



arm

Get Started with Physical Computing

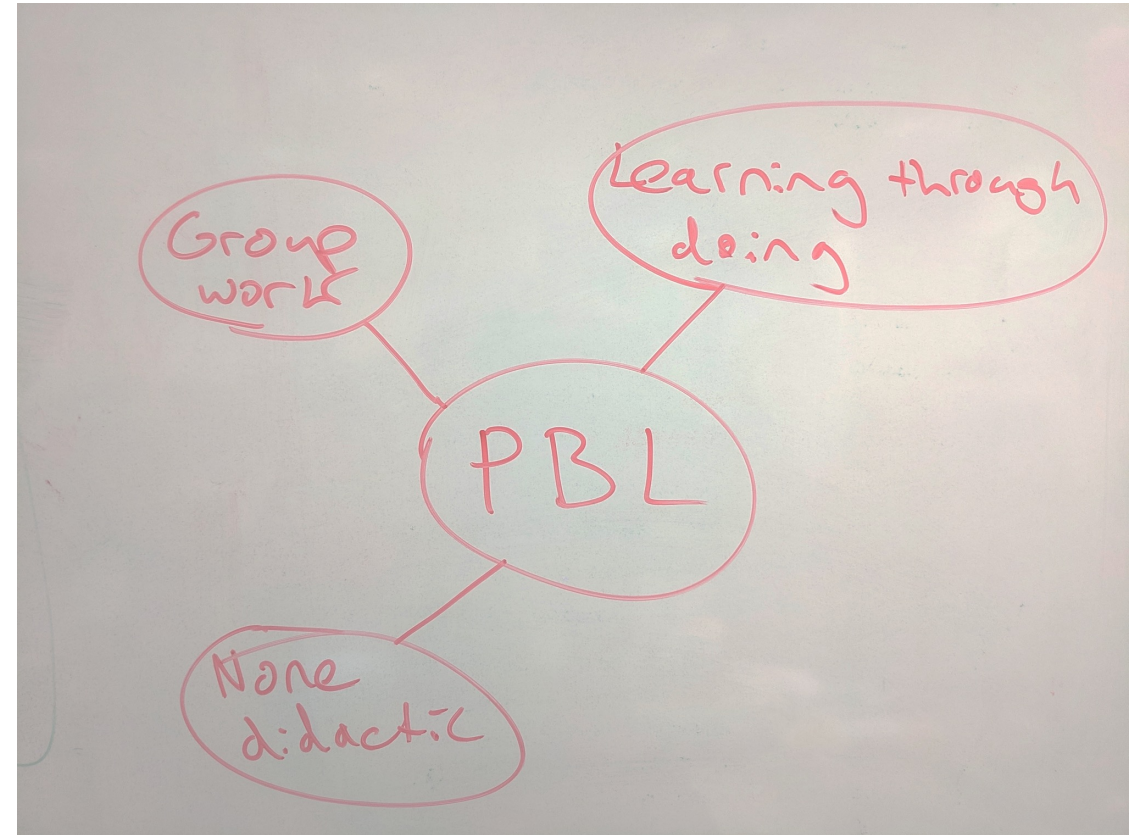
Hardware and technologies to enhance learning

Arm School Program

What is Project Based Learning?

Reflection points:

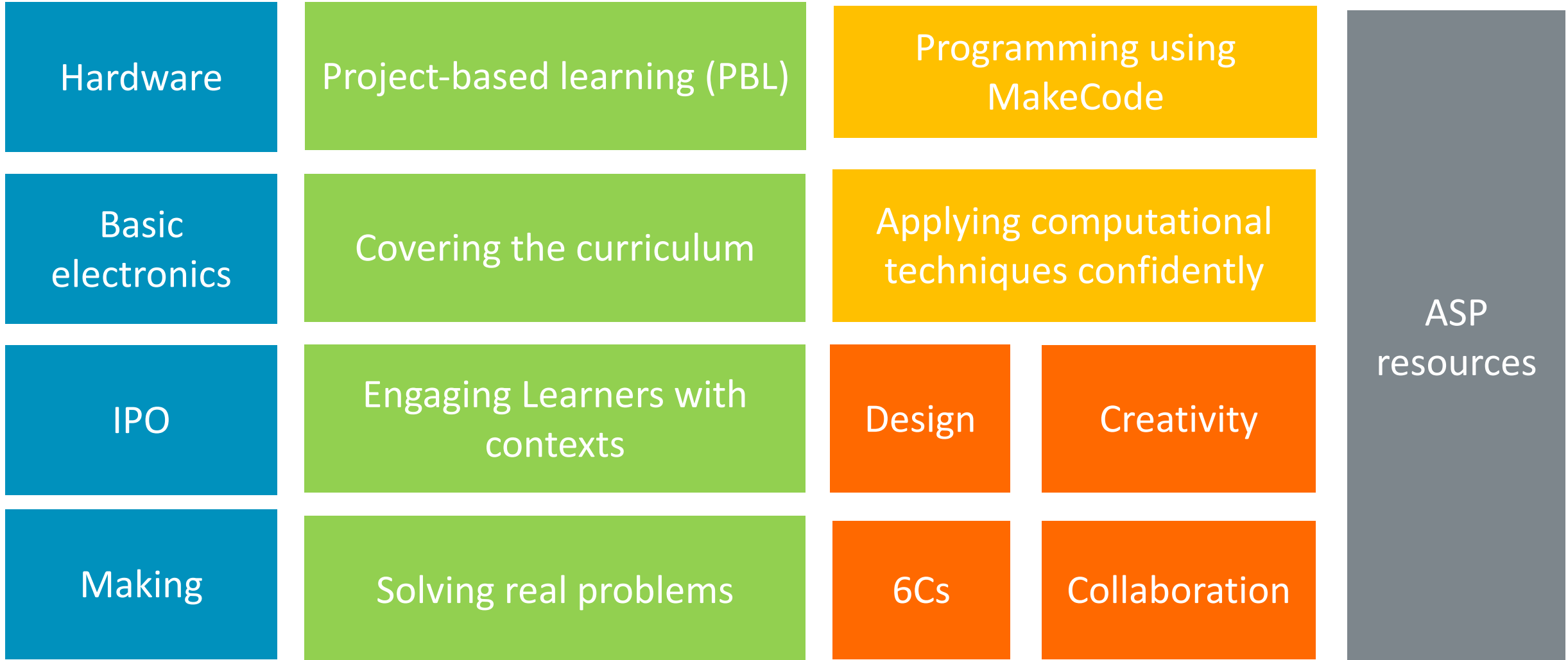
- What are the assumed negative connotations?
- Why is it effective?
- Why don't we do more of it?
- How can we use it to teach Computing?



Agenda

- What is Physical Computing/Project Based Learning?
- Hardware and interfaces
- Basic electronics needed
- The blocks
- The core computational elements to teach well
- Solving real problems
- Micro:pet
- Engaging learners
- Next steps

The core content



Reflection point

How confident are you in applying Physical Computing using Project Based Learning?

Defining physical computing

“Using programmable devices to build artifacts in order to teach Computing through problem solving by building confidence in applying computational techniques to solve real world problems”

Narrowing the definition of Physical Computing

Key concepts:

- A reaction to didactic teaching approaches (rooted in constructivism)

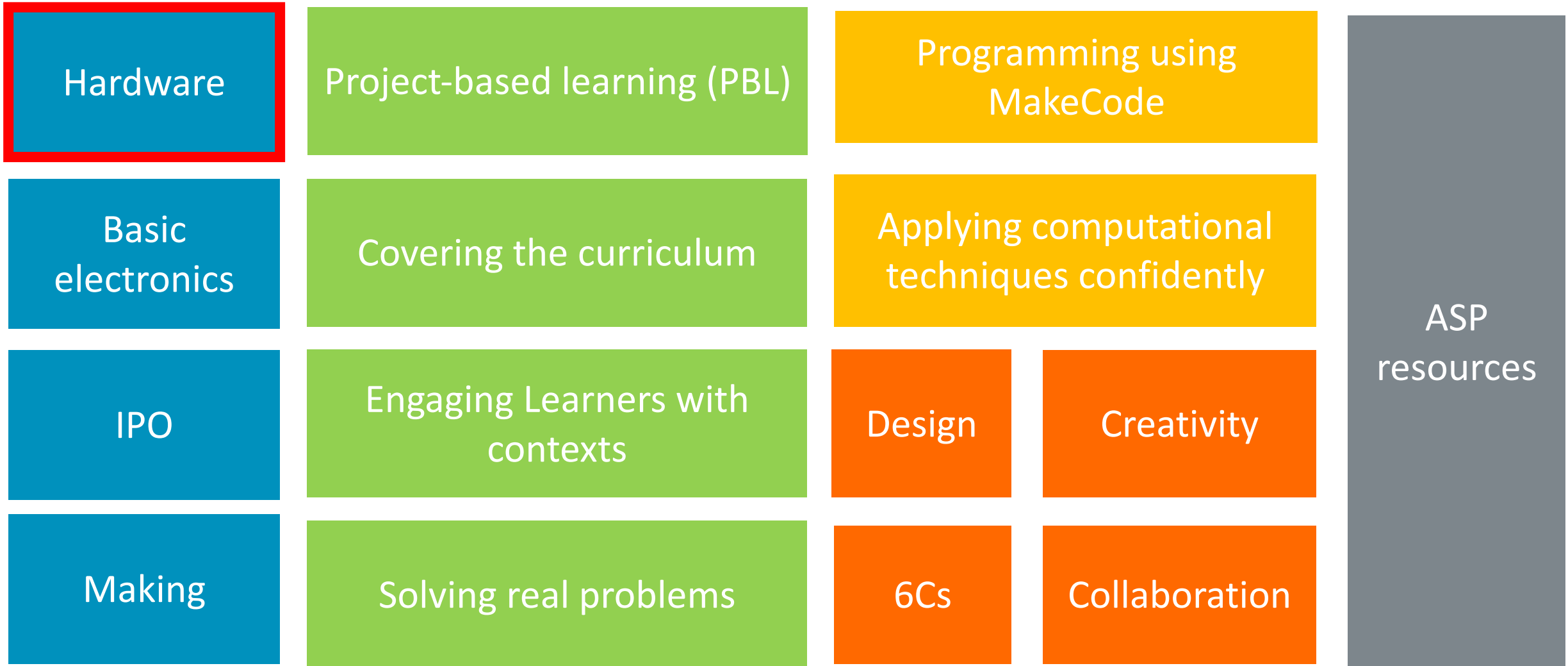
Limitations:

- Assumption that only exploration yields learning
- Not all learners will discover the intended learning
- Assumes a deep desire for learning and unlimited engagement

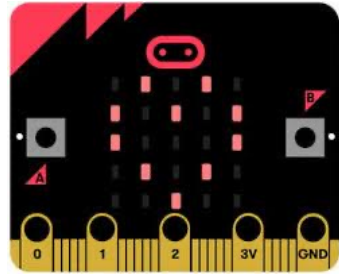
Response:

- Structured and contextualised curriculum linked projects
- Schema for all projects to guide learning
- Contextualised artifact creation in all projects

The core content

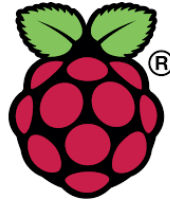


Arm based programmable devices



micro:bit

ARDUINO



Raspberry Pi



Resources for Schools (K-12)

Our teaching and learning resources help teachers deliver engaging and inspirational lessons in Computing using physical computing devices, such as the micro:bit. Applying the Arm School Program's project-based learning pedagogical approach, the resources encourage learners to develop soft STEM skills, such as creativity and resilience, while gaining the skills and knowledge needed for exam success.

To learn about the Arm School Program approach, download our brochure [here](#).



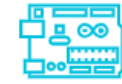
EdX: Teaching with Physical Computing
Professional development program for teaching with micro:bit, Arduino or Raspberry Pi through project-based learning.

[Learn More >](#)



Smart Schools on Arduino
Accessible and engaging projects based on the more advanced features of Arduino in real-world contexts. Ages 11-18.

[Learn More >](#)



Arduino Projects for Schools
A hands-on introduction to microcontrollers, the internet of things and data science. Ages 11-18.

[Learn More >](#)



Computing on micro:bit
Interactive activities and engaging projects with MakeCode as the programming interface. Ages 7-14.

[Learn More >](#)



Robotics and IoT
Projects use micro:bits and peripherals to create autonomous cars and smart cities. Ages 11-16.

[Learn More >](#)



Programming with MicroPython
Introduction to programming in MicroPython on a micro:bit. Ages 11-16.

[Learn More >](#)



Computational Thinking Tasks
Resources for the UK GCSE on computer science computational thinking, with interactive activities. Ages 14-16.

[Learn More >](#)



Micro:course
An introductory book of projects guiding learners from first plugging in their micro:bit to programming in Python. Suitable for all ages, and now available for micro:bit v2!

[Learn More >](#)



Computing for International Schools
A complete curriculum covering grades 5 to 7 based on a project-based learning approach using micro:bits and MicroPython. Ages 9-12.

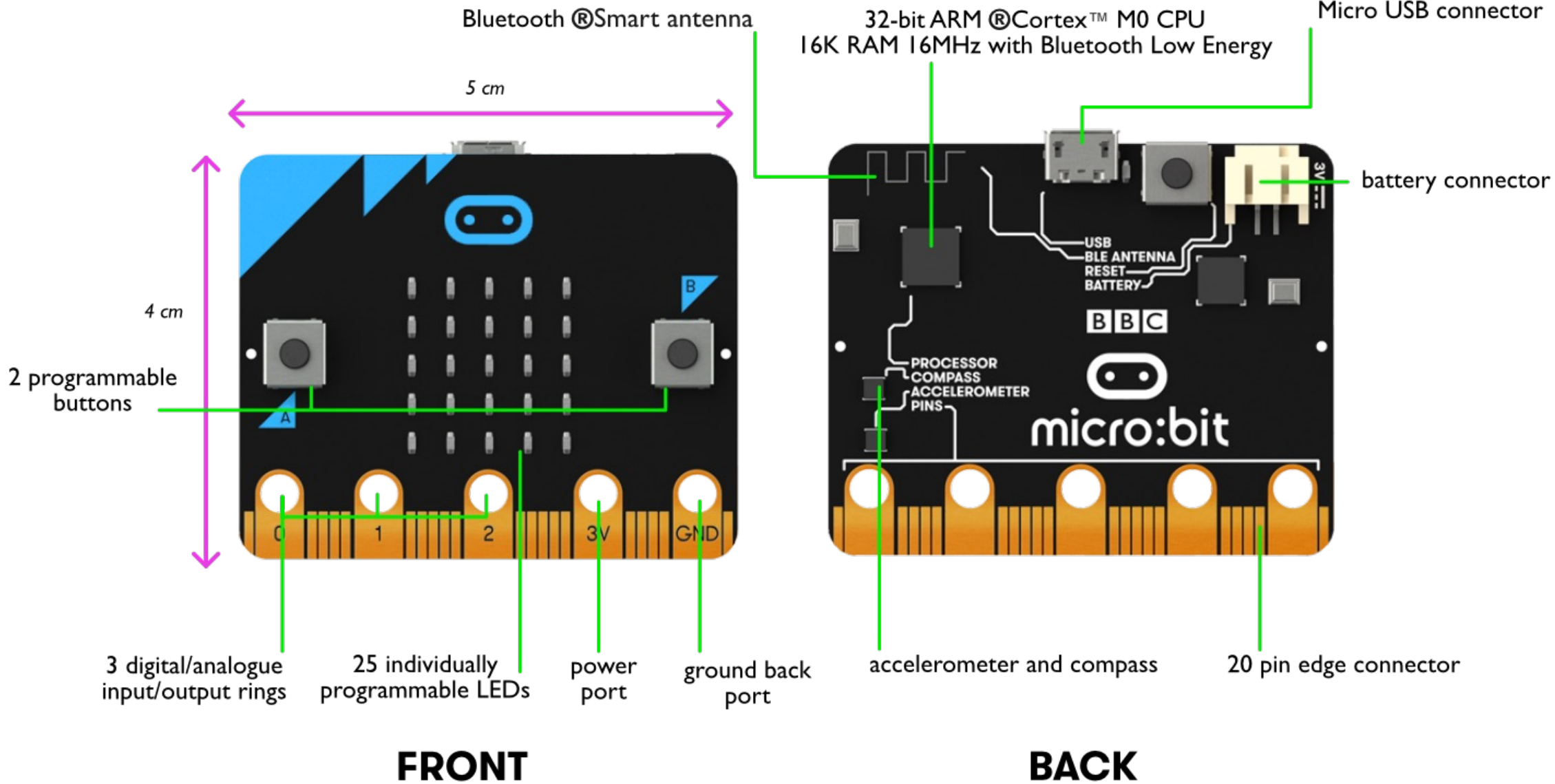
[Learn More >](#)



Raspberry Pi Pico Projects for Schools
Raspberry Pi Pico Projects for Schools: Explore cutting-edge topics in Computing, including ML and IoT. Ages 16-18.

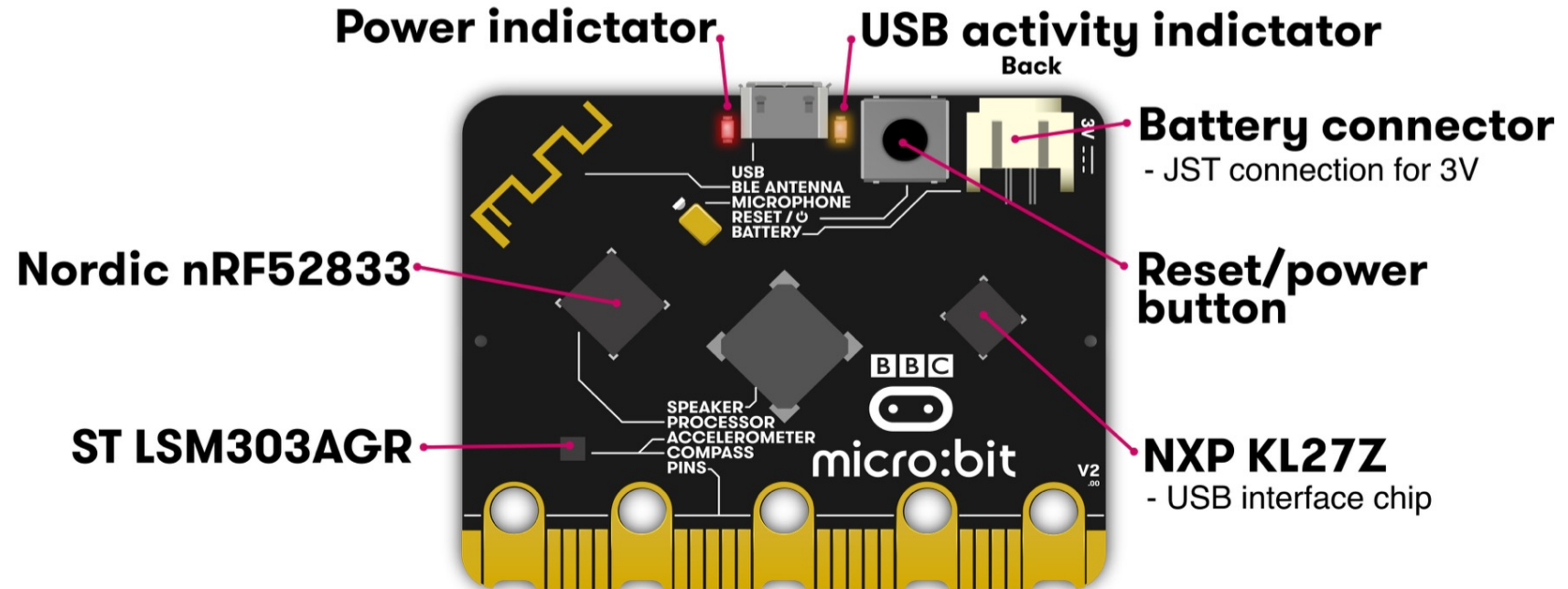
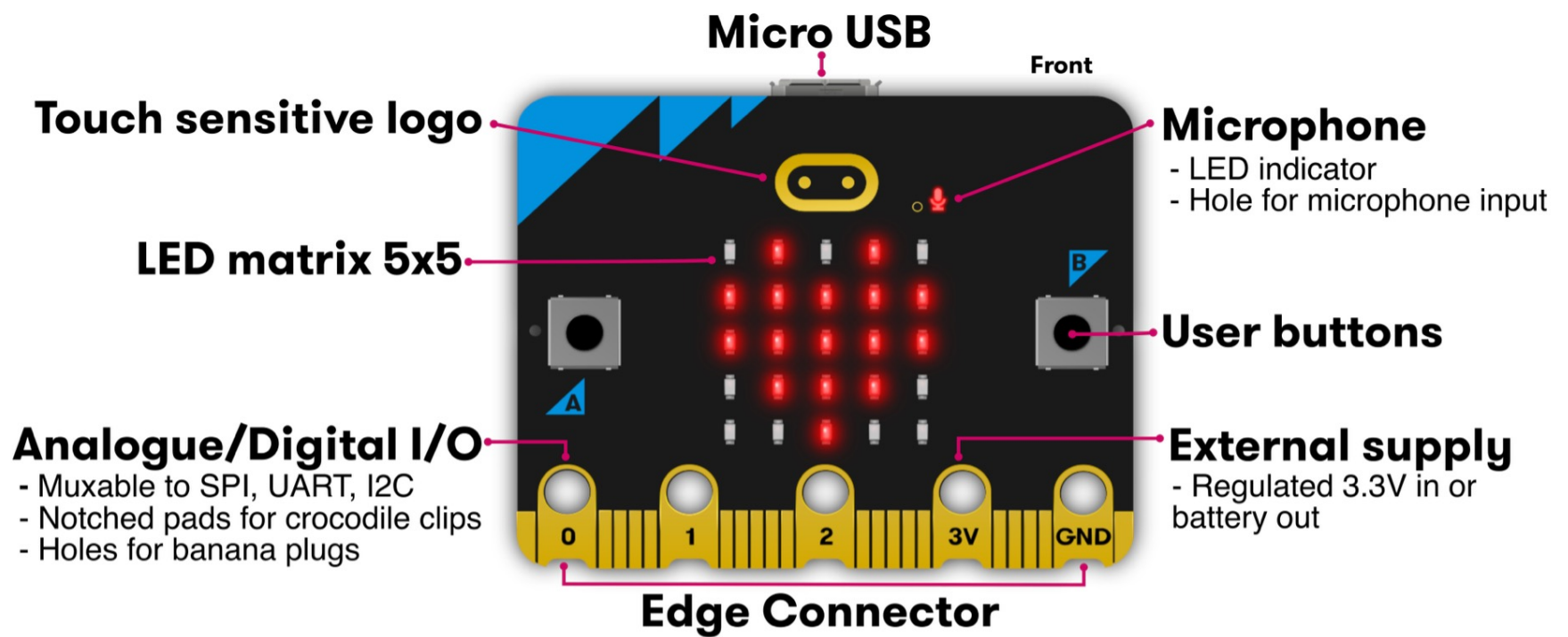
[Learn More >](#)

The micro:bit v1



The micro:bit v2

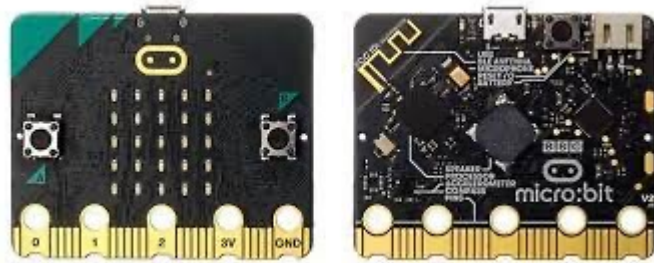
- Speaker
- Microphone
- Faster
- More memory
- Extra button on logo



Useful comparison

Micro:bit

- + Buttons
- + Capacitive touch (logo)
- + Screen (LEDs)
- + Bluetooth
- + Speaker
- + Microphone
- + Other sensors
- + USB



Smartphone

- + Buttons
- + Capacitive touch screen
- + Screen (LEDs)
- + Bluetooth
- + Speaker
- + Microphone
- + Other sensors
- + USB



Practical point

The micro:bit needs power from somewhere

Battery pack

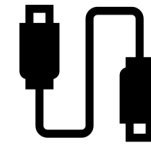


Power from USB



Things to consider:

- You plug your micro:bit into your PC using the USB cable (Macs will need a USB C adapter)
 - USB ports on PCs will need to be working – speak to your technician beforehand
 - The make:code website will need to be allowed
- USB cables
 - Default ones are short – you may need longer ones depending on your PCs
- Batteries
 - Often DOA – have spares or use re-chargeable ones



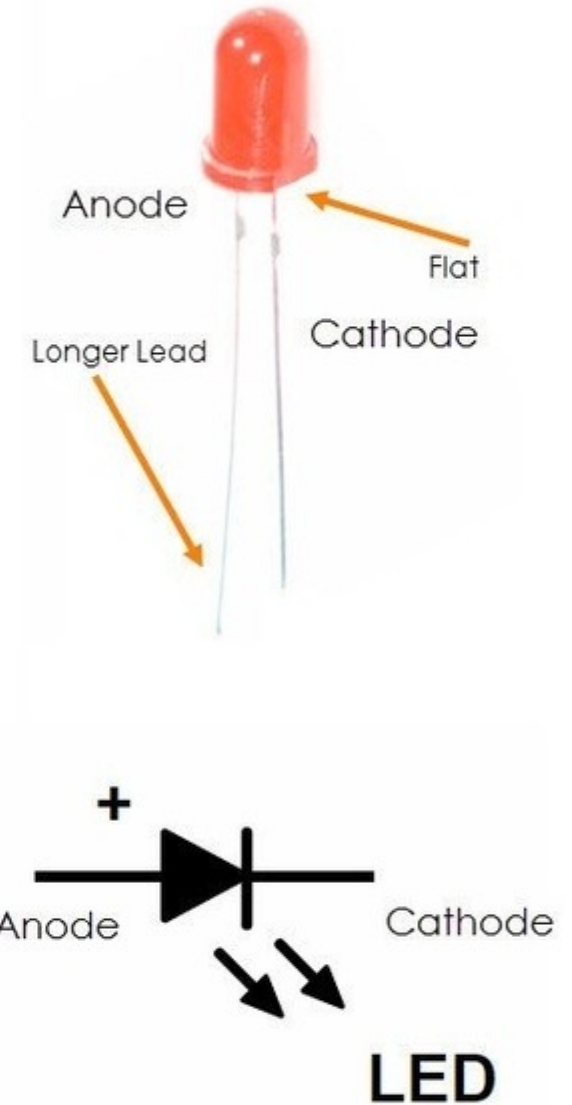
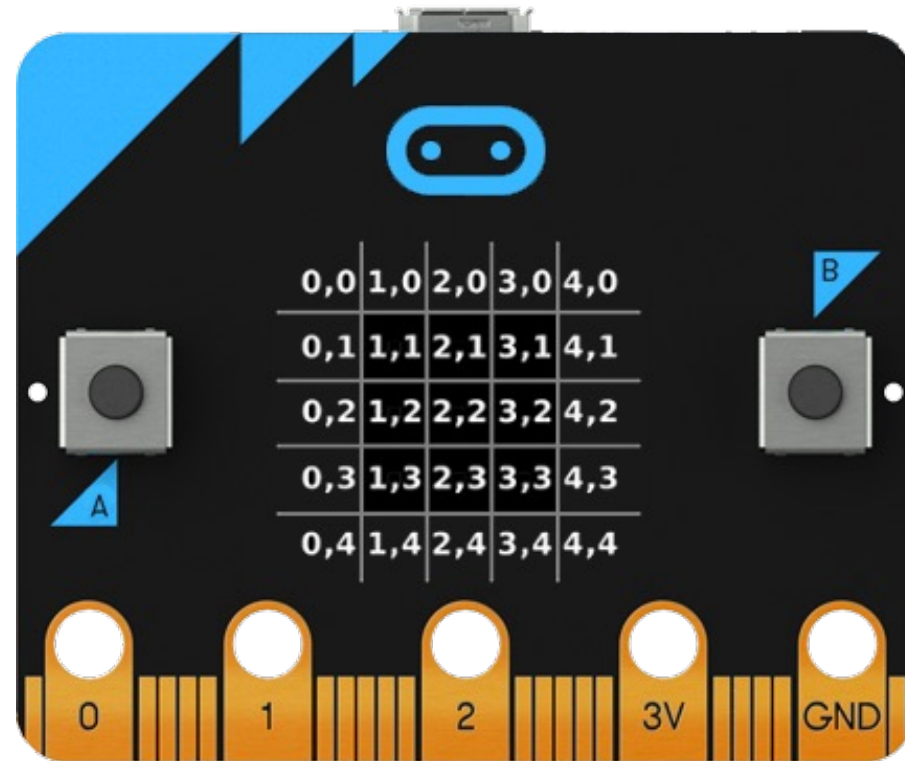
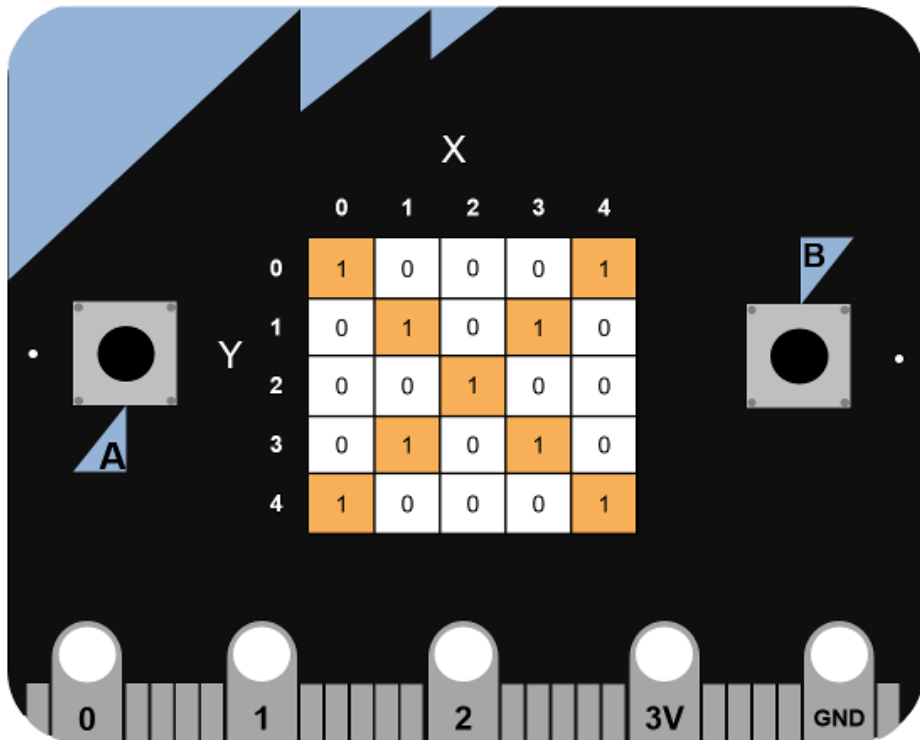
Calibrating the micro:bit

New micro:bits need calibrating when they are first used. Follow the instructions on the micro:bit:

- Press the buttons
- Shake it (vigorously)
- Tilt it to move the LED to the other LED (like snake)

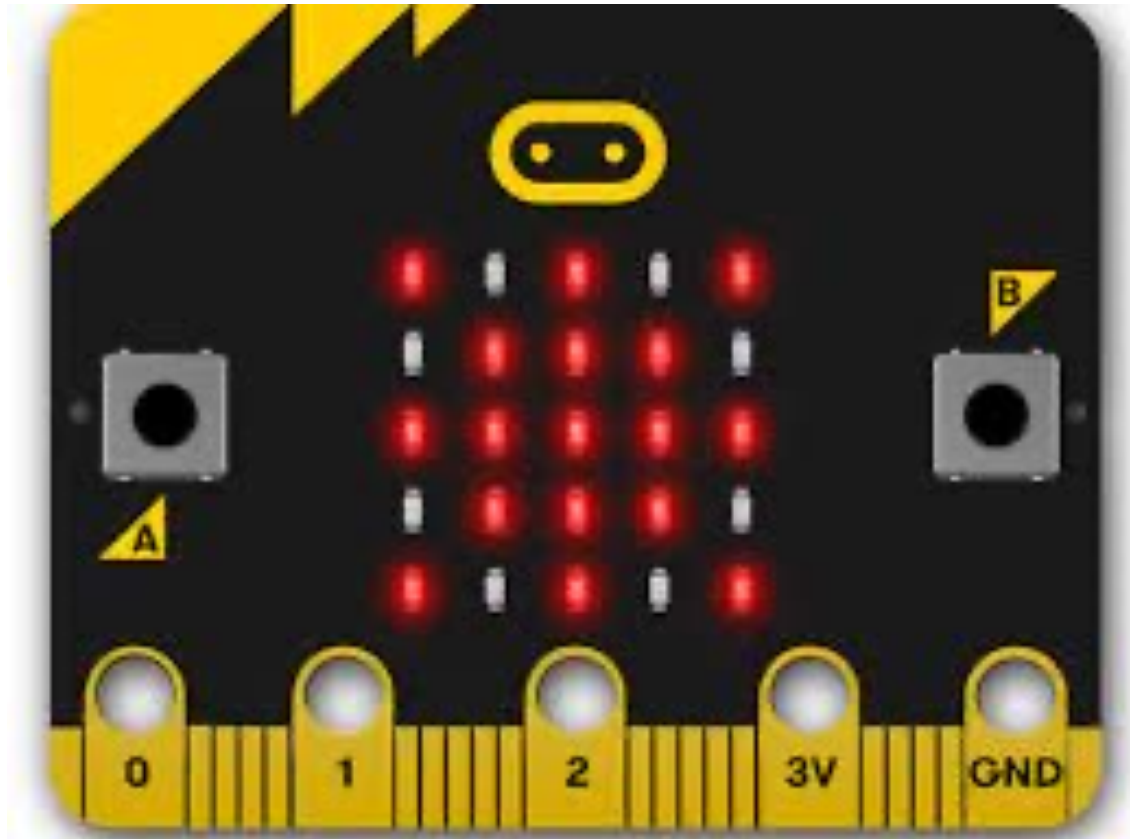
It's a good idea to calibrate the compass before using it in lessons, this can be a starter activity for example. Doing so ensures accuracy.

Light Emitting Diodes – 5x5 matrix

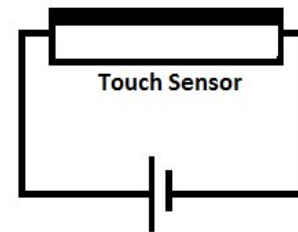


Also the light sensor (10 levels – via algorithm)

The buttons

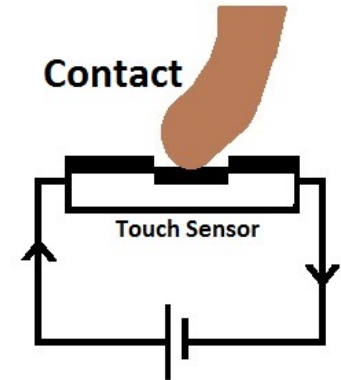


No contact



No Flow of current

Contact



Flow of current

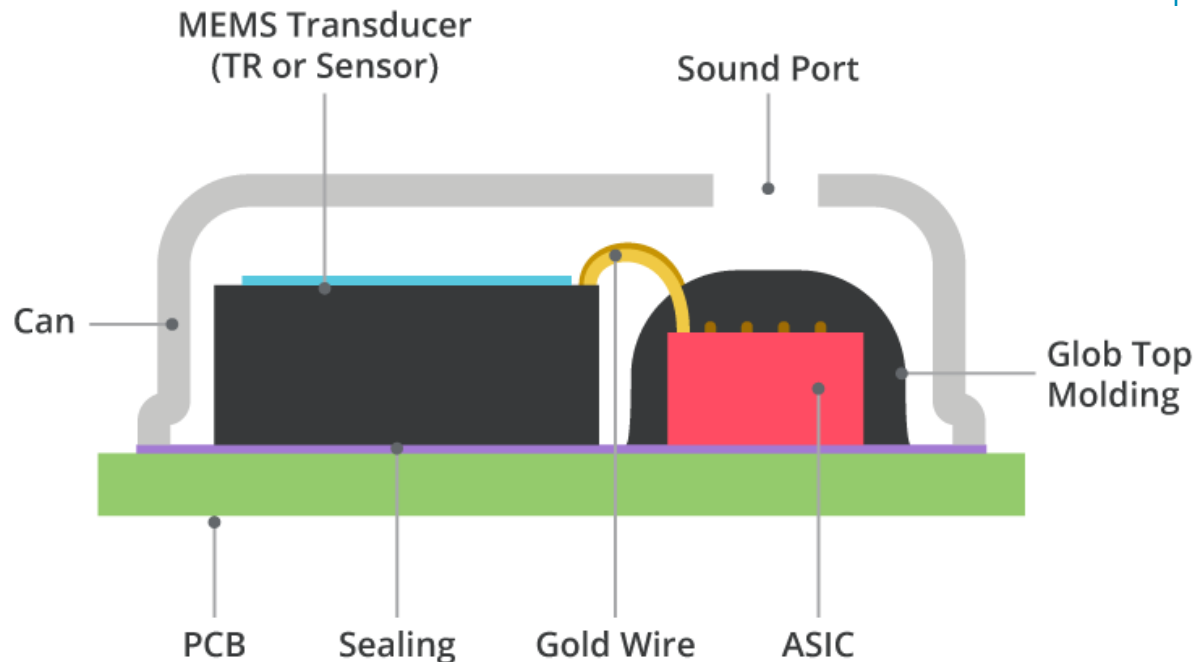
New speaker and mic (V2)

MEMS microphone

- + (micro-electromechanical systems) mic is a pressure-sensitive diaphragm
- + Input also via mic input pin

Speaker

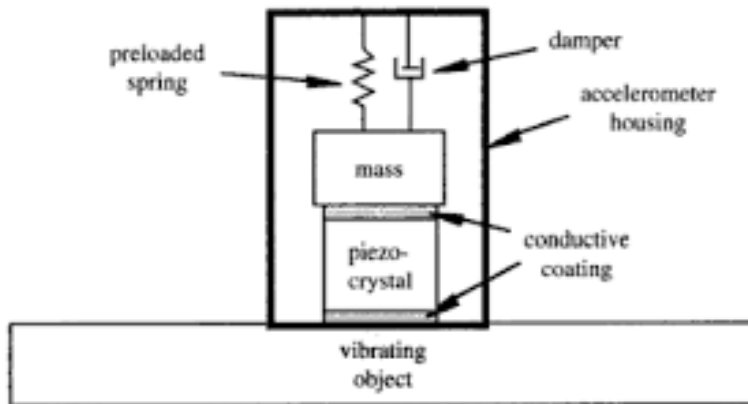
- + Magnetic speaker
- + Sound output via PWM on pins also



Other sensors

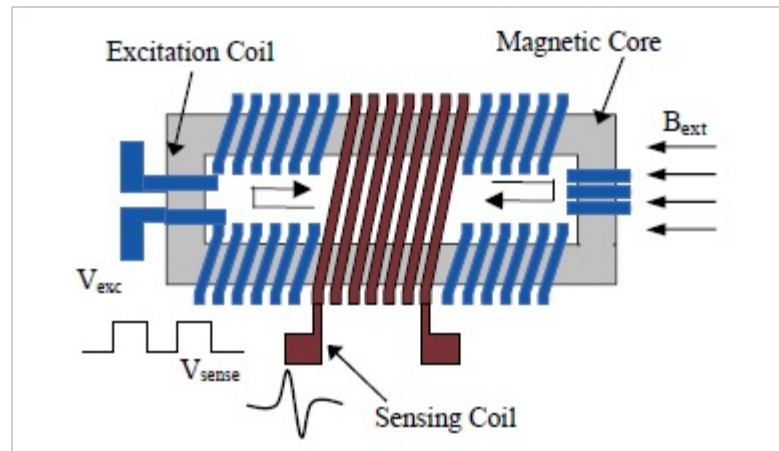
Accelerometer

- + Senses G_s (acceleration)



Magnetometer (compass)

- + Spring and magnets
- + Hall sensor gives digital signal

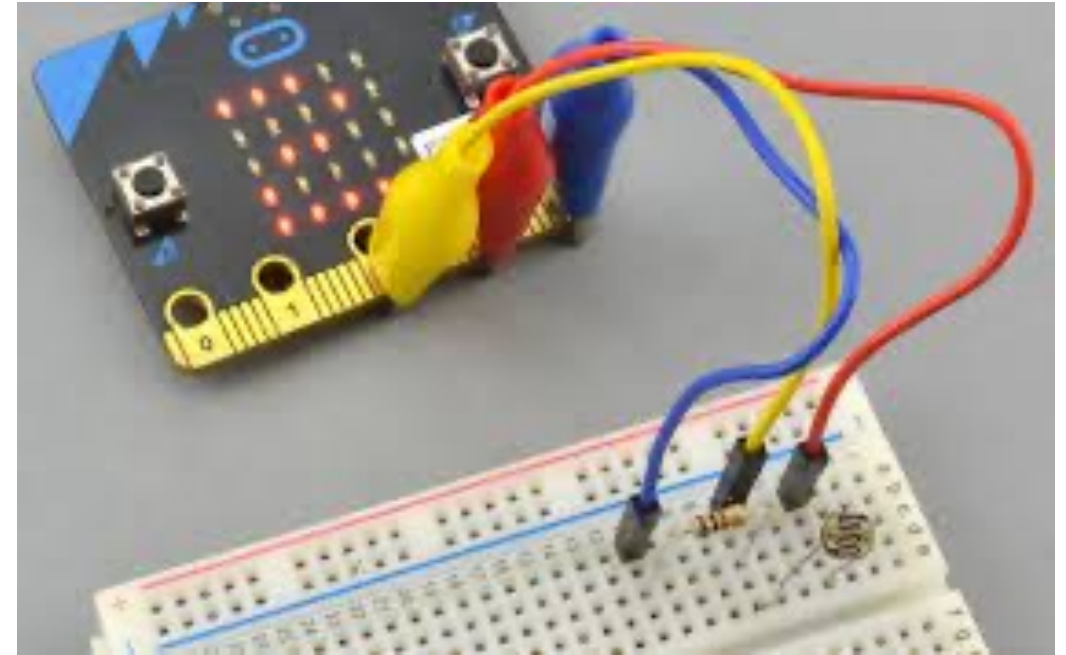
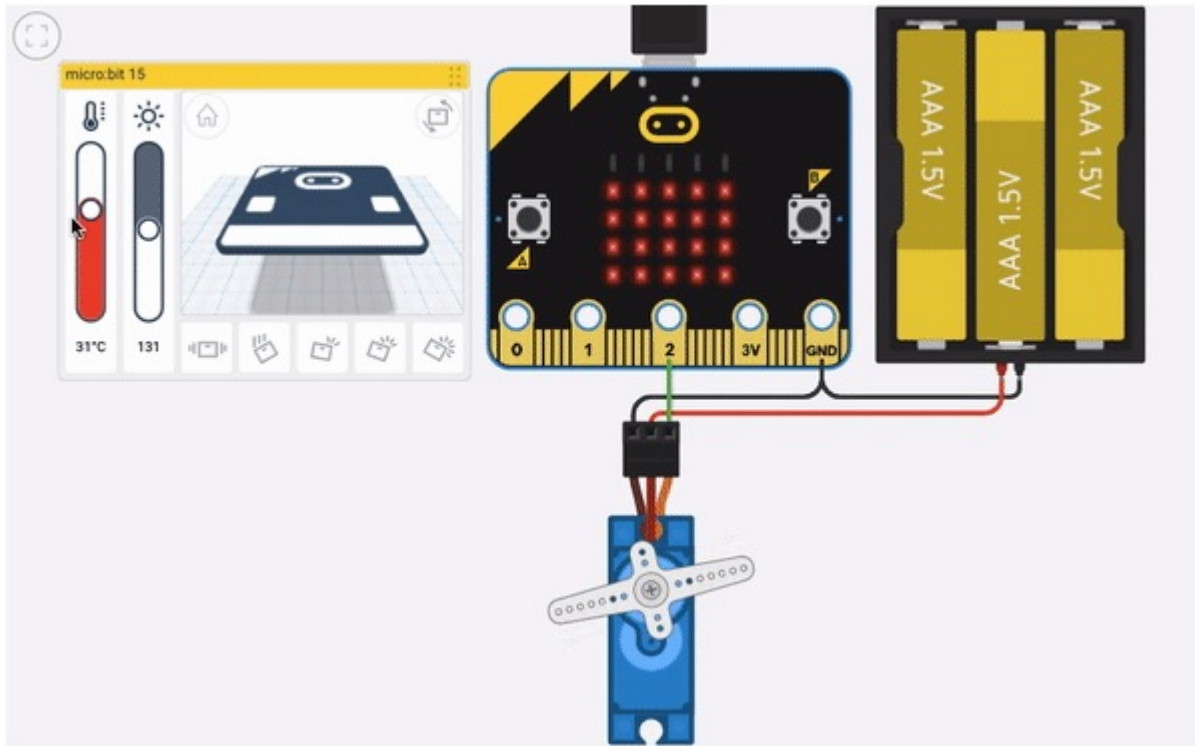


Temperature (on CPU)

- + Resistance changes at different temps.

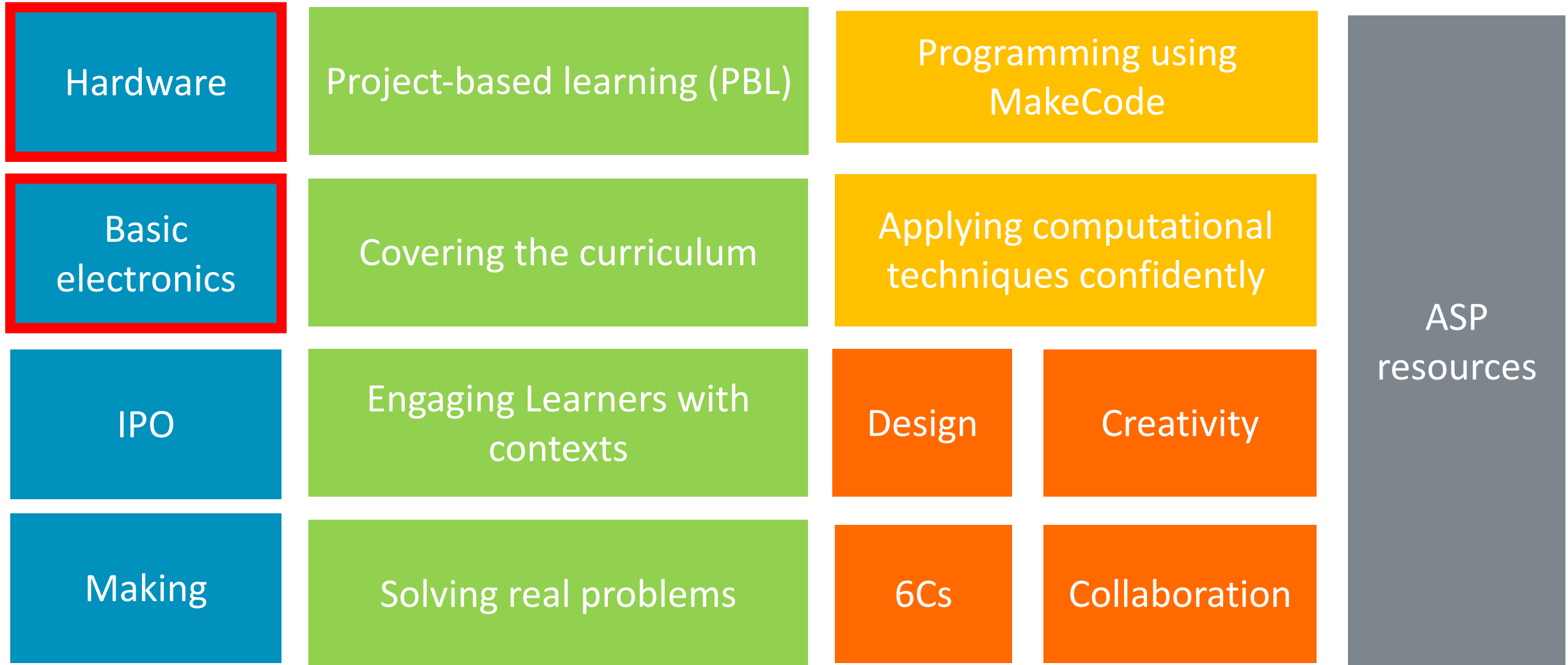


Circuits with a micro:bit – real and simulated



+ <https://www.tinkercad.com/blog/explore-microbit-with-tinkercad>

The core content



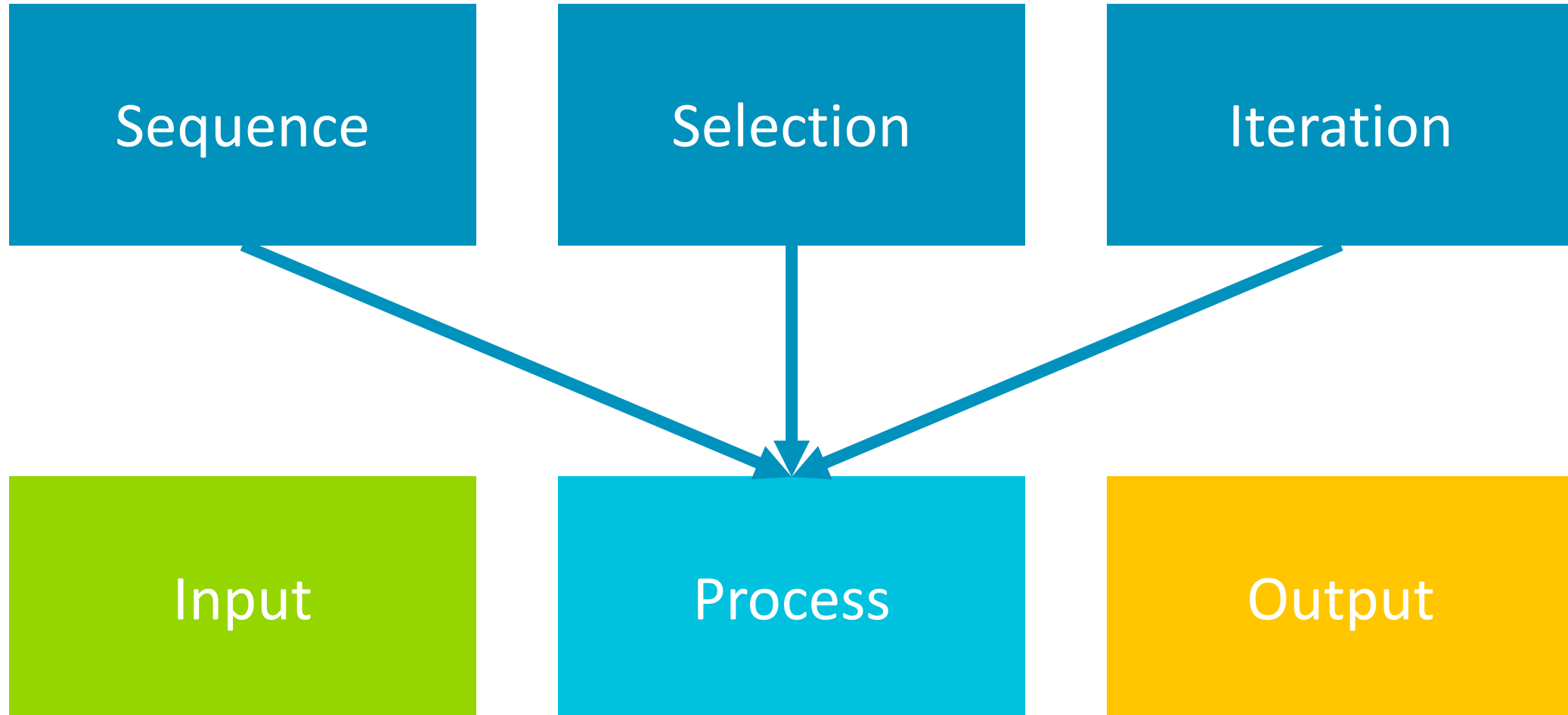
The Input Process Output (IPO) Model

- All computer systems take data into a system using 'Inputs', carry out processes on the inputs and then display the result of that processing using 'Outputs'
- We can use **Input, Process, Output** worksheets to identify what the outputs will be

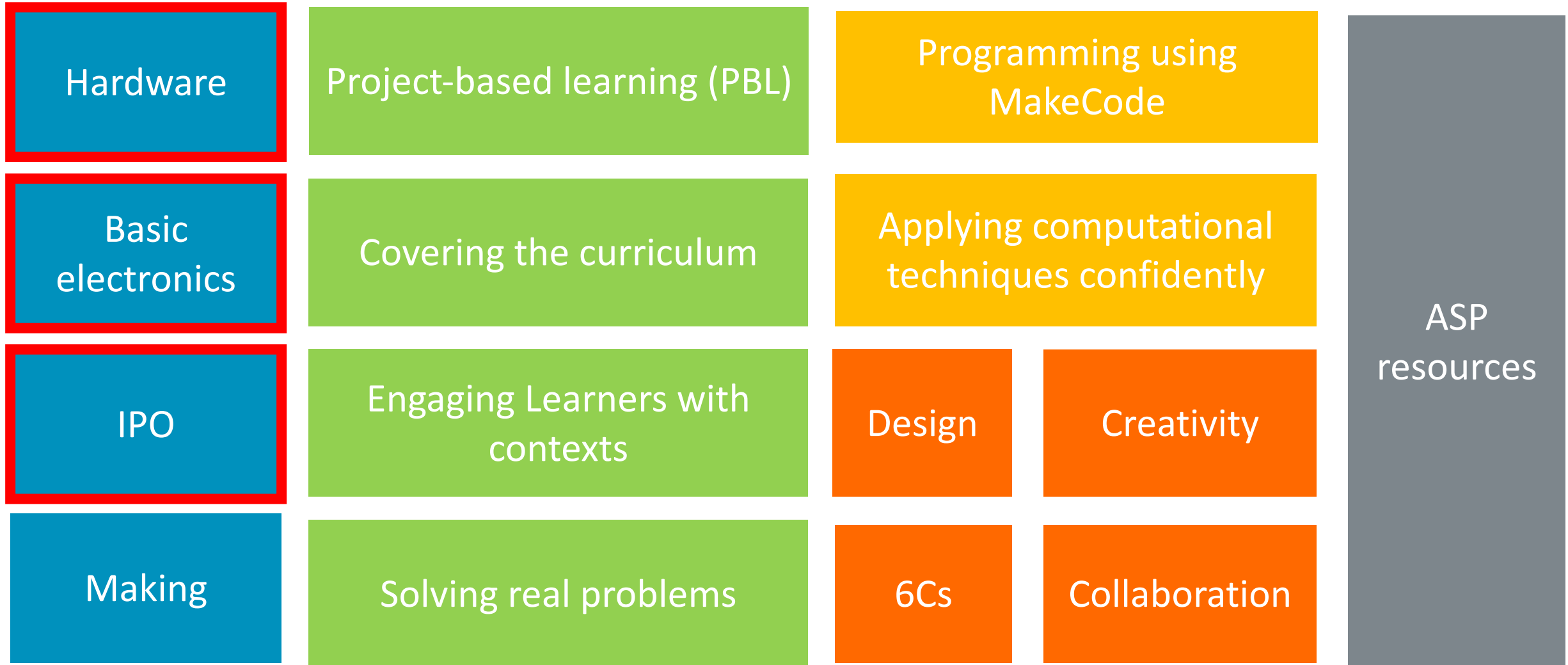


Basic Programming Constructs

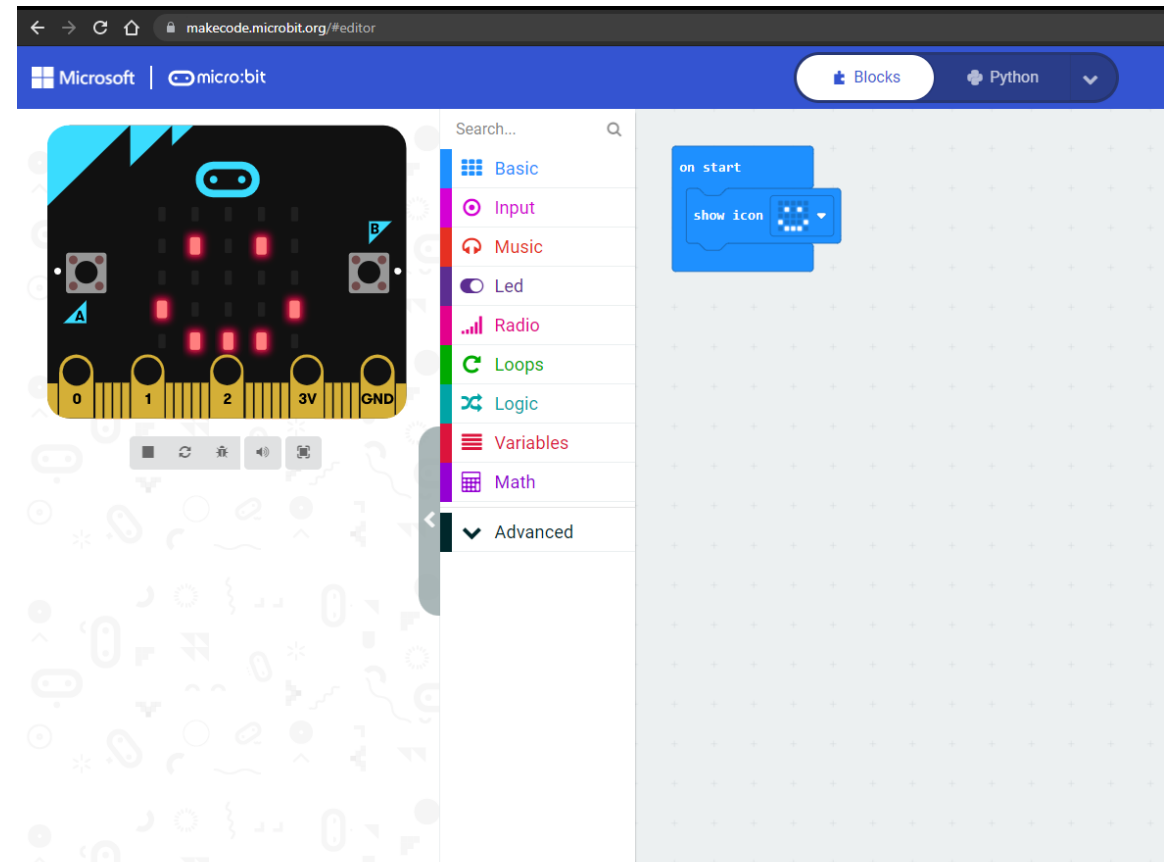
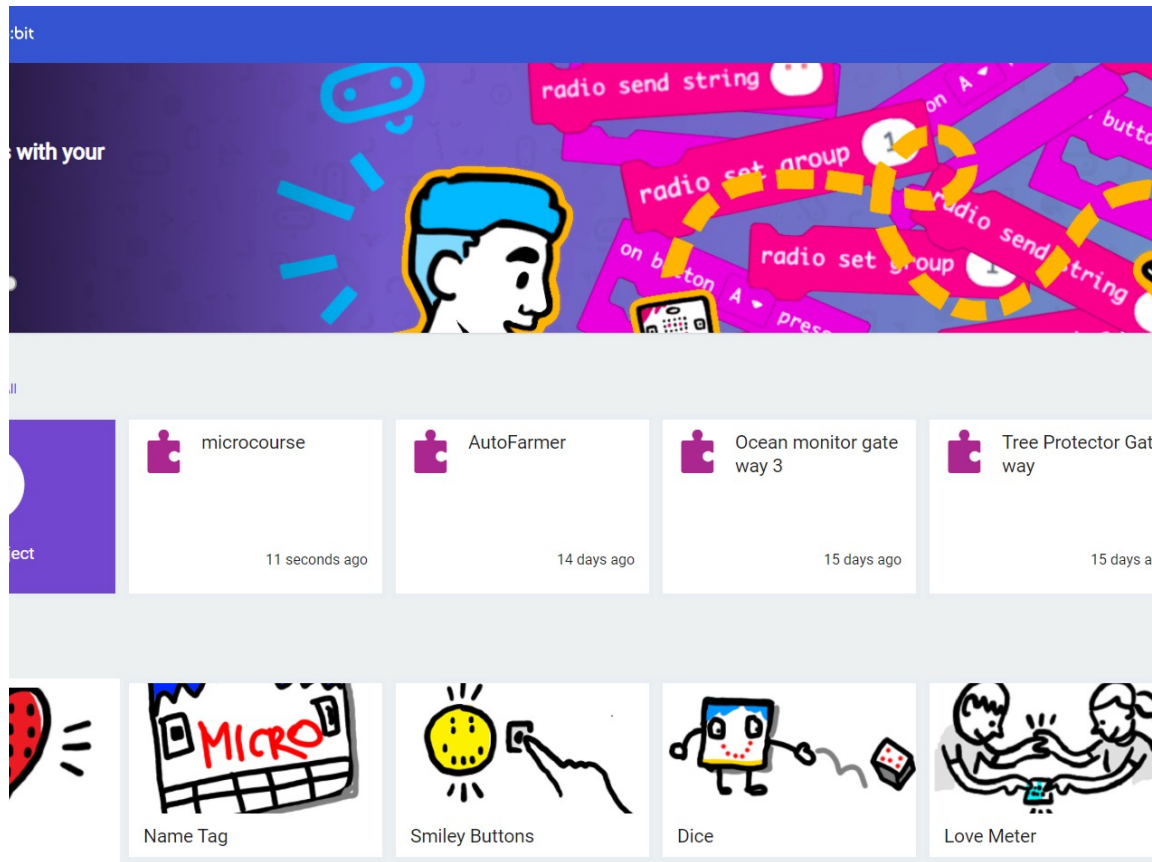
- There are three main programming constructs:



The core content

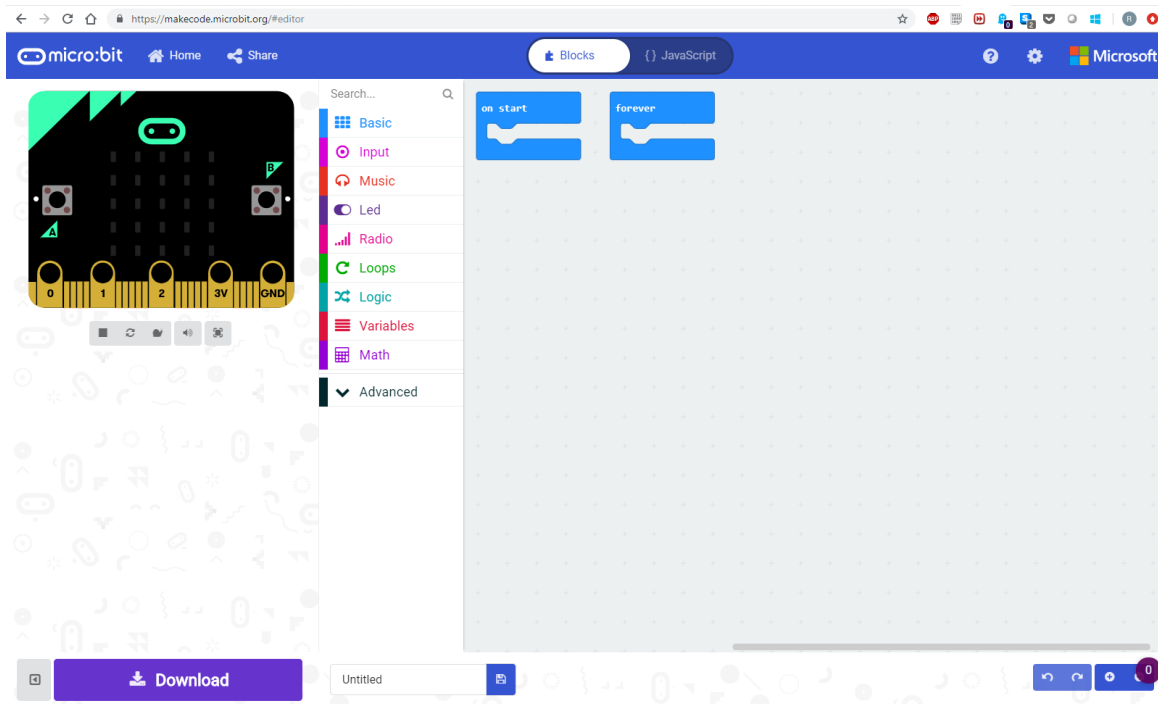


makecode - <https://makecode.microbit.org/>



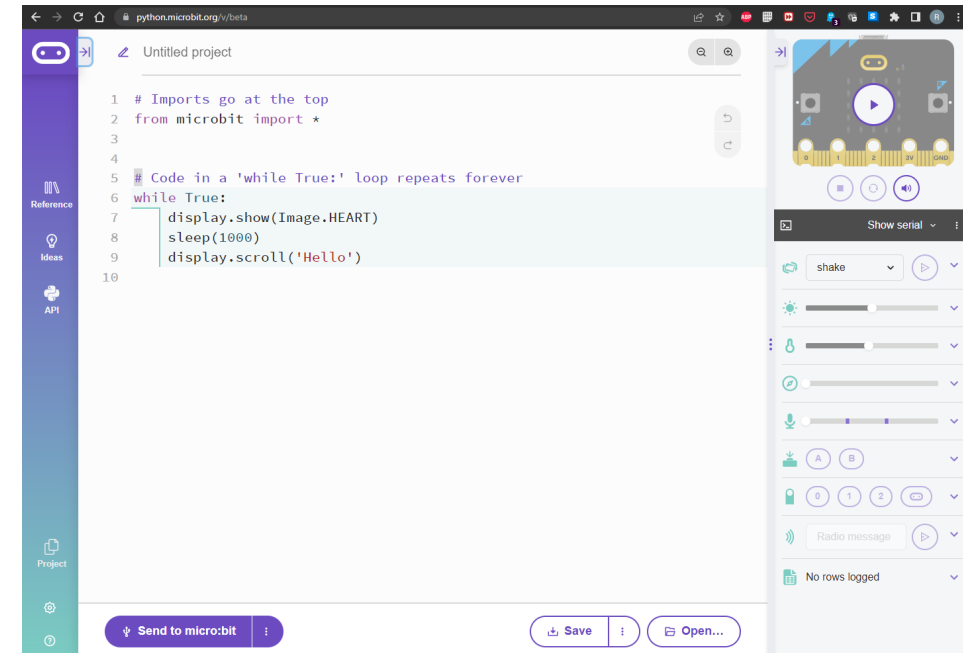
Makecode and MicroPython

Block based



<https://makecode.microbit.org/#editor>

Text based (MicroPython)












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

Programming a micro:bit



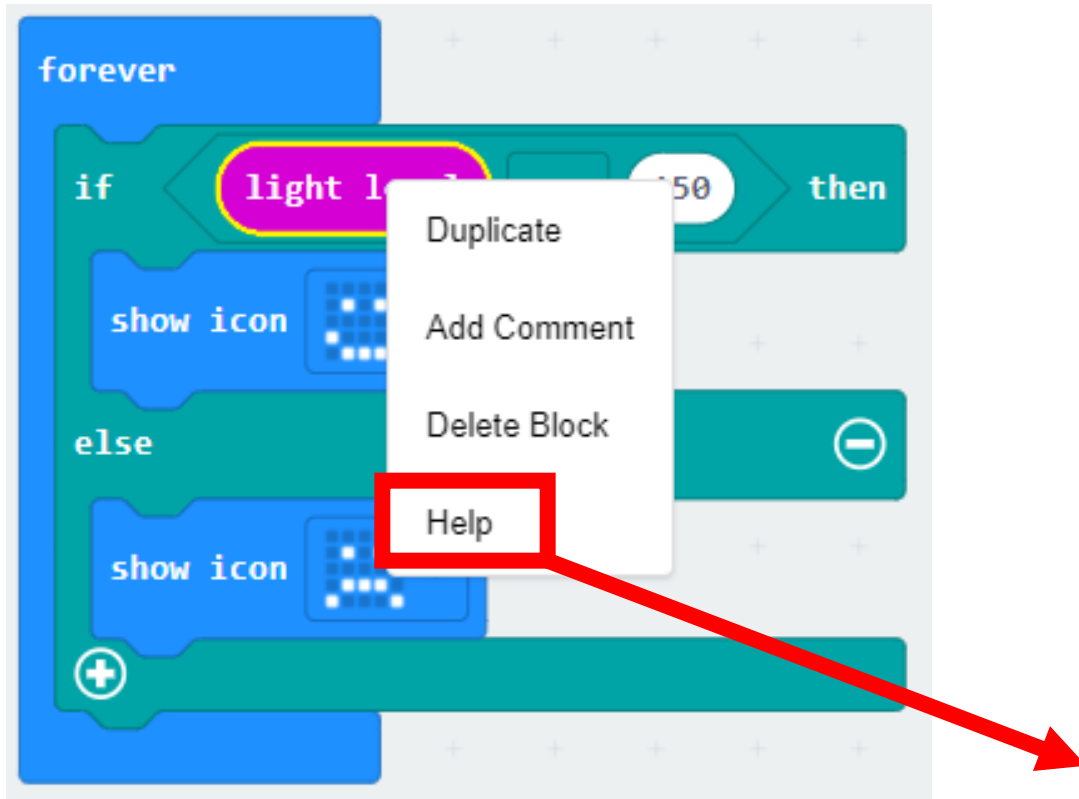
- Save to micro:bit
- Or
- Save and then copy to micro:bit

The blocks and IPO

 Basic
 Input
 Music
 Led
 Radio
 Loops
 Logic
 Variables
 Math

- **Output** and some flow control
- **Input** sensors, buttons
- **Output**, speaker
- **Output**, fine LED control
- **Input/Output** – between micro:bits
- **Process** - Iteration - sequence
- **Process** - Selection – if x do y
- **Process** – contain data to be re-used/changed
- **Process** - Math – used in loops and logic to compute things

Getting help

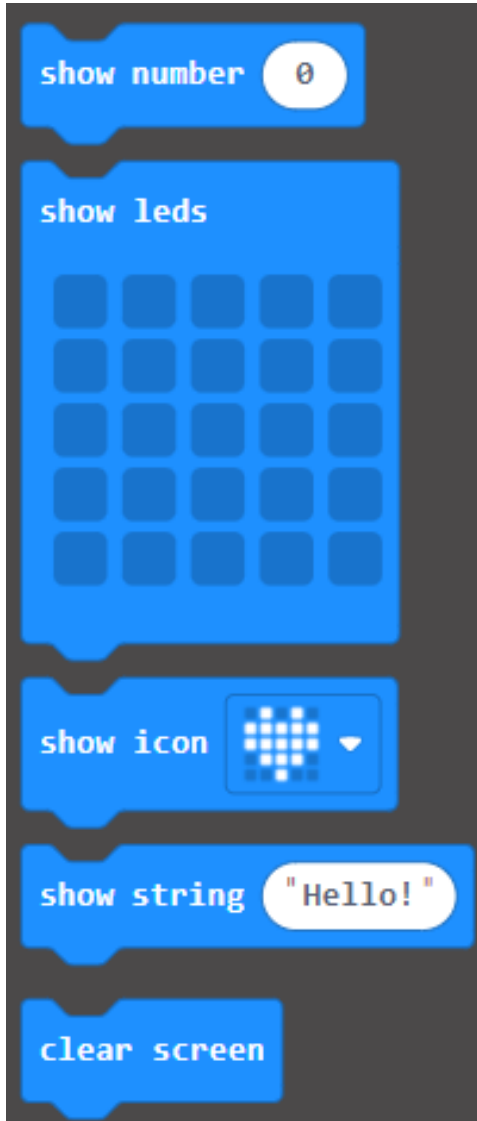


- You can get help for every block that explains what it does
- It does not however unpick your logic or suggest how to apply logic
- Teach learners to be resilient by using this tool – literacy demand

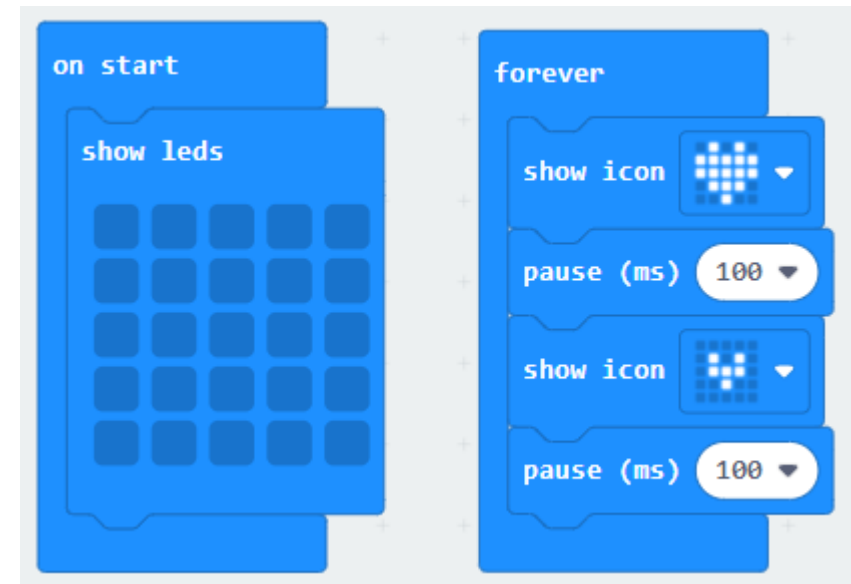
Light Level

Find the light level (how bright or dark it is) where you are. The light level 0 means darkness and 255 means bright light. The micro:bit measures the light around it by using some of the LEDs on the [LED screen](#).

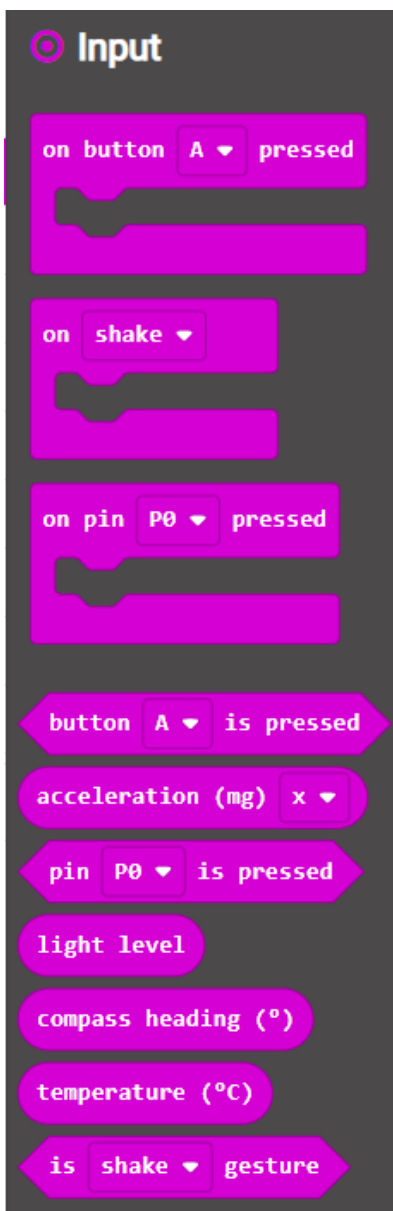
The basic blocks



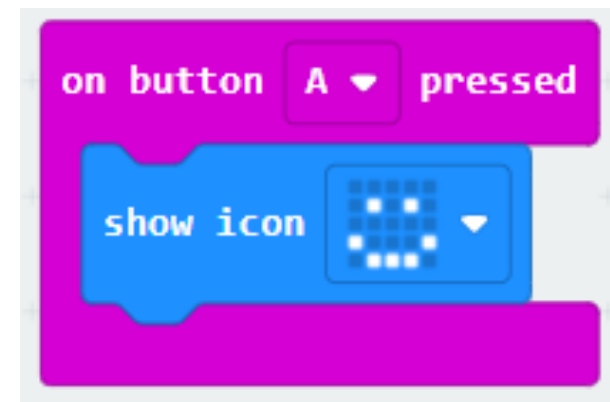
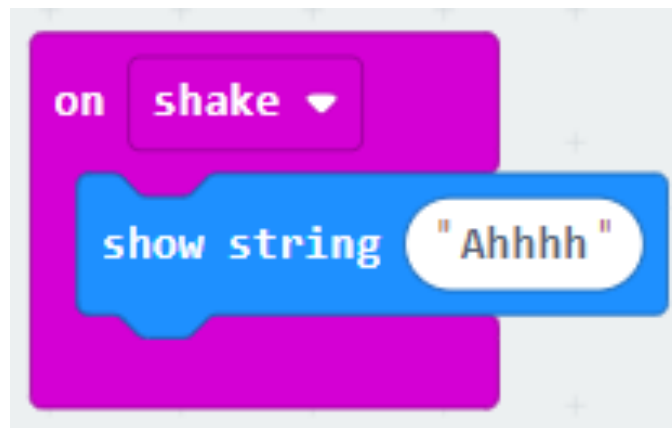
- Best place to start
- The most used blocks



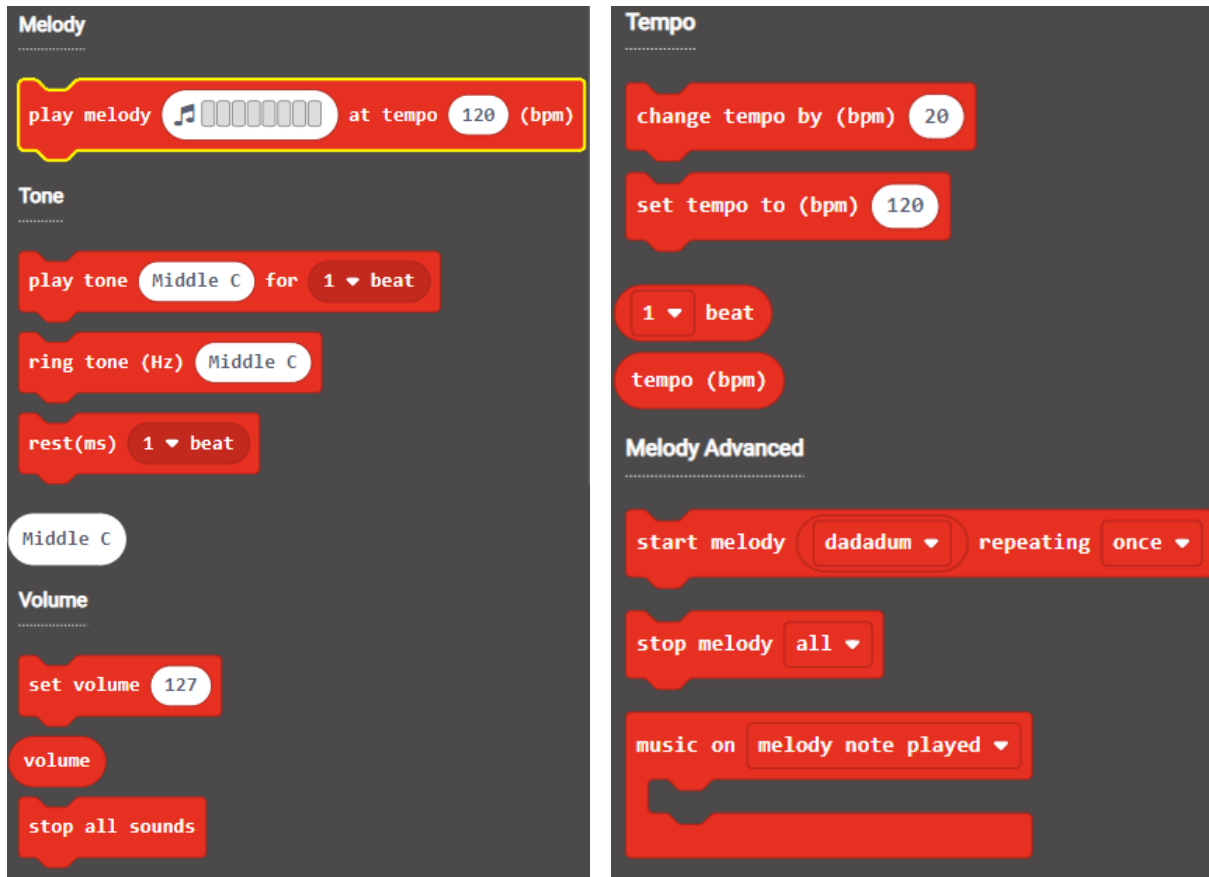
Input blocks



- Key blocks for interaction and PBL generally
- Takes in real world input/data
- Lots of different types of input to experiment with
 - Light sensor
 - Heat sensor
 - Accelerometer
 - Magnetometer
 - Buttons
 - Mic (v2)



Music blocks



The screenshot shows the Music blocks library in Scratch, organized into several categories:

- Melody:** Includes a 'play melody' block with a musical staff icon and a tempo input field set to 120 (bpm).
- Tone:** Includes 'play tone' (with 'Middle C' and '1 beat' inputs), 'ring tone (Hz)' (with 'Middle C' input), and 'rest(ms)' (with '1 beat' input).
- Volume:** Includes 'set volume' (with '127' input), a 'volume' block, and 'stop all sounds'.
- Melody Advanced:** Includes 'start melody' (with 'dadadum' and 'repeating once' inputs), 'stop melody' (with 'all' input), and 'music on' (with 'melody note played' input).

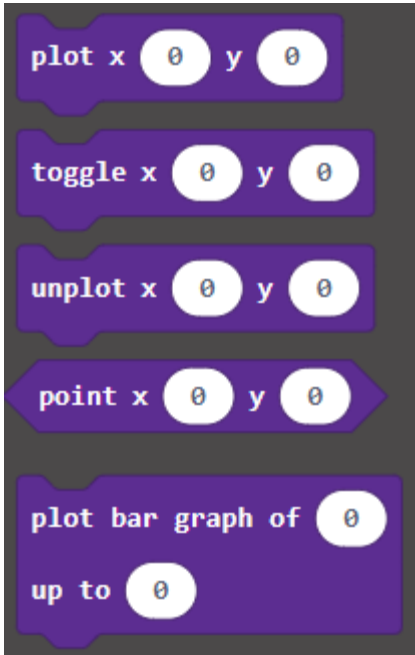
- Great for cross-curricular projects
- Combine with loops for more complex musical algorithms
- Needs headphones for V1



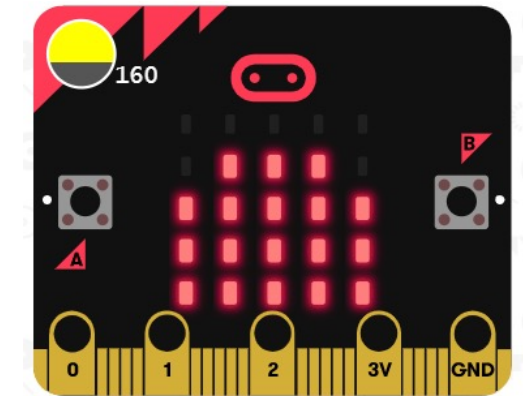
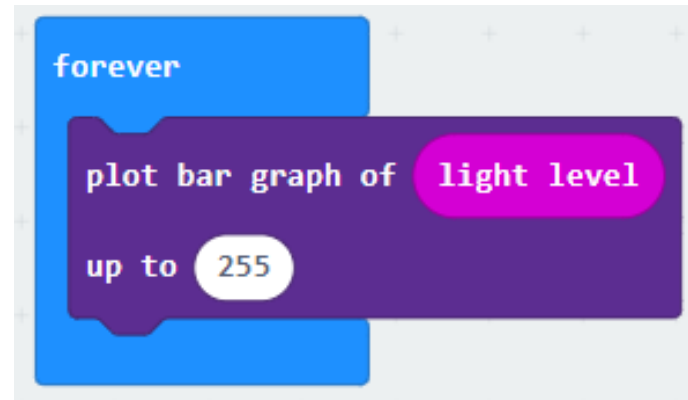
The screenshot shows a Scratch script with the following blocks:

- on start:** A blue block containing a red 'set volume' block with the value 127.
- on button A pressed:** A purple block containing a red 'start melody' block with 'dadadum' and 'repeating once' inputs.
- on shake:** A purple block containing a red 'play melody' block with a musical staff icon and a tempo input field set to 120 (bpm).

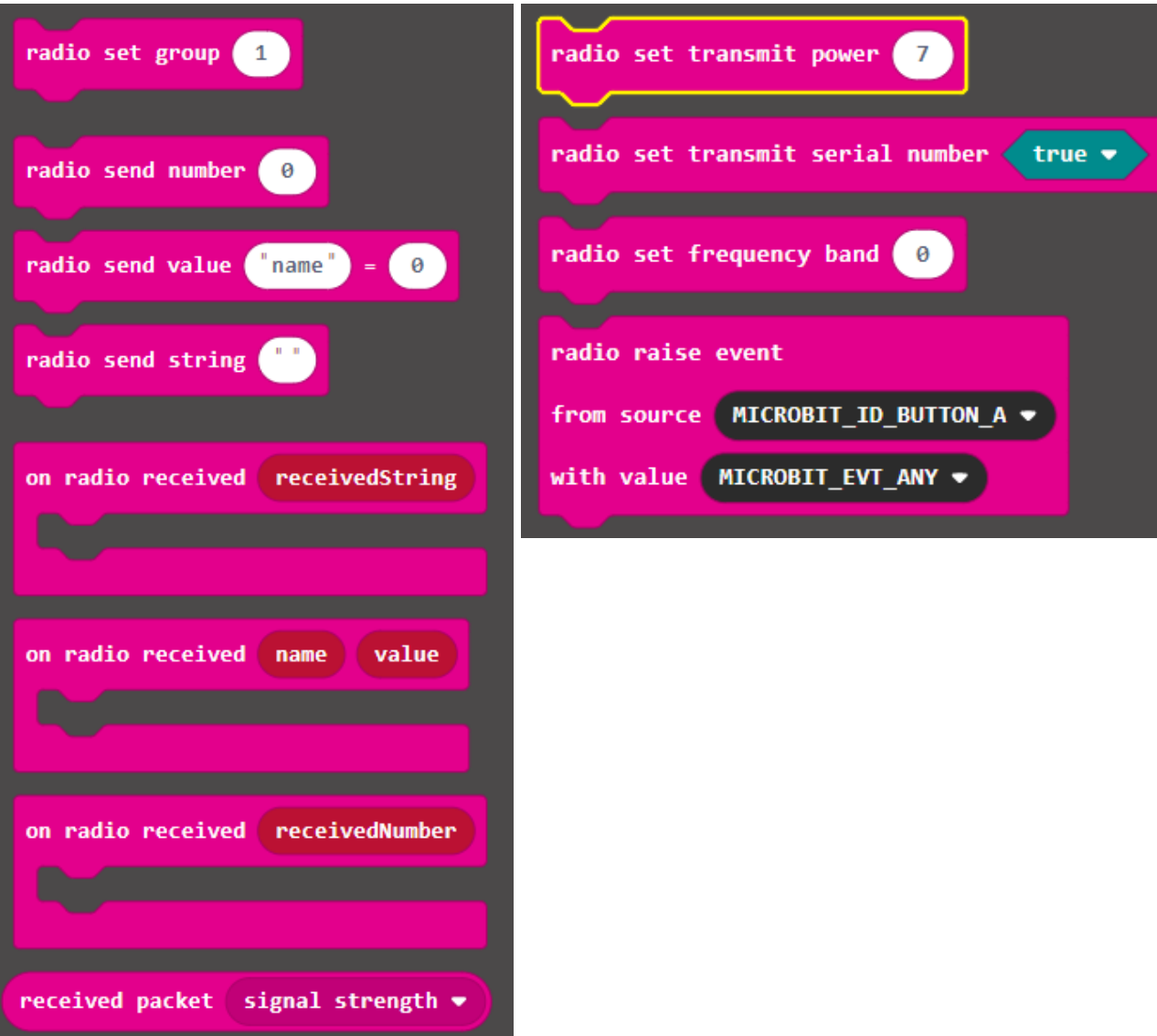
LED



- Gives fine control over the 5x5 LED matrix
- Allows plotting of data from inputs
- Can change brightness
- Great for UX and interactions



Radio



The image shows a Scratch Radio block palette with two columns of blocks. The left column contains blocks for setting the radio group, sending data, and receiving data. The right column contains blocks for setting transmit power, serial number, frequency band, and raising events. The 'radio set transmit power' block is highlighted with a yellow border.

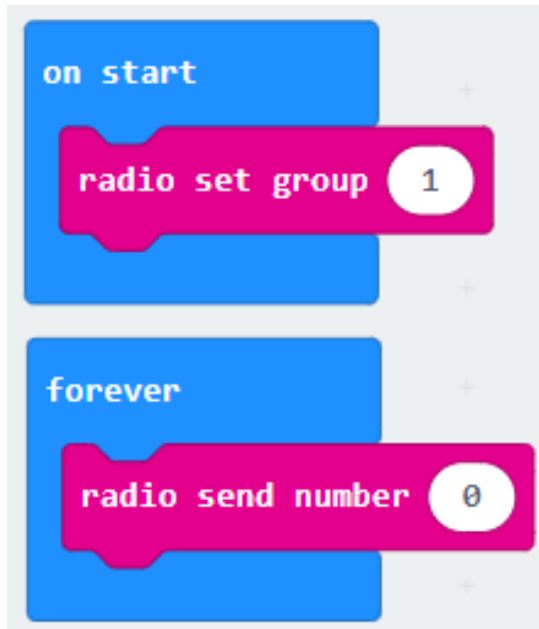
```
radio set group 1
radio send number 0
radio send value "name" = 0
radio send string ""
on radio received receivedString
on radio received name value
on radio received receivedNumber
received packet signal strength

radio set transmit power 7
radio set transmit serial number true
radio set frequency band 0
radio raise event
  from source MICROBIT_ID_BUTTON_A
  with value MICROBIT_EVT_ANY
```

- Allows transmission of data between micro:bits
- Must set to same radio 'group' on each micro:bit
- Can send number, value and string
- Can change signal strength
- Use to trigger events on other micro:bits
- Caution when using lots in a class at once as they can interfere with each other easily

Triggering events on another micro:bit

Micro:bit 1

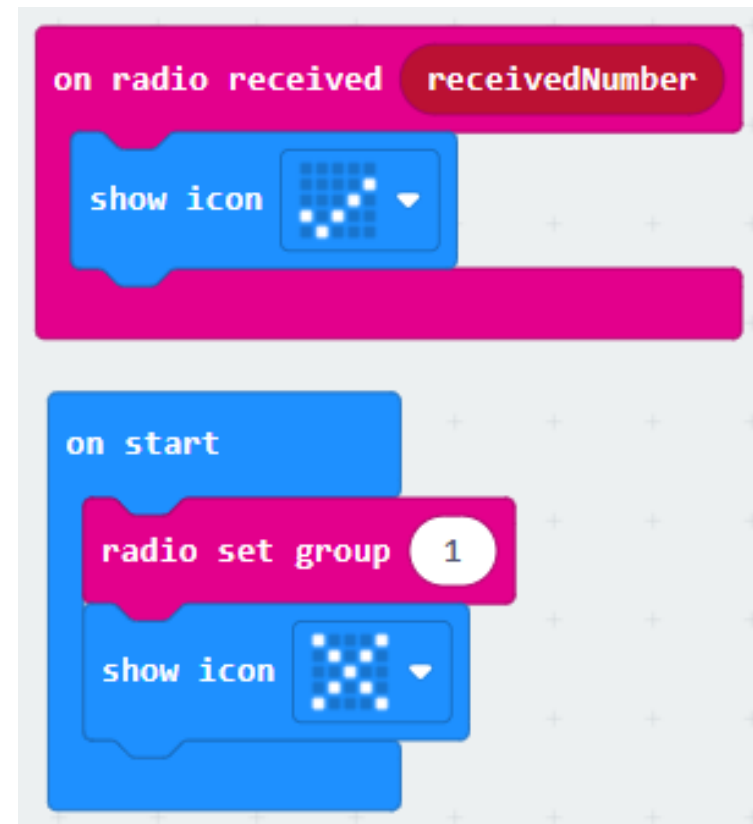


```
on start
  radio set group 1

forever
  radio send number 0
```

The code for Micro:bit 1 consists of two main blocks. The first is an 'on start' block (blue) containing a 'radio set group' block (pink) with the value '1'. The second is a 'forever' loop block (blue) containing a 'radio send number' block (pink) with the value '0'.

Micro:bit 2

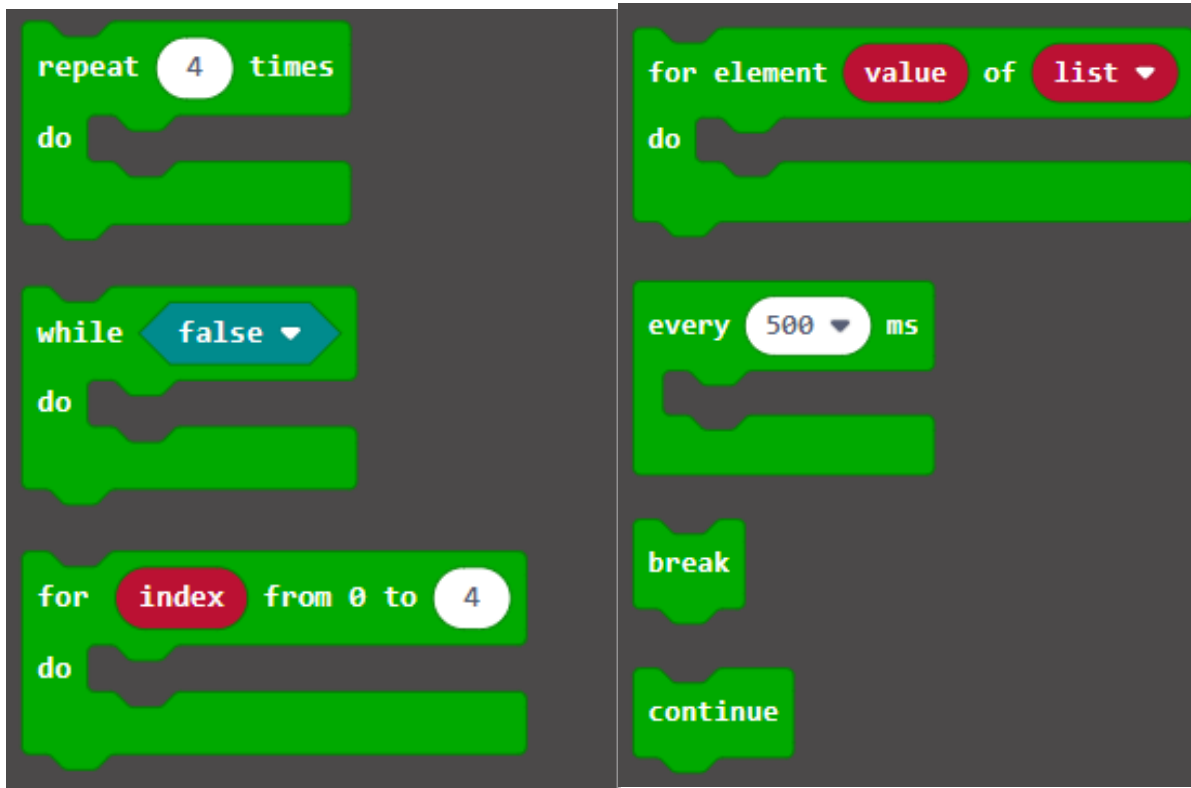


```
on radio received receivedNumber
  show icon [grid icon]

on start
  radio set group 1
  show icon [grid icon]
```

The code for Micro:bit 2 consists of two main blocks. The first is an 'on radio received' block (pink) with the parameter 'receivedNumber', containing a 'show icon' block (blue) with the 'grid icon' selected. The second is an 'on start' block (blue) containing a 'radio set group' block (pink) with the value '1' and a 'show icon' block (blue) with the 'grid icon' selected.

Loops



- Used for iteration, or looping through other blocks based on a condition
- Repeat, FOR and WHILE loops
- Arguably the most complex to use



Logic

Conditionals

if true ▾ then

+

if true ▾ then

else -

+

Comparison

0 = ▾ 0

0 < ▾ 0

" " = ▾ " "

Boolean

and ▾

or ▾

not

true ▾

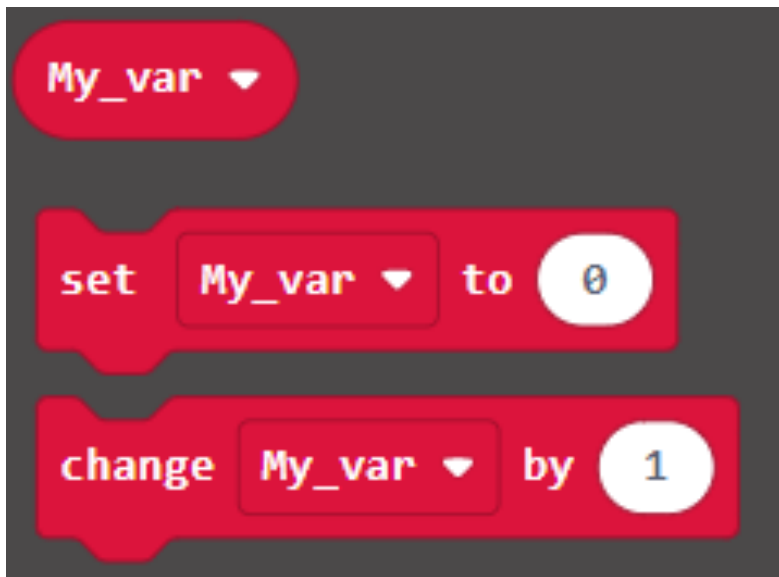
false ▾

- Super important to understand
- Core element of 'computational thinking'
- Selection is condition based If then x
- Comparisons are very useful! Links to Mathematics

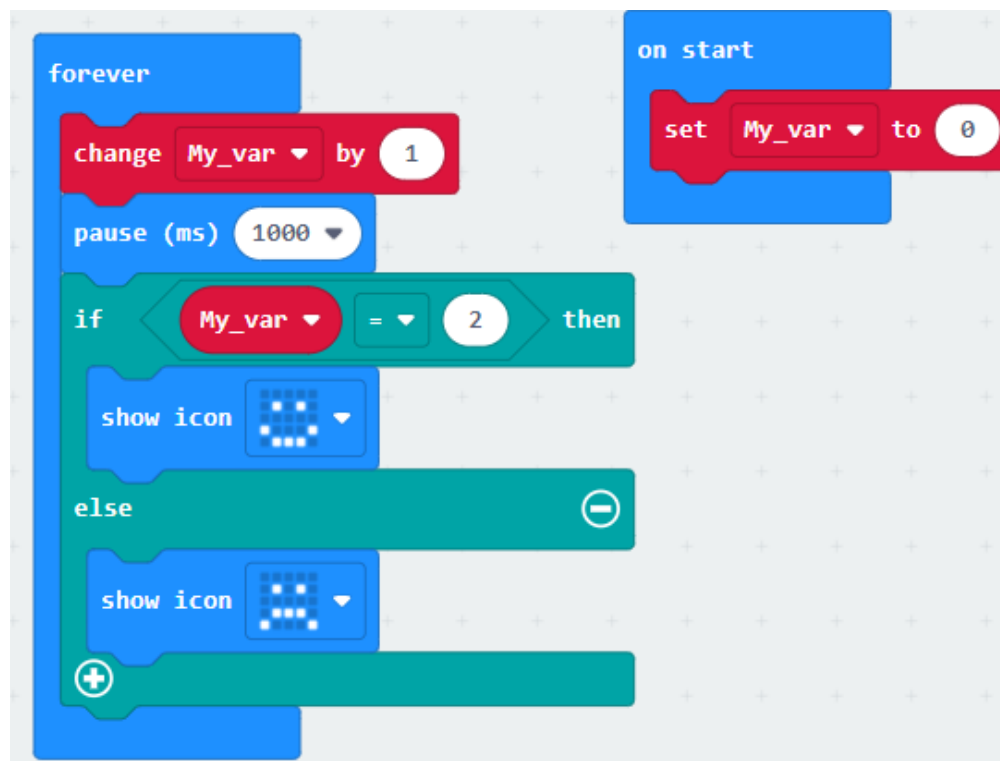
```

forever
  if light level < 150 then
    show icon [grid icon]
  else -
    show icon [grid icon]
  +
  
```

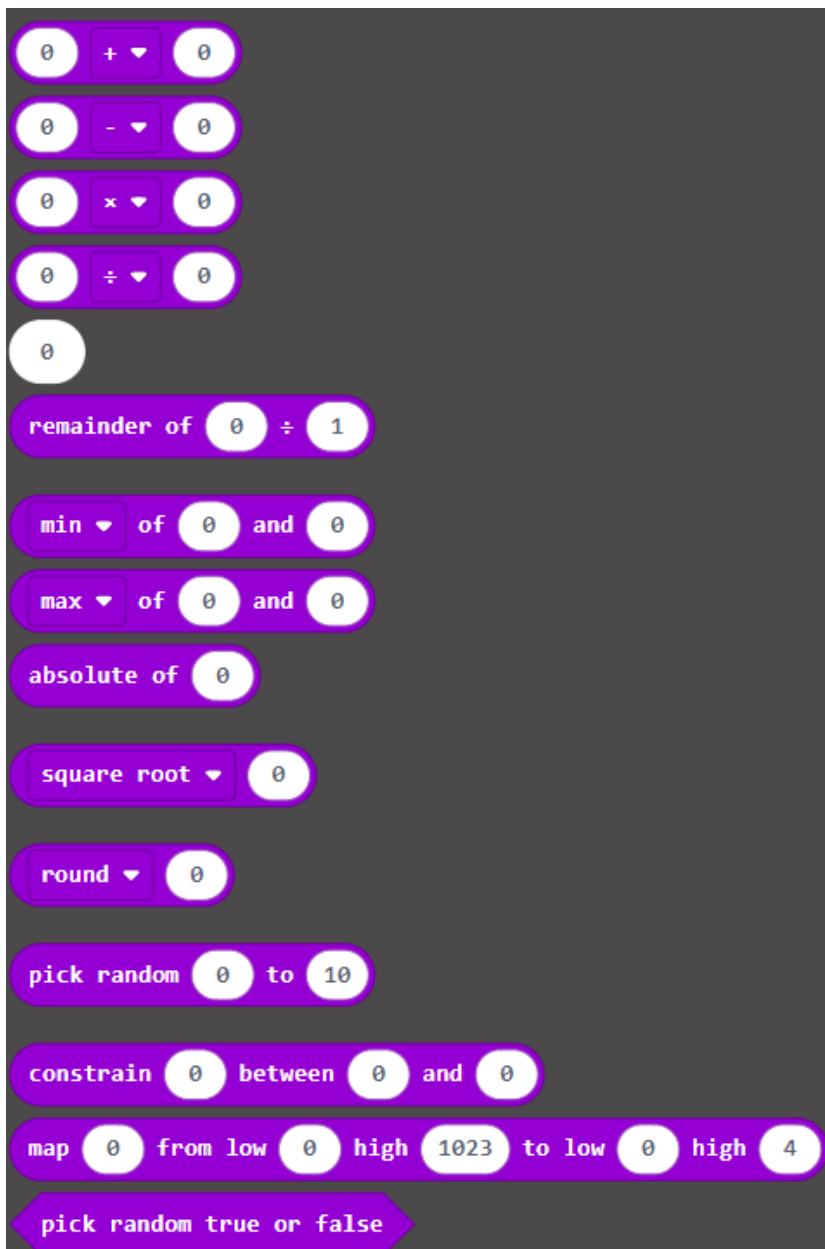
Variables



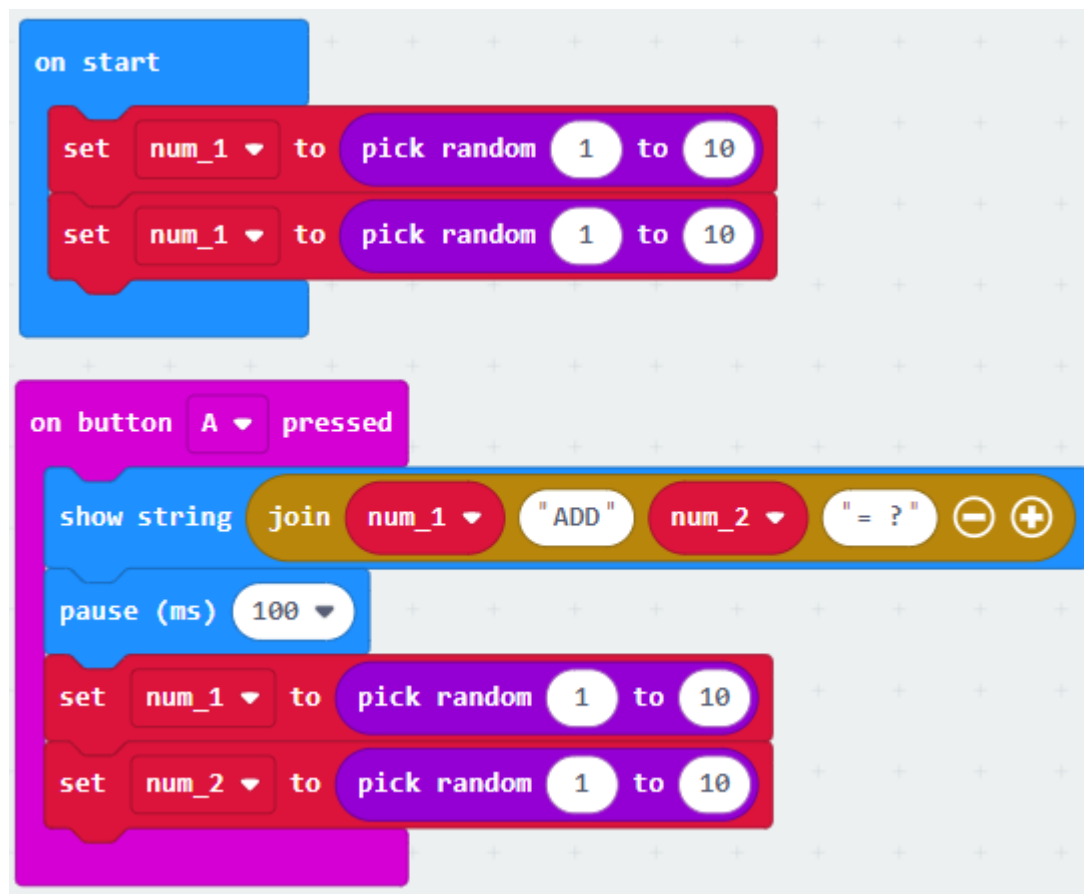
- Must be created with a name – use sensible names!
- Contain any data type
- Can be changed/set/incremented in other blocks
- Key part of Computer Science theory



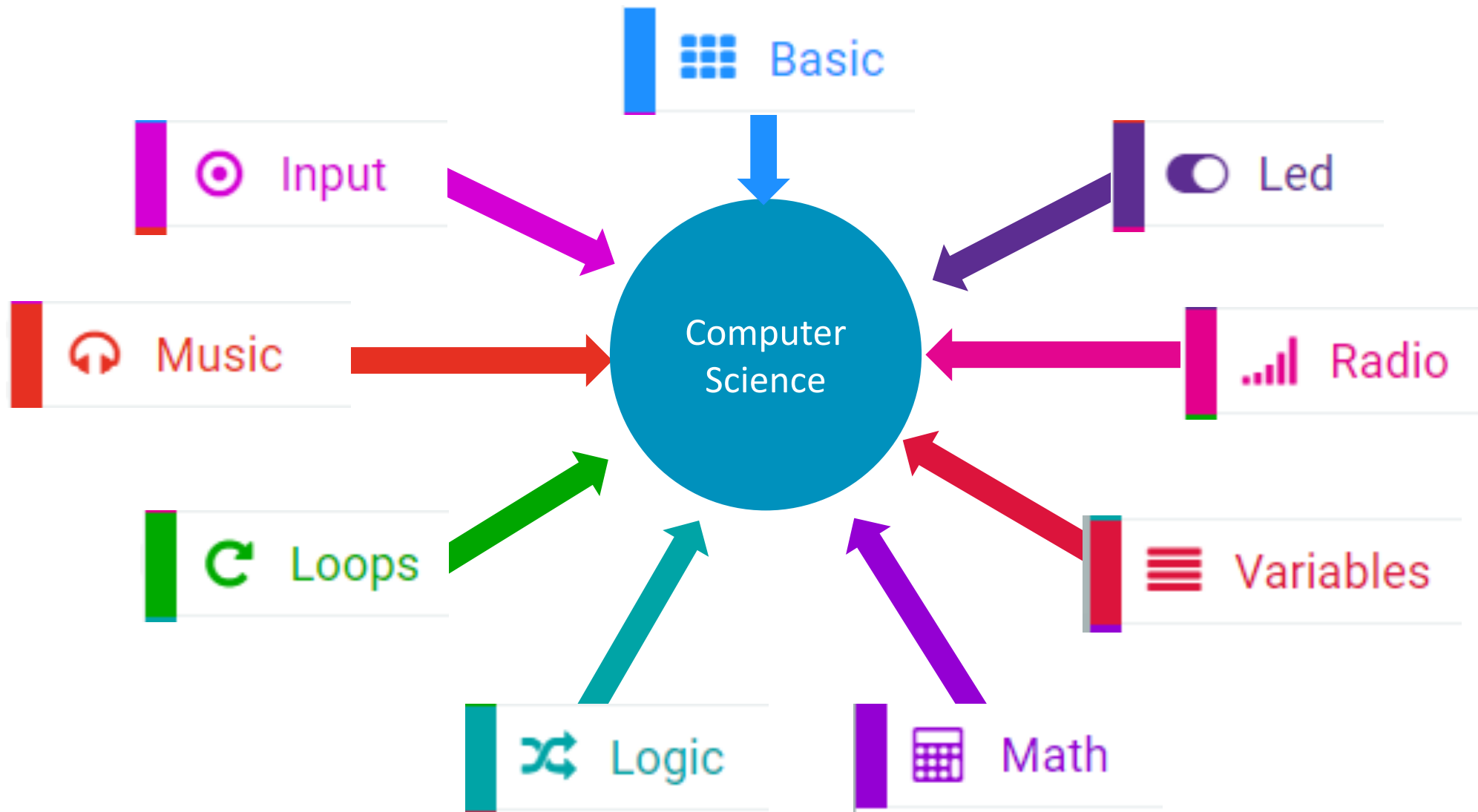
Math



- Hopefully most of these are familiar
- Includes RANDOM number and Boolean generator



Combining the blocks = complexity



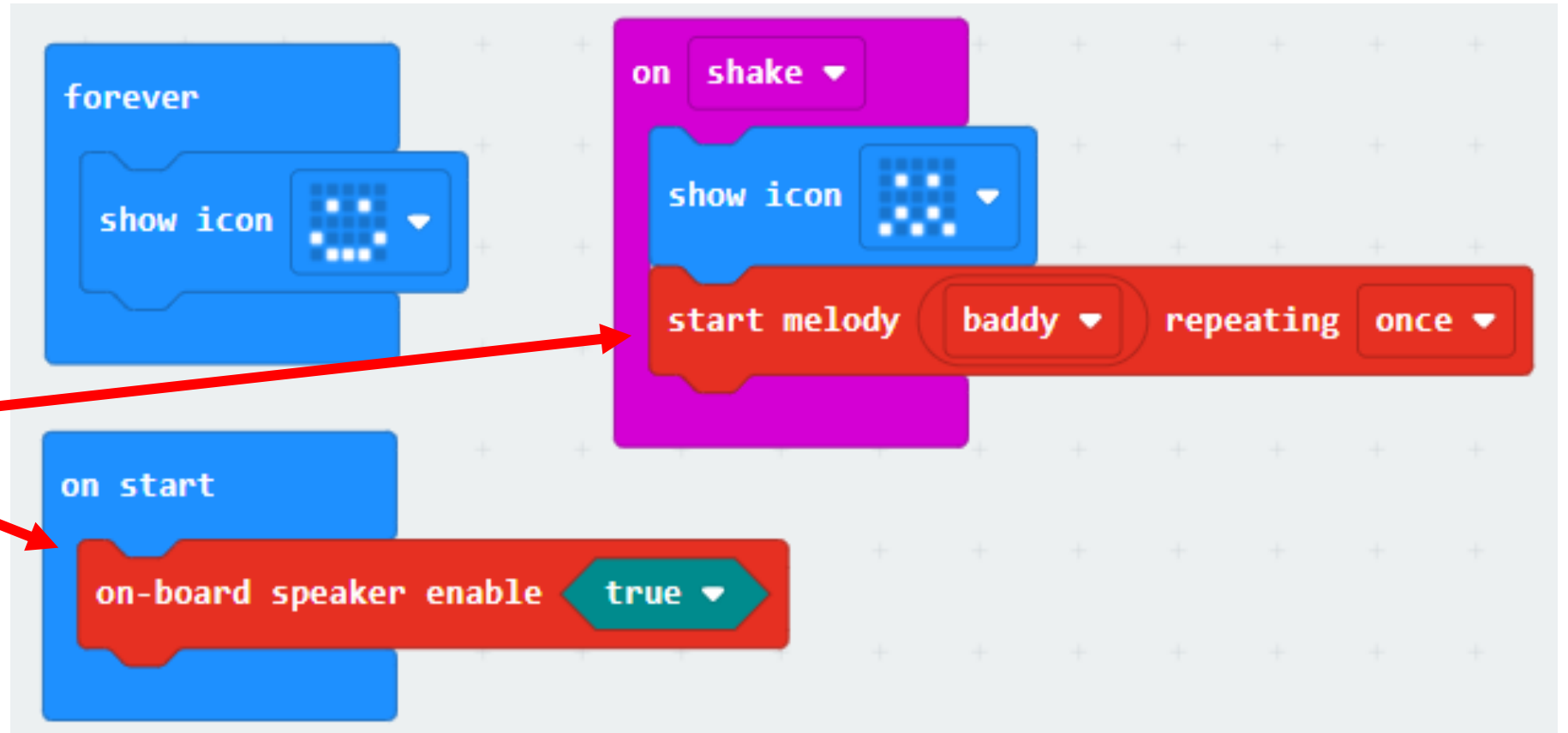
Blocks and over-confidence

The image displays two columns of Scratch code blocks. The left column contains the main game logic, starting with an 'on shake' event that sets a random choice for player 1 (0, 1, or 2). It then uses three 'if' blocks to handle choices 0, 1, and 2, each showing a string and sending a radio message. Below this is an 'on start' block that shows 'Shake to play' and sets a score of 0. At the bottom, a 'forever' loop sets a radio group to 1, and an 'on button A+B pressed' block sets the score to 0. The right column shows the 'on radio received' logic for the received number. It uses a series of 'if' and 'else' blocks to compare the received number with the player's choice. If they are equal, it shows 'Draw!'. If the received number is less than the choice, it shows 'Win!', increments the score, and shows the score. If the received number is greater than the choice, it shows 'Lose!'. There is also a specific 'if' block for the case where the player chose 2 and the received number is 0, which also shows 'Lose!'.

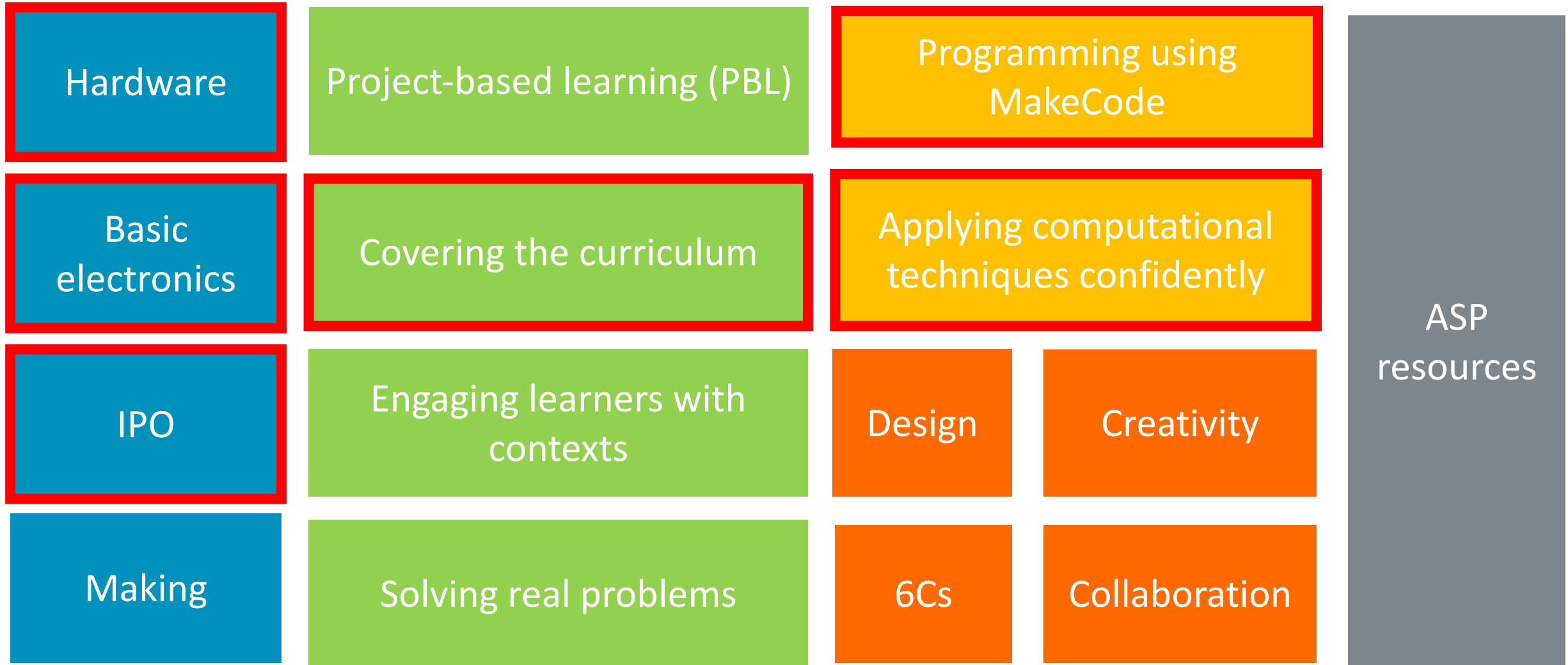
- Assumption of simplicity
- Allows rapid prototyping
- JS/Python there if needed
- CT over syntax
- Complex enough for schools/K12

V2 blocks

V2 only



The core content



MICRO:PET

micro:
project

Setting the scene

Loneliness and isolation is a real problem for children staying in hospitals for long periods, especially in rural areas. You have been tasked with creating a digital pet that can be played with and keep people company while they stay in hospital.

Success criteria

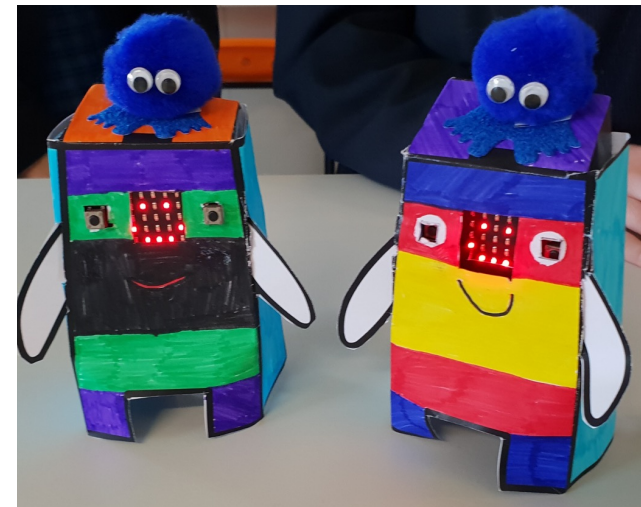
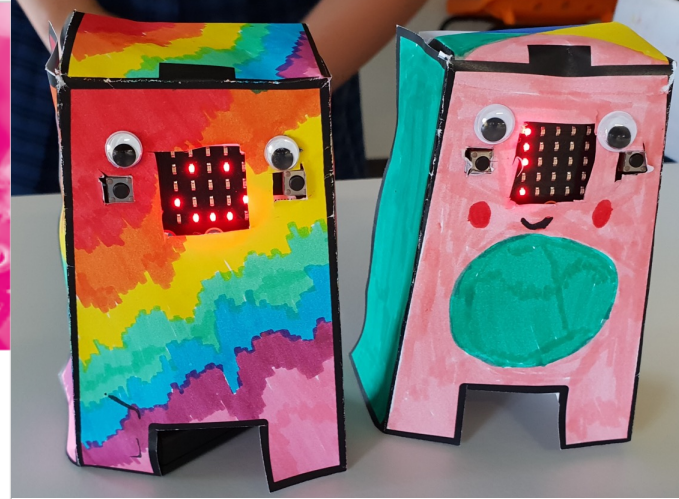
The product must be suitable for one of the users listed below and the pet must:

- ✓ Look like a friendly pet (be creative)
- ✓ Be robust enough to be played with
- ✓ Contain a micro:bit that users can interact with
- ✓ Have a face to express emotions when interacted with
- ✓ Have one or more interactions programmed so it behaves like a pet to keep the user company

Some ideas

Here are some possible ideas that could be programmed for your pet:

- Reacting to playing/shaking (accelerometer)
- Feeding (every few hours)
- Needing attention (gets lonely if not interacted with frequently)
- Sleeping and waking (light sensor)
- Reacting to temperature (temperature sensor)
- Mini games
- Communication/interaction between pets
- Use of other inputs such as other types of sensors (requires additional hardware)
- Use of other outputs such as sound or movement (requires additional hardware)



Micro:pet IPO design

Input

- Shaking the micro:pet (accelerometer)
- Temperature sensor
- Light sensor
- Food button

Process

- + G force thresholds
- + Temperature thresholds (happy between 18-24 degrees C)
- + If dark, goes to sleep
- + Compares food variable value to comfort thresholds


Output

- + Sad face/noise
- + Teeth chattering output and sweaty output
- + Sleep face and snoring sound
- + Hungry face/noise

Design thinking

- + IPO tables
- + Concept designs and rapid prototyping
- + Iterating on designs

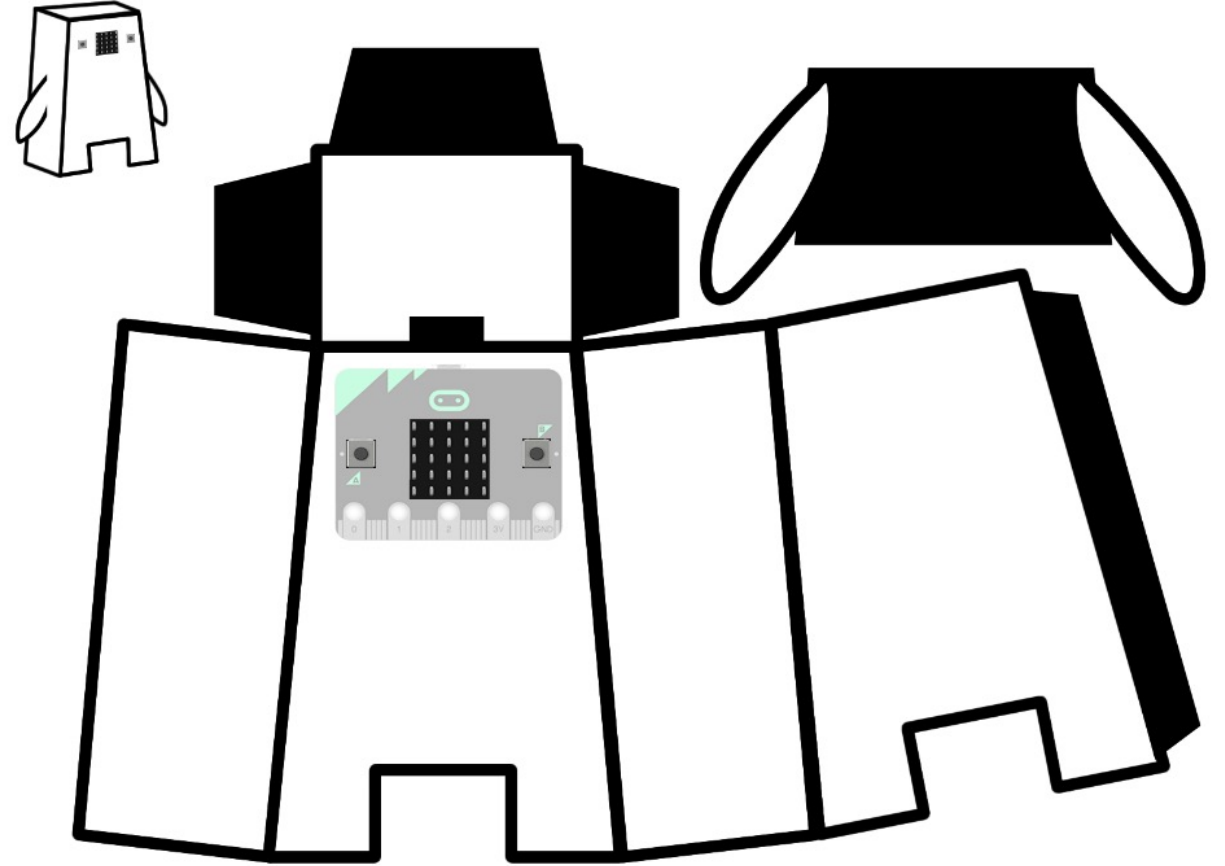
Input	Process	Output

Objectives:	Users:	Materials:	Features Essentials:
Sketch:			Nice to have:
		How is this better?	Success criteria:
		Branding:	How to make it better?
			

Success criteria

The product must be suitable for one of the users listed below and the pet must:

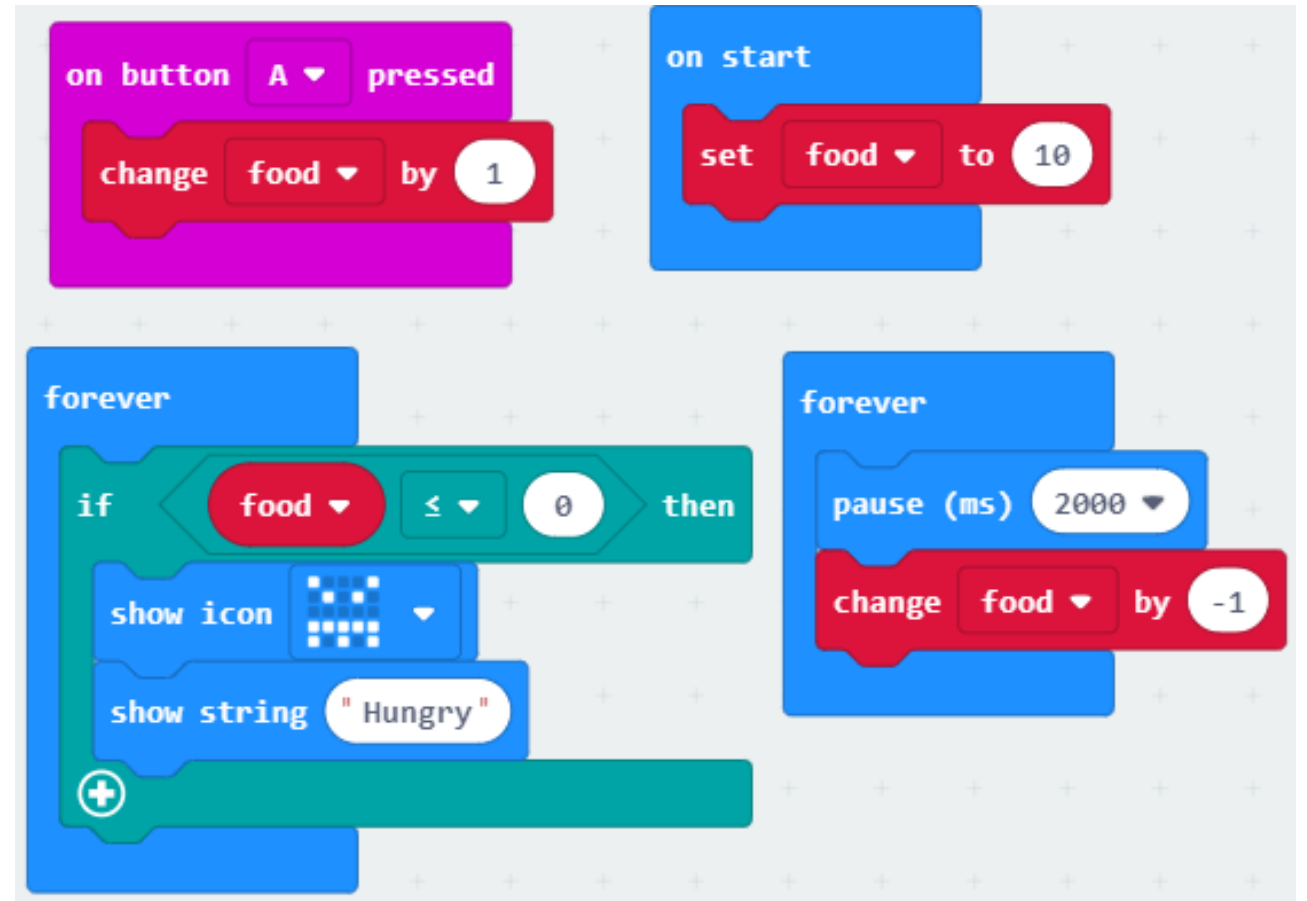
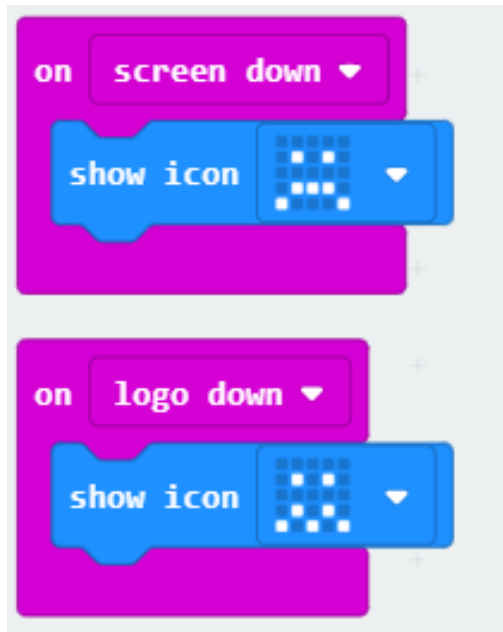
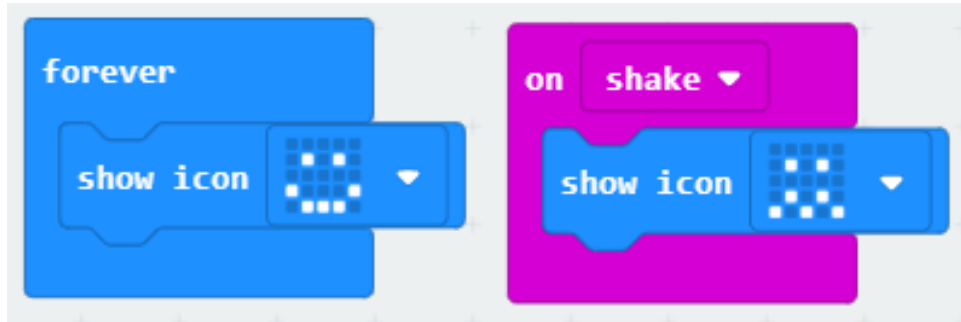
- look like a friendly pet (be creative)
- be robust enough to be played with
- contain a micro:bit that users can interact with
- have a face to express **emotions** when interacted with
- have one or more **interactions** programmed so it behaves like a pet to keep the user company



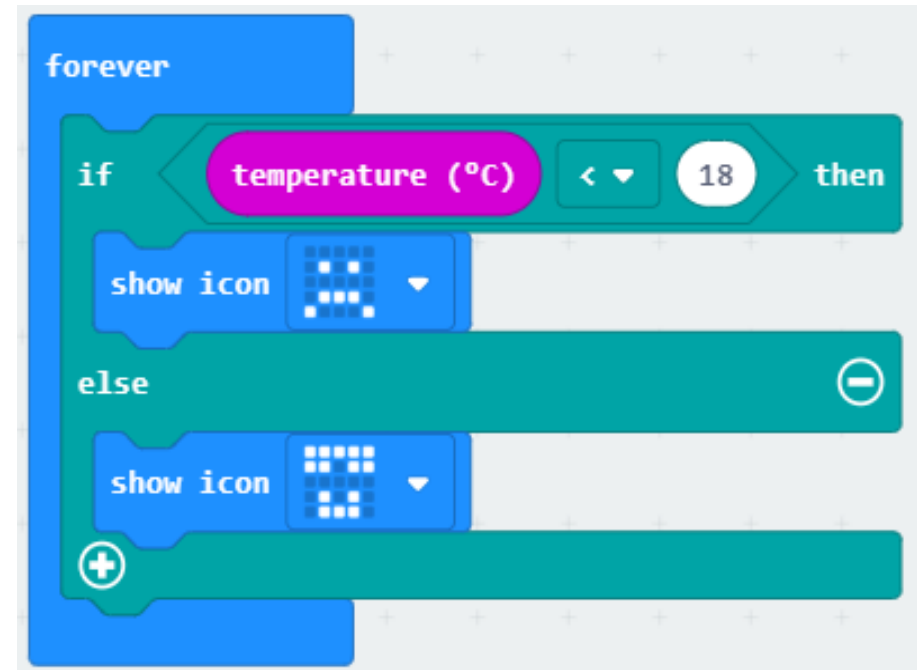
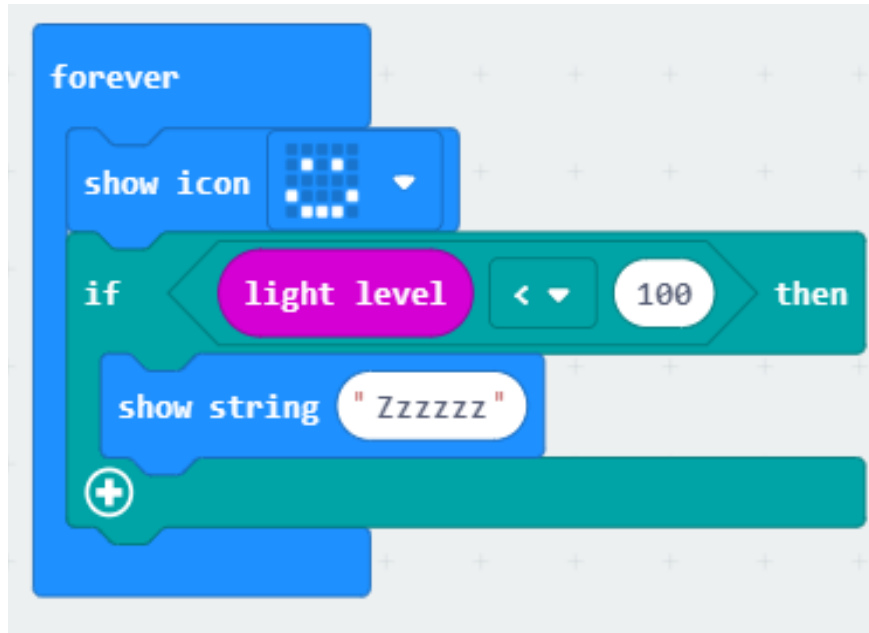
micro:pet feature ideas

- Reacting to playing/shaking (accelerometer)
- Emotions that are effected by interaction or noise
- Feeding (every few seconds/minutes)
- Needing attention (gets lonely if not interacted with frequently) and tells you!
- Sleeping and waking (light sensor) - think snoring
- Reacting to temperature (temperature sensor)
- Communication/interaction between micro:pets (advanced)
- Use the new speaker and mic to make your pet come alive

Interaction

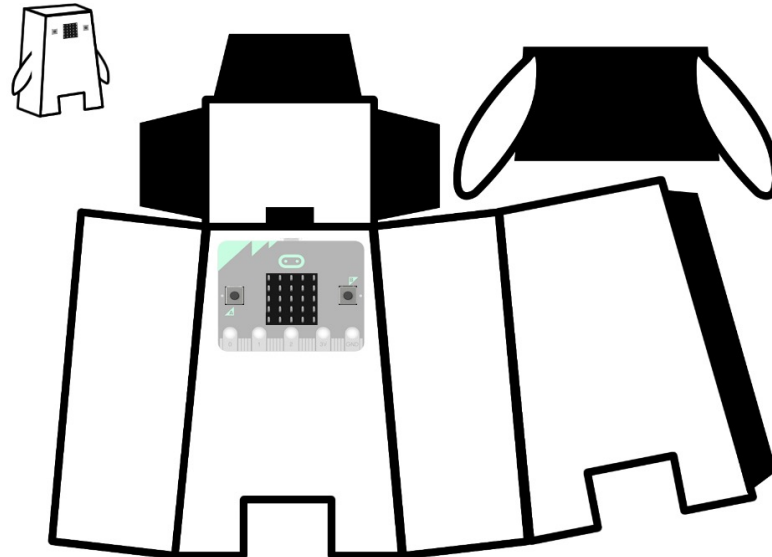
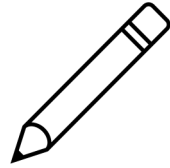
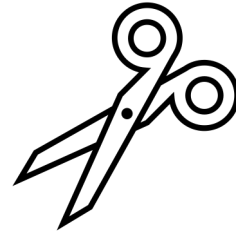


More features



Making

- Card or thick paper
- Scissors
- Tape
- Colouring in pens/pencils
- Decorative bits – pipe cleaners, googly eyes
- Velcro tape



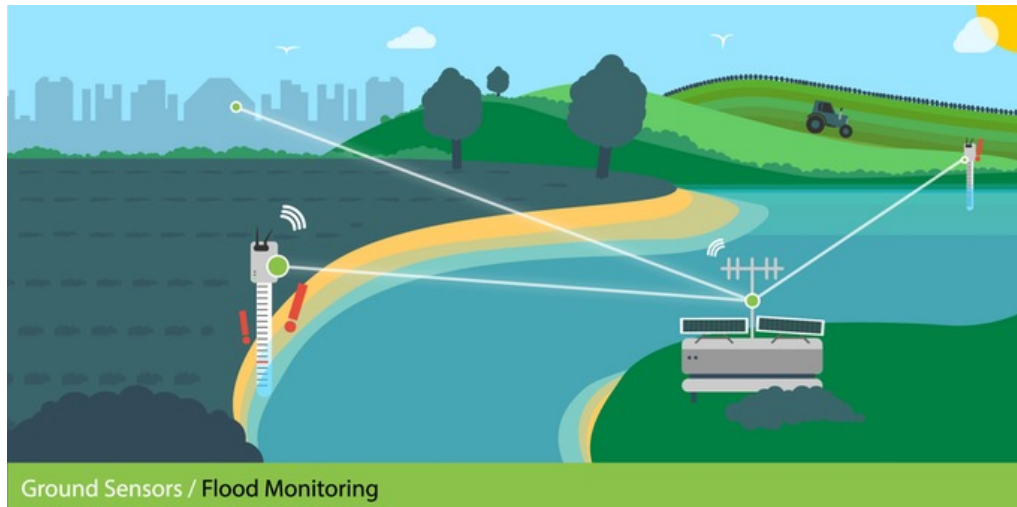
What is IoT? – a useful context

Internet of Things

- Adding electronics to everything
- Electronics such as:
 - Sensors (cameras, temp, light, humidity etc.)
 - Microprocessors
 - Transmitters
 - Screens
 - GPS
 - RFID
- Everything can link to the internet/each other
- Anything electronic can be controlled
- Can stream data continuously



Sensing the world



What else could IoT devices be used for?

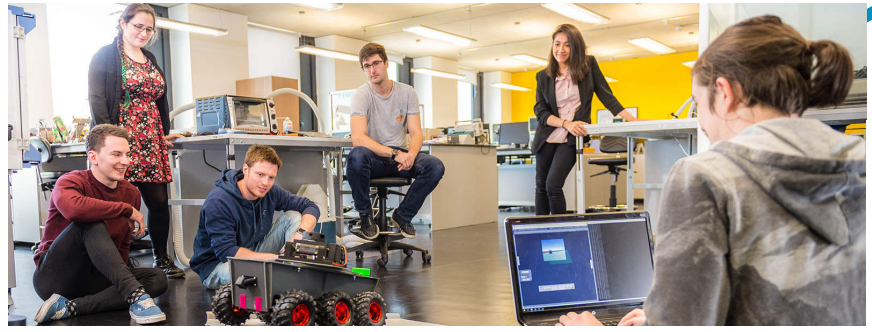
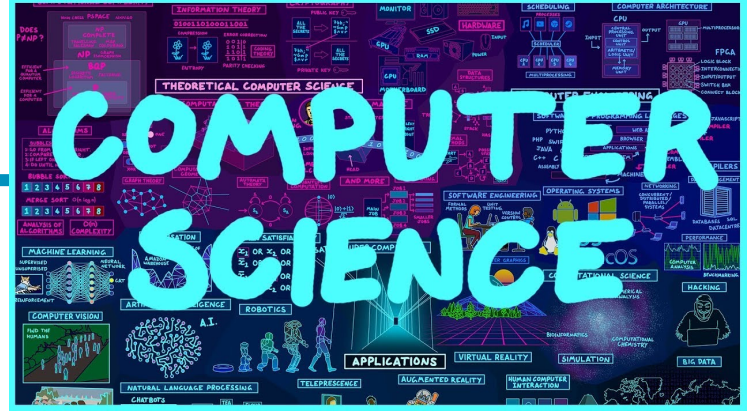
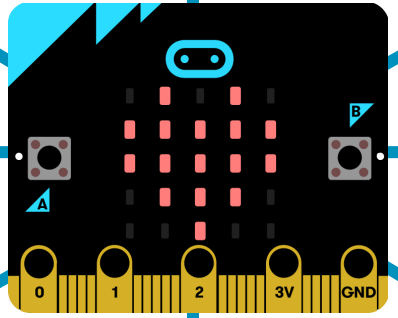
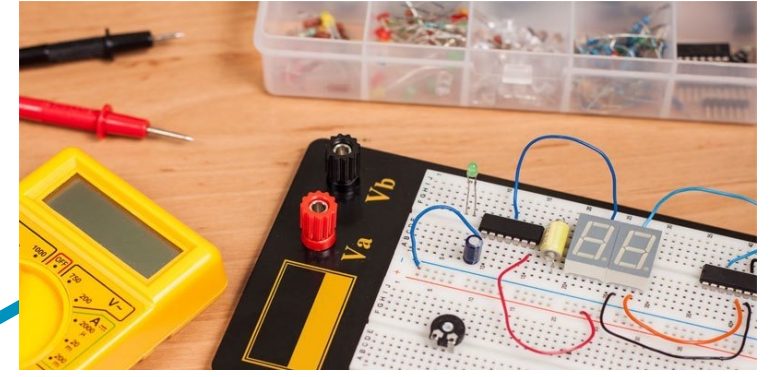
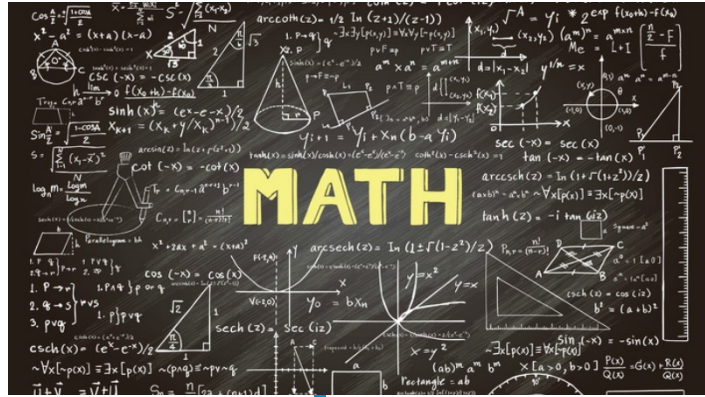
Think about their use in:

- Law enforcement
- Medicine
- Retail
- Cars

THE GLOBAL GOALS

For Sustainable Development

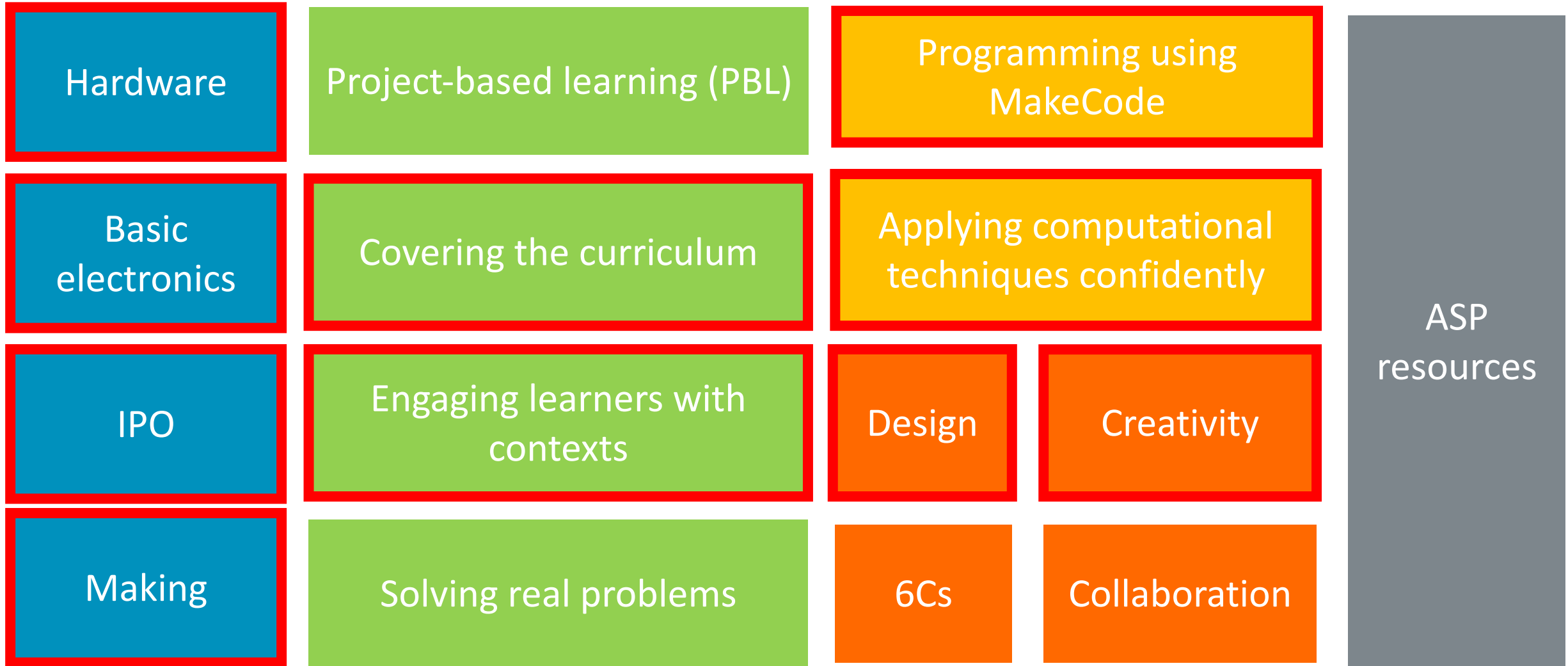




DATA SCIENCE



The core content



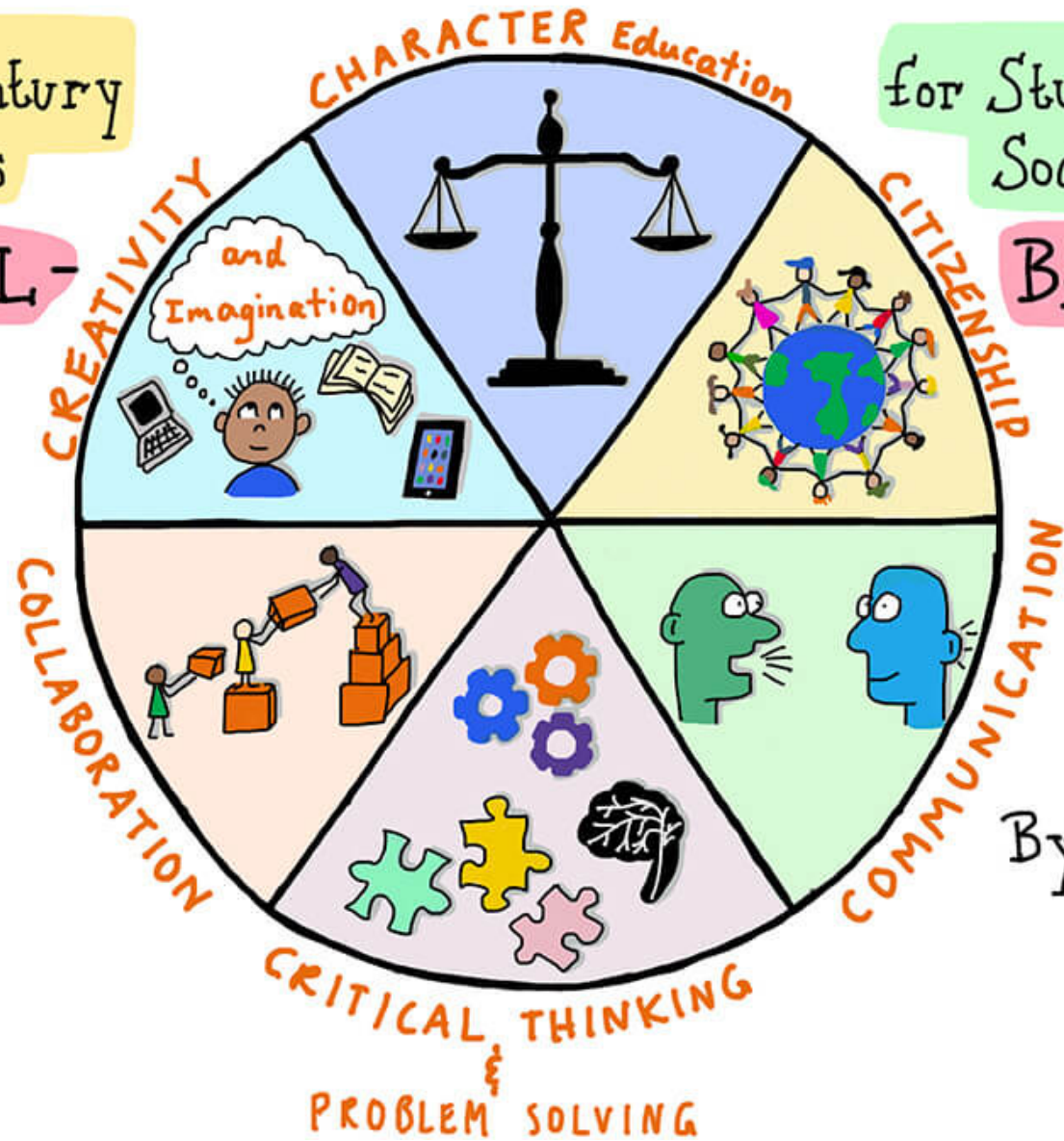
THE 6 C'S of Education

21st Century skills

for Student and Society

WELL-

BEING



By Michael Fullan

@sylvia duckworth

Why making matters – thinking in systems

‘Engineers use a unique mode of thinking based on seeing everything as a system. They see structures that aren’t apparent to the layperson, they know how to design under constraints, and they understand trade-offs. Adopting an engineering mindset can help you in any field.’

Engineering mindset:

- The ability to see a structure where there’s nothing apparent
- Designing under constraints
- Understanding trade-offs



Engagement

The Computational Thinkers

concepts



Logic

Predicting & analysing



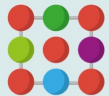
Evaluation

Making judgements



Algorithms

Making steps & rules



Patterns

Spotting & using similarities



Decomposition

Breaking down into parts



Abstraction

Removing unnecessary detail



approaches



Tinkering

Changing things to see what happens



Creating

Designing & making



Debugging

Finding & fixing errors



Persevering

Keeping going



Collaborating

Working together

We're all computational thinkers here!

When you think about it, whether we're parents, pupils or teachers - we're all natural computer scientists, capable of computational thinking.

Our brains, like computers, process, debug and make simple algorithms every day!



Supported by BT

arm School Program

Building resilience in learners

4 B's

- + Brain
- + Book
- + Buddy
- + Boss

Classroom strategies

- + Fail Early Fail Often
- + Open questioning
- + Appropriate scaffolding
- + Building confidence
- + Collaborative groups

Engaging girls

Obvious STEM disparity

What disengages girls?

- + Peer pressure
- + Competition
- + Things over people
- + Subject nature?
- + Perceived lack of creativity

How can we re-engage?

- + Relevant contexts
- + People centric problems
- + Creative application of skills
- + Role models
- + Application of non-trivial soft skills

Derailers

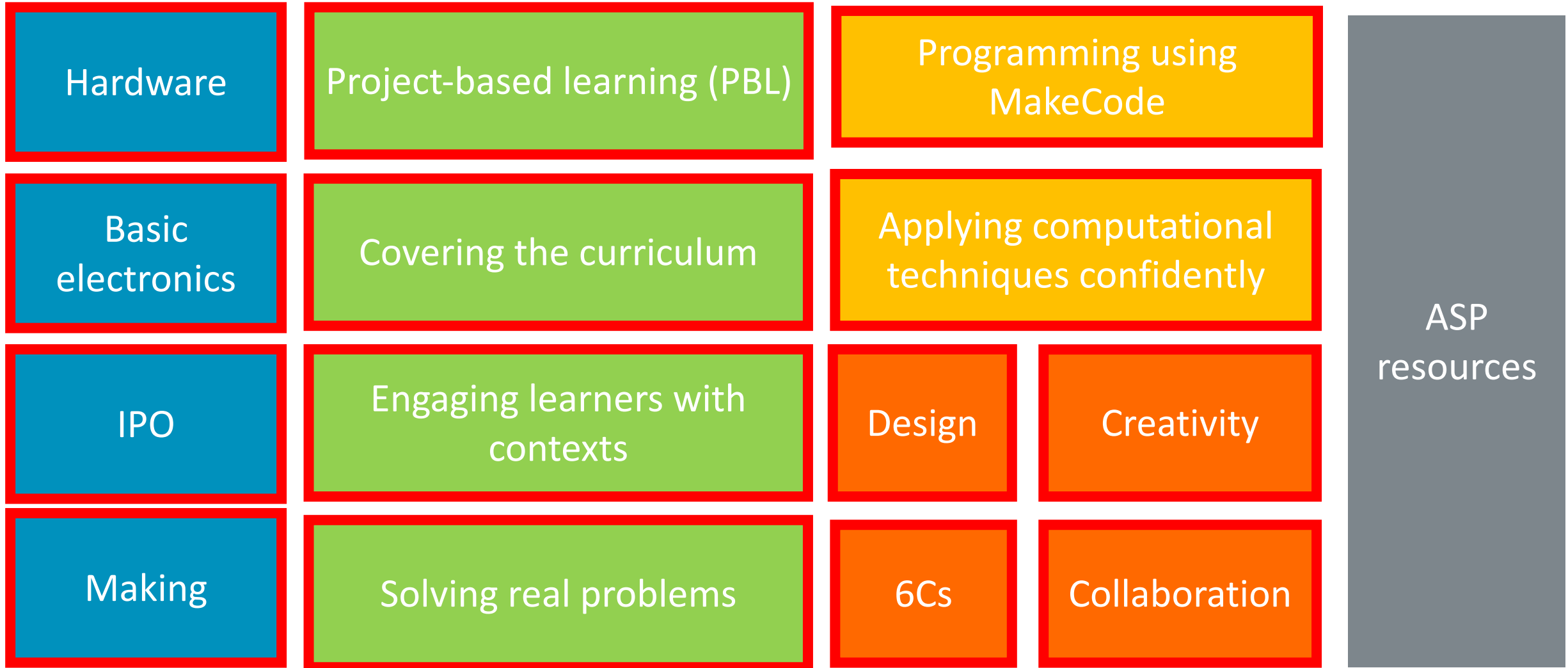
Physical

- + Broken hardware
- + USB ports not allowed/working
- + IDE/libraries blocked by IT provider
- + Batteries
- + Appropriate 'making' hardware
- + Storage between lessons
- + Hardware accessibility
- + Hardware availability
- + Cost

Practical

- + Over complicating things
- + Not iterating
- + Lack of collaboration
- + Getting 'stuck'
- + Lack of resilience
- + Ignoring the success criteria
- + Teacher confidence

The core content



Why aren't we using Physical Computing more?

86%

of students said BBC micro:bit made Computer Science more interesting

85%

of teachers agree it has made ICT/Computer Science more enjoyable for their students

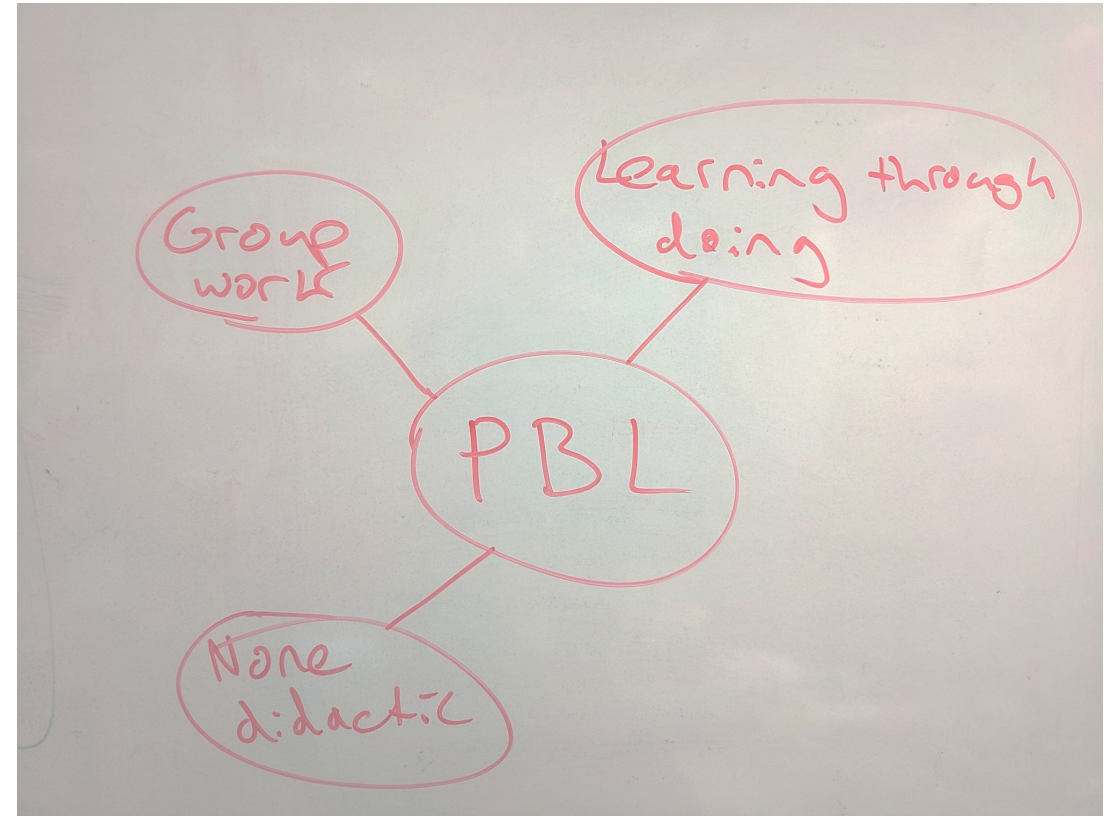
70%

more girls said they would choose Computing as a school subject after using the micro:bit

Describe PBL again

Following work with a new concept, reflect, record your thinking in relation to the following:

- 3 insights gained
- 2 strategies or ideas
- 1 thing I'd like to explore further?



micro:course – v1 and v2 versions

- Set of projects that build in complexity, PBL in nature, ASP schema, micro:bit based

QUICK START

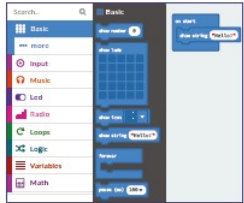
micro:project

Setting the scene
In this introductory project we will learn about the basic functionality of the micro:bit and how to program it.


Success criteria

- ✓ Create a simple program using MakeCode
- ✓ Download the program and upload it to the micro:bit

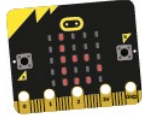
1
Log into your computer and go to <https://makecode.microbit.org>
Click the **Basic** tools and drag across the **on start** and the **show string** blocks as below



2
Change the text from **Hello!** to something else (maybe your name)

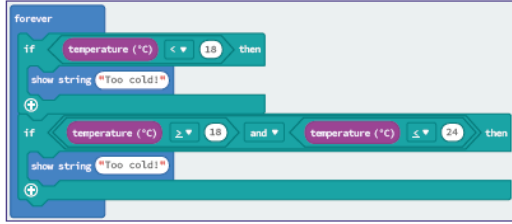


3
Congratulations!
You have created your first program for the micro:bit. Your program will be "simulated" for you so you can see what the output will be.




6 Quick start Arm School Program


6
The message "Too cold" will only show if the temperature is *less than* 18°C.
That should now have met the first success criteria. The second success criteria will be slightly more complex as it will need two tests to see if the temperature is above 18°C and below 24°C. To do this we need to use another Logic block with an *and* in it to carry out two tests at the same time.



6
The final *if* block is very similar to the first block but with a *>24* for the temperature input and "Too hot!" string. Duplicate the first *if* block and change the values to speed things up.



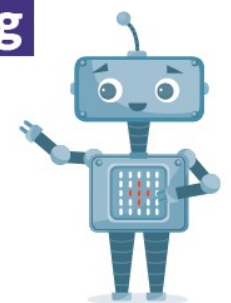
PRO TIP
You can **right click** on any block and **duplicate** them to save time!



10 Temperature sensor Arm School Program



learn
computing
with the
micro:bit



ASP resources

<https://www.arm.com/resources/education/schools/content>

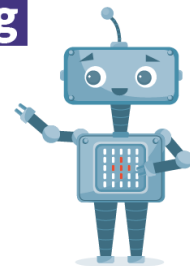
Teaching and learning resources

- + Introduction to Computing Using micro:bit
- + Robotics and Internet of Things Course
- + Introduction to Programming Using MicroPython
- + Computational Thinking Tasks
- + Micro:course (book)

 arm School Program

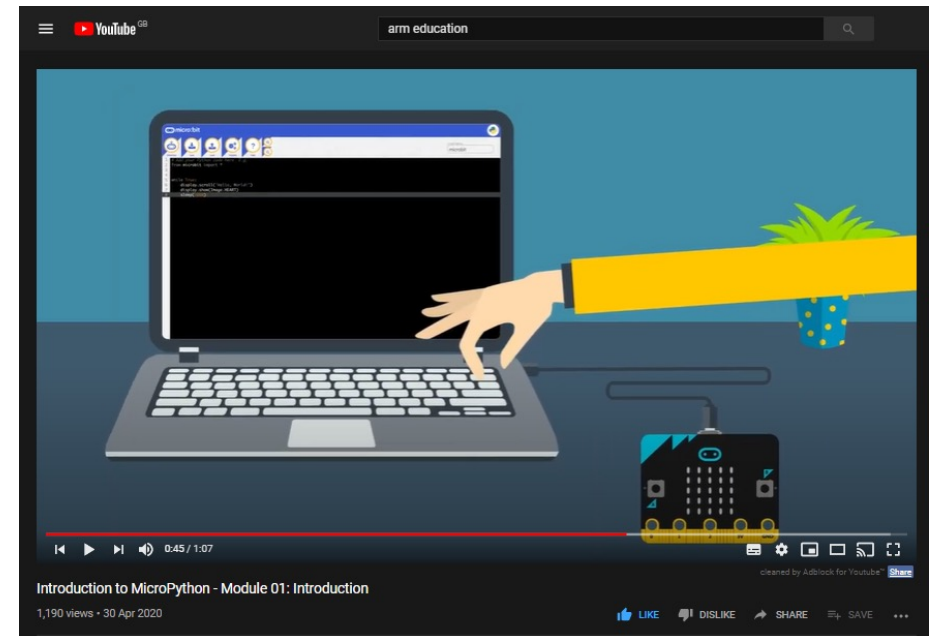


learn
computing
with the
micro:bit



YouTube

- + Intro to Computing with micro:bit
- + Intro Programming with MicroPython



Teaching with Physical Computing

A new series of PD courses from the Arm School Program

A course for teachers on Physical Computing and how to apply it through Project-Based Learning in the classroom.

Teaching with Physical Computing

Search for “Project-Based Learning” on edX.org

Course 1 Introduction to Project-Based Learning

Course 2 Practical application and classroom strategies for PBL

Course 3 Assessment of Project-Based Learning

Course 4 Soft skills, teamwork and the wider curriculum



arm

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה



The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks