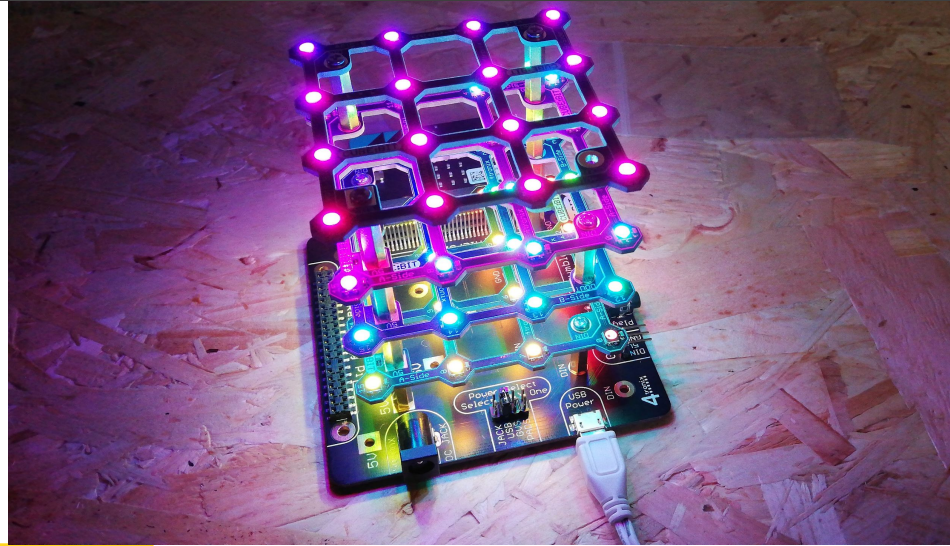
hello@micromag.cc

[@micro mag](https://twitter.com/micro_mag)



4tronix cube:bit

Build a micro:bit 3D LED cube with cube:bit/



the 'slices' together. You don't have to have any experience at all to build it.

Once you have built your cube, you locate the micro:bit edge connector port on the base and slot in your micro:bit to connect it all up.

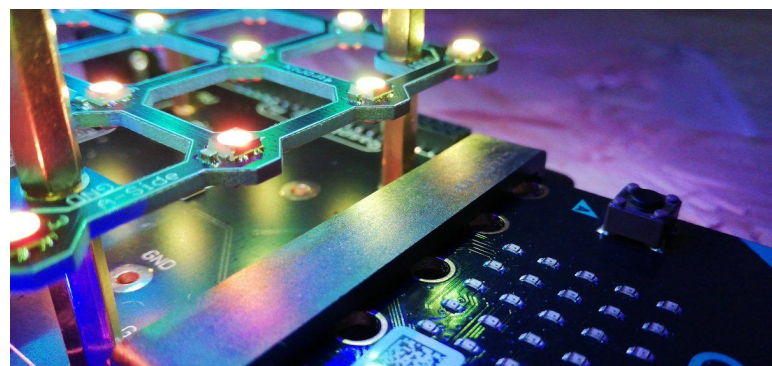
You may have seen the amazing cubert project

9/10

built by Lorraine Underwood. This dazzling project which is a micro:bit powered 8x8x8 was the main inspiration for maker company 4tronix's latest product, cube:bit.

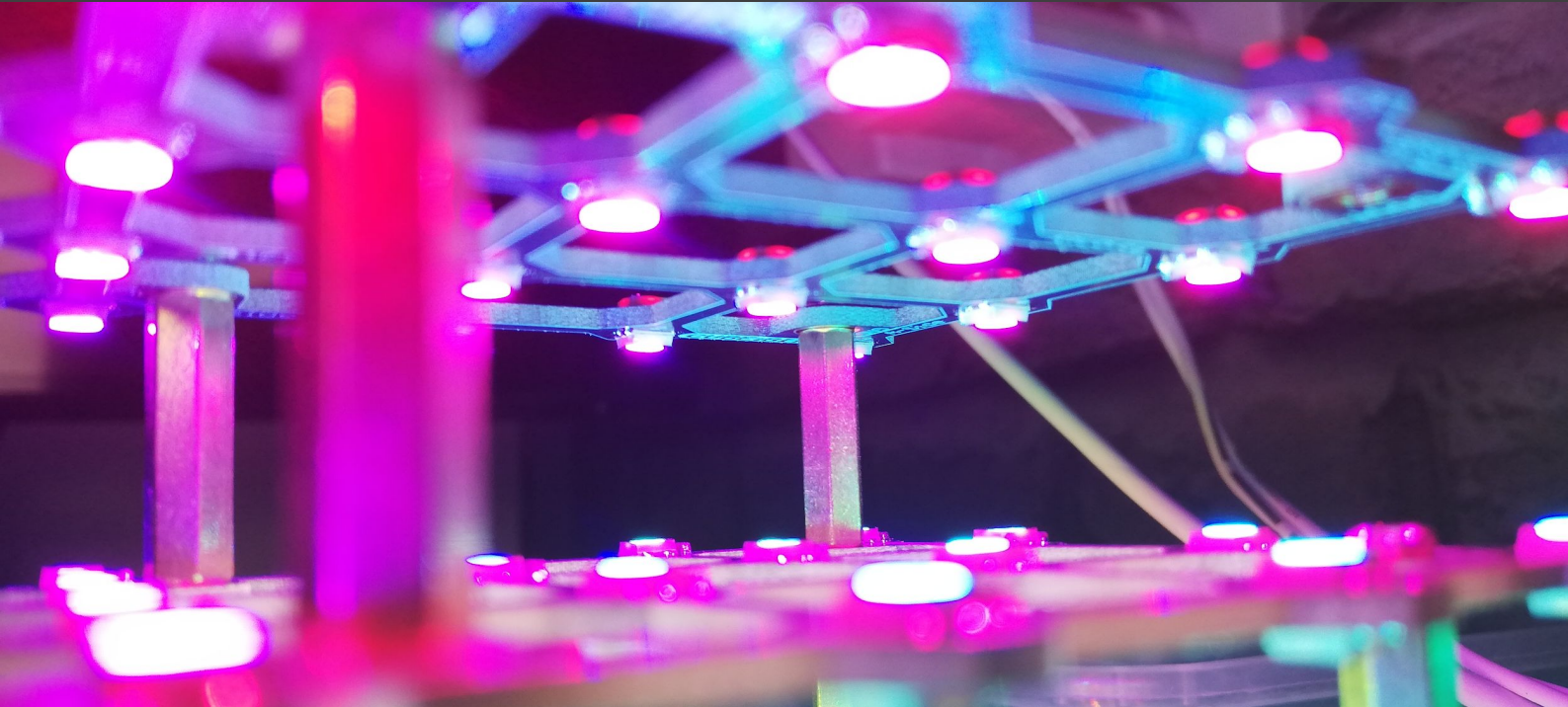
This dazzling micro:bit add-on consists of 3 different versions, a 3x3x3, 4x4x4 and a 5x5x5 cube. The cube itself can be used with crocodile clips connected to the micro:bit or for £10 extra, you can purchase the base board where there is an edge connector for micro:bit and other connections.

The cube is a little fiddly to build but it's just a case of following the handy instructions at go.micromag.cc/cubeguide and using some metal standoffs to put all



BBC micro:bit slotted into the cube:bit base.

Due to the high power of the LEDs on cube:bit, extra power is needed if you are to use the cube to its full potential. You can provide extra power through many different ports on the base, for example, we used a 5V micro USB phone charger to power our cube and this seemed to work ok. It's easy to switch between the different power inputs using a jumper on a power select header on the base, this makes it easy for beginners as it's not a complicated task to do.



The cube:bit has LEDs on the top and bottom giving it a nice effect.

Gareth from 4tronix has created a lovely easy to use MakeCode package which you can install into the micro:bit MakeCode editor. This enables you to have full control over your cube but in a way that is easy and user friendly. Also on the website is a few code examples that you can copy into MakeCode, however, we found that there is a lack of in depth tutorials for this product which lets it down slightly by not providing user guides for beginners.

You can also control your cube:bit using MicroPython by treating it as a string of neopixels, although, it's more complex to create animations with MicroPython as there is no library for the cube:bit.

Cube:bit is also cheap to buy, with the basic 3x3x3 model priced at £22, the cubes also come in a premium style packaging which is a nice touch.

The cubes are based off WS2811 or Neopixels as you may know them and they are placed on both sides of the slice. The handy labels on the slices themselves make it easy to reference the pixels when coding them, as well as the A + B side labels when your building them. The metal standoffs make the connections between the slices, and this is a really effective way to connect the cube together.

Overall, we think this cube:bit is a shiny addon for the micro:bit which is really well designed and easy to use in MakeCode. However the lack of detailed guides for beginners and no python library sets this back from getting a 10.

Buy the 4tronix cube:bit:
go.micromag.cc/cubebuy
Remember to get the base for micro:bit support!

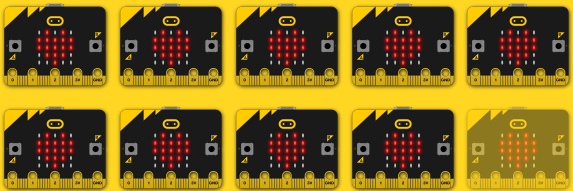


Daniel Pers is a STEM teacher (K8 to K12) in the Lycée Pilote Innovant International (Poitiers, France).

daniel.pers@ac-poitiers.fr

DFRobot Micro:bit Driver Expansion Board review

An original solution, versatile, powerful, but low-cost for projects with motors



9/10

DFRobot has just released the expansion board DFR0548 for micro:bit with 9 inputs / outputs, 2 I2C ports, 8 outputs for 8 servo motors and 8 outputs for 4 motors (or 2 stepper motors).

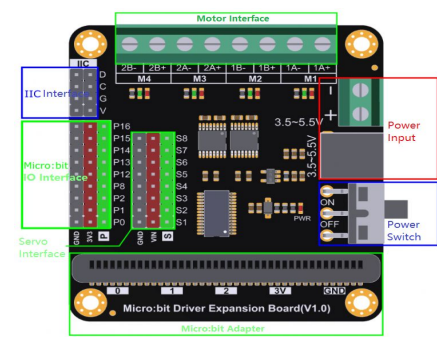
Inputs / outputs

This DFR0548 board has significantly more inputs/outputs than the competition. Unfortunately, it has only 3 analog inputs because the 6 pins used by the LED matrix are not accessible. This simplifies its use and allows to have a fairly compact board but it's still a shame

not to use them, especially since the LED matrix can easily be disabled by the program.

The two I2C ports of this board don't allow to directly connect (without cable) an OLED display because they are male connectors, with a different order of the pins, ... Elekfreaks does better for example with its EF03405 board.

The originality of this DFR0548 board comes mainly from the fact that all the motors are controlled by I2C via a circuit PCA9685 which brings 16 additional outputs.



DFRobot Expansion Board :review

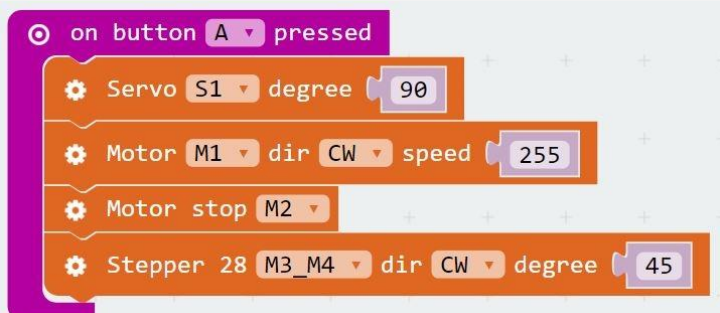
The library provided for MakeCode

Motor control is simple thanks to the package developed by DFRobot

go.micromag.cc/DFRobotDocs.

Some improvements must, however, be made:

- The control pulses of the servo motors have a width twice too large (1.2ms to 4.8ms instead of 0.6ms to 2.4ms).
- The PWM signal for controlling the speed of the motors has a frequency of only 50 Hz, which is much too low (impossible to filter the signal of a current sensor for example).
- The speed of the stepper motors can't be controlled.



Supply and power interfaces

The board is supplied by a voltage V_{in} between 3.5V and 5.5V from a wiring terminal or a DC 2.1 plug. DFRobot had a good idea to provide with this board a cable to connect the power supply to a USB port (a charger for example). The presence of an On / Off switch is useful.

The micro:bit board is supplied by 3.3V from this voltage V_{in} , but also from its micro USB port used to download a program.

The servo motors are powered directly from V_{in} . The motors are powered by 2 high-performance DRV8833 H bridges with a rated current of 1.5A. This allows the use of small standard geared motors, but I regret that DFRobot has chosen to use the same V_{in} power supply voltage while the DRV8833 support a voltage of 2.7V to 10.8V, especially since it has put two connectors for the power supply.

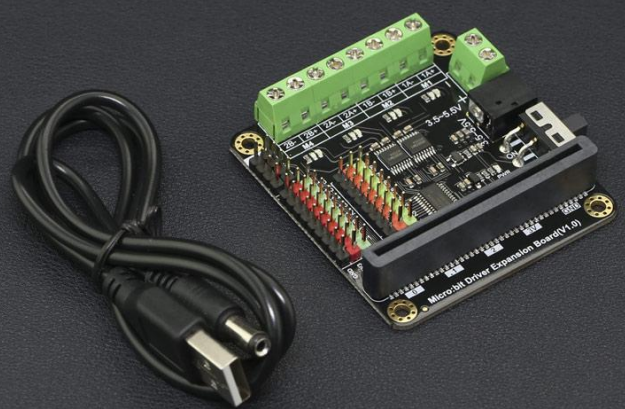
The ideal would have been a 3.5V to 10.8V power supply (practically compatible with a Li-ion 1S or 2S battery) + 5V USB power supply or 5V regulator for servo motors (with a jumper to choose).

Red and green LEDs indicate polarity for each motor. They seem useless. I would prefer an analog current sensor.

Conclusion

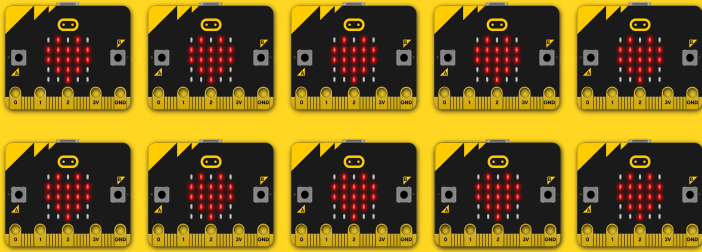
This expansion board is for me the best solution available today. In addition, it is one of the cheapest. Where can I buy one?

<https://www.dfrobot.com/product-1738.html>
(and at gotronic.fr in France).



Dexter GiggleBot

Making & Programming robots just got easier!



Here at micro:mag we love micro:bit robots.

10/10

These come in all shapes, sizes and prices but this new offering from Dexter industries makes it very accessible for kids. Let's take a look at what it has to offer.

Hardware

- 2 x DC Motors
- 9 x WS2812 Neopixels
- A pen Holder (Yes you can code GiggleBot to draw!) this is located between the wheels.
- 2 x line following sensors can be used to follow a thick black line on the floor
- 2 x light sensors one located on each side of the robot, you can code the light sensors to follow a torch.
- 2 x servo mounts

- 1 x AA battery holder for 3xAA
- 2 x connectors for I2C addons
- ATmega328PB microcontroller (the brains of the robot)
- DRV8837C 1A H Bridge motor controller (controls the motors and makes your robot move)
- A broken out micro:bit edge connector.

Build

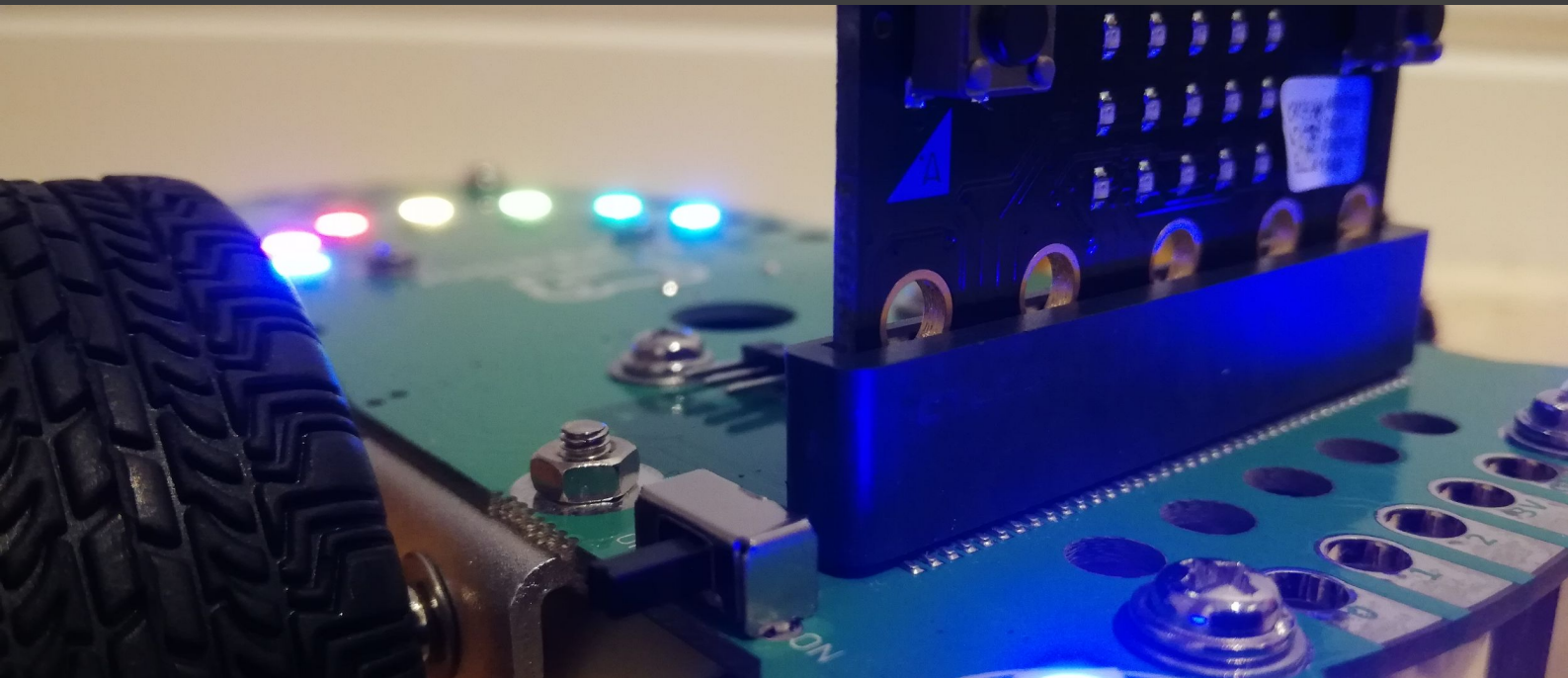
GiggleBot is probably the easiest robot to build as it comes almost completely assembled all we needed to do was push the wheels on to the motors and insert some batteries. Then it was ready to insert our micro:bit and get coding!

Programming

GiggleBot can be programmed using MicroPython using the [Mu editor](#), [EduBlocks](#) and [MakeCode](#), this is because it is powered by a micro:bit.

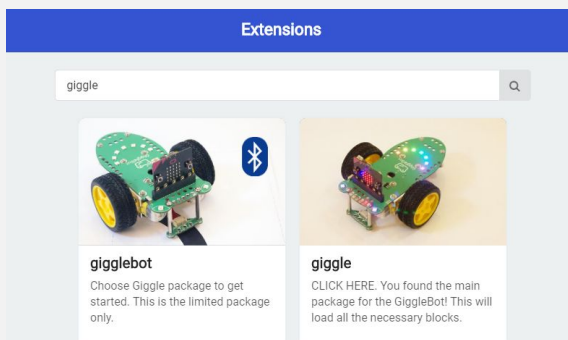
When test driving GiggleBot we used the MakeCode editor. To get started you have to install the giggle extension to do this go to Advanced -> Extensions and in the search box

Dexter Industries GiggleBot :review



Not only is the Gigglebot an incredibly good robot to use, it also looks amazing with its plethora of lights and lime green PCB chassis.

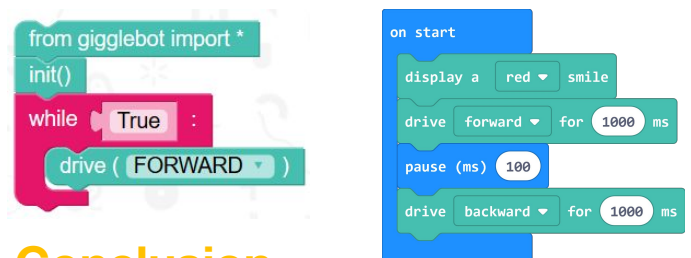
type giggle and press Enter.



With GiggleBot aimed at children, we are happy to report that it is very easy to program. Within a few minutes of looking at the code blocks, we were able to figure out how to make our robot move. We tried out our GiggleBot at a local code club, the kids really enjoyed using the robot and were instantly getting it to move, light up and sense.

Another thing we are really impressed with is the fact the robot is compatible with nearly all of the “major” micro:bit editors, allowing people of all abilities to use the GiggleBot making it the perfect robot for beginners and experts alike.

The MakeCode extension developed by Dexter is simple to use and has lots of blocks that let you control the robot with ease and For those who want to get a bit more advanced too, the EduBlocks and Python libraries are great for them, allowing you to use the GiggleBot with standard python features.



Conclusion

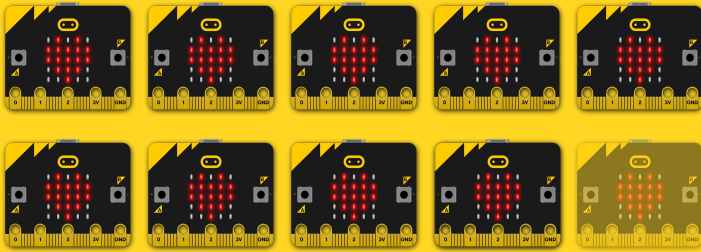
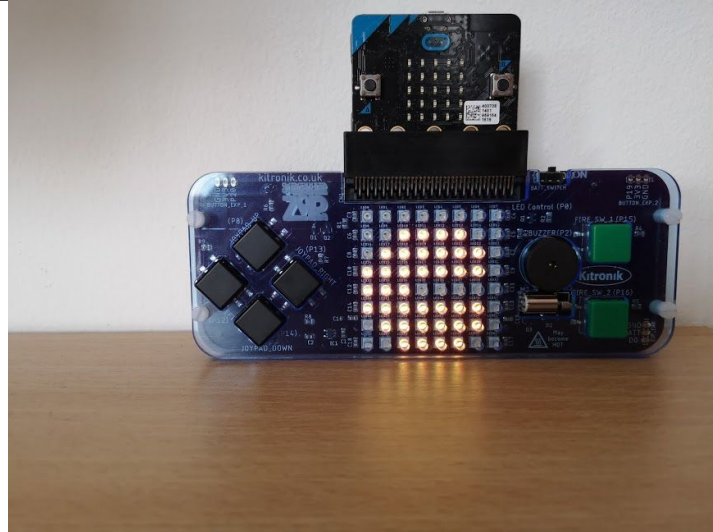
GiggleBot is priced at \$60 USD which is about £45 GBP which for what the robot offers, we think is a very fair price. Also, if you're in the US, Dexter offer free trials for schools in which you can loan some GiggleBots for 45 days. Overall this is an excellent beginners robot and is one of the easiest to use we've seen for the micro:bit.

Buy a GiggleBot:

go.micromag.cc/buygiggle

Kitronik :GAME ZIP 64

Break out your micro:bit into a portable games controller



Add Package... ?

Game Zip



kitronik-zip-64

Custom blocks for
www.kitronik.co.uk/5626-:GAME
ZIP64 for micro:bit

Ever tried to make your own games for the micro:bit but been limited by the two buttons? Well, the Game Zip 64 solves this problem by breaking out the micro:bit pins into 6 buttons, a speaker, a vibration motor and an 8x8 NeoPixel display. The Game Zip 64 also has room for 3 AA batteries on the back, which powers the Game Zip 64 and the micro:bit.

9/10

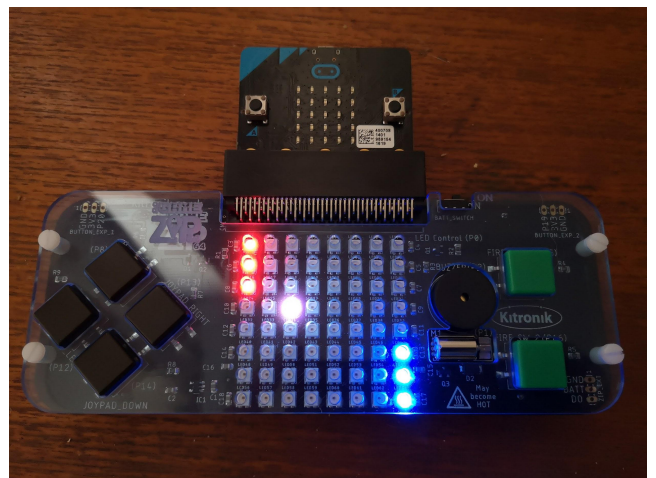
If you have a bit of experience with the micro:bit and the coding editors you can start creating games quite quickly. You are limited to the types of games you can make on the 8x8 NeoPixel matrix, but it is still a great way to start programming and being able to take your games with you.

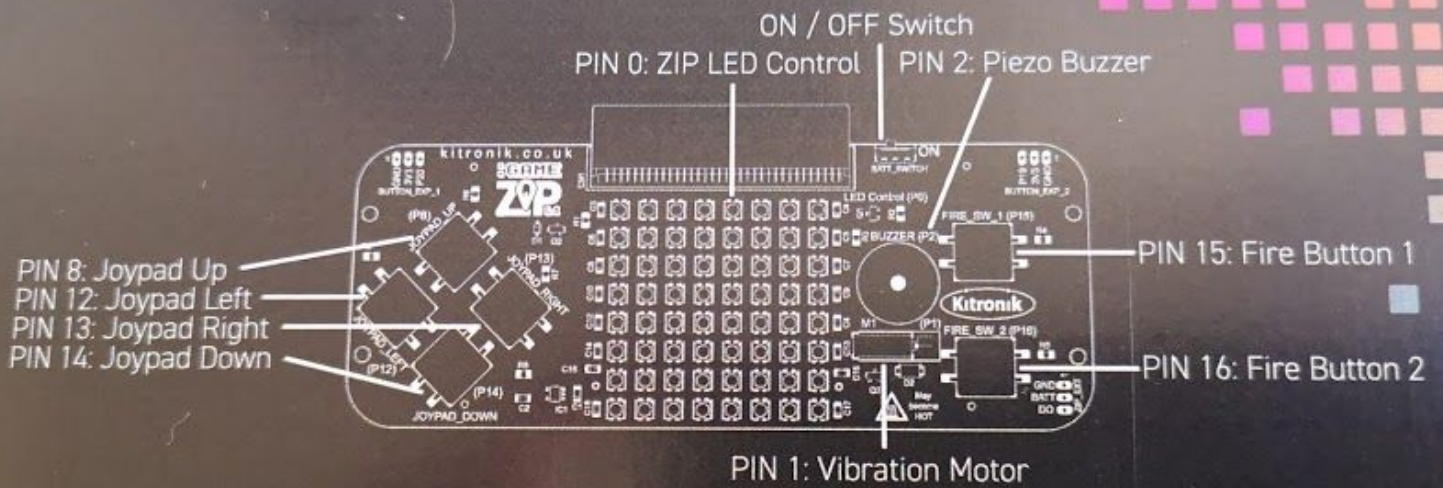
Kitronik has some really nice lesson plans to follow along with too. This is what we used to try out the programming of the Game Zip.

Programming

The Game Zip can be programmed with MakeCode and the beta version of MicroPython at the minute.

To get the Game Zip custom blocks in make code, go to Advanced -> Add Package then type in Game Zip and click on the Game Zip package that shows up, see image below.

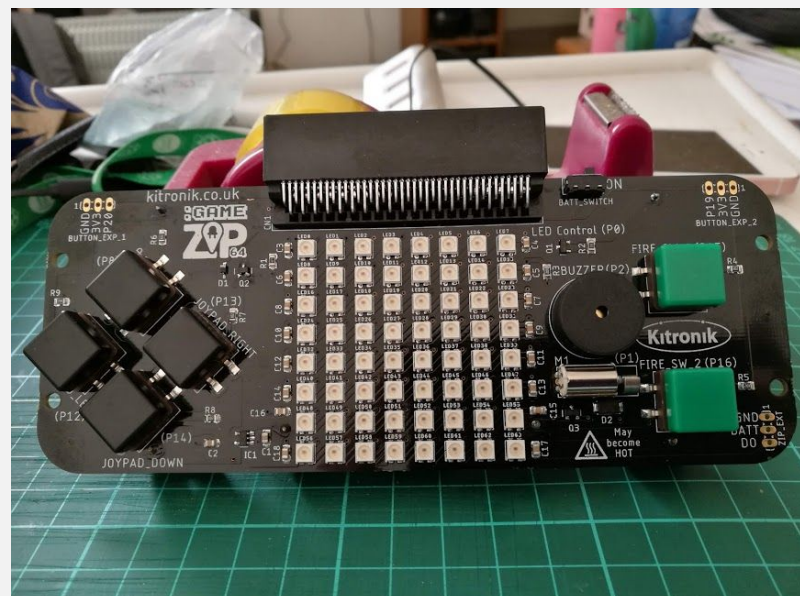
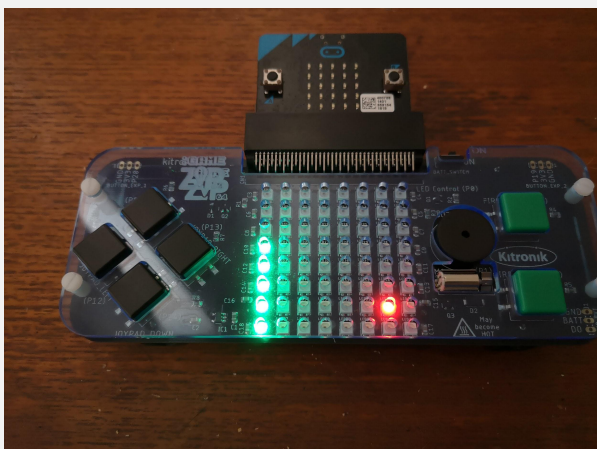




BBC micro:bit NOT included

Conclusion

The Game Zip is a simple yet effective way to get children into programming by creating simple games like Snake and Pong. Kitronik has these as demos that can be downloaded from their website at [kitronik](http://kitronik.co.uk). They were rather fun to play!



Buy the Kitronik :Game Zip 64
go.micromag.cc/buygamezip

micro:mag

micromag.cc