Micropython for micro:bit

A custom course for Grade 7 learners

Overview:

	Topics	Resources
1	Introduction to the course Theory and project lessons Success criteria Blocks to code Initial baseline assessment (30 minutes)	
2	Getting started with micro:bit and micro python Mu – what is an IDE? Scrolling text "Hello world" Displaying built in images	Micro:bit (1 per learner) Access to PC and Mu editor
3	Importing Creating your own images Variables Lists Functions as commands (arguments)	
4	Event handling Buttons / how they work Events Selection Event handling (interrupts)	
5	Variables Strings Storing data in variables	
6	Using the pins How pins work Responses loops	
7	Project 1 Healthy Eating - intro	Making material
8	Project 1 Healthy Eating – build and testing	
9	Project 1 Healthy Eating	
10	Making music Speaker hardware – how a speaker works Setting up the speaker Modules Lists	Crocodile clips Speakers
11	Speech Poetry/rapper auto generator	Crocodile clips Speakers
12	Musical performances	Crocodile clips Speakers
13	Random and it uses True random? Random numbers Crypto and uses in real world	
14	Gestures and movement	
15	Direction Compass	
16	Accelerometer	

	x, y, z, axis Taking a reading Responding to a reading	
17	Crash detector for a bike – design and intro	Making material A bike (for demonstration)
18	Crash detector for a bike – build container	
19	Crash detector for a bike – program code	
20	Crash detector for a bike – testing with bike	
21	Storage	
22	Machine related code	
23	Temperature sensor and readings	
24	Light sensor	
25	LEDs and classes	
26	Neopixel What are LEDs? What are addressable LEDs? Iteration	
27	Real time neo pixel	Neopixel or equivolent
28	Radio 1	
29	Radio 2	
30	Pin use and making touch buttons	
31	Servo motor	A servo motor Crocodile clips
32	Servo arm movements	A servo motor Crocodile clips
33	micro:PET introduction, planning and design	Making material Micro:pet net Design sheets
34	micro:PET speech and interactions	
35	micro:PET project and functions	
36	micro:PET building the physical model	
37	Building, programming, testing and improvements	
38	Coding, testing and preparing	
39	micro:PET Presenting the solutions and evaluation	
40	Evaluation and summative assessment	

Aims and outcomes

The aim of this course is to introduce learners to the key STEM topics, physical computing, sensors, programming and IOT through project based learning. Learners will be immersed in a series of projects which involves rapid team based development of a solution to meet a criteria. These sprints of project based work are interspersed with theory lessons that cover the core concepts necessary for the following projects.

Key features

- Team project-based learning
- Design ideation, development, evaluation and refinement
- Authentic STEM based contexts
- Hands on physical Computing and Engineering challenges
- IOT themes
- Entrepreneurship

Pedagogy

This is a blended learning course where all the resources are hosted on a VLE. Learners can interact with the resources as and when they need them and this interaction with the platform is interspersed with formal delivery from the teacher where necessary. The course also includes many projects and employs a Project Based Learning (PBL) approach to delivering this content that builds on prior learning, challenges learners and encourages collaborative ideation and creation to solve authentic problems. The approach is outlined below:

- Explore and define
    - Abstraction
- Imagine and identify processes
    - Imagine
    - Generalisation/pattern recognition
    - Decomposition
    - Algorithm
- Do and review
    - Collaboration
    - Perseverance
    - Fail early, fail often (FEFO)
- Evaluation, iteration and celebration
    - Generalisation/pattern recognition

The projects are designed to require minimum directed teaching but a "Lesson flow" is provided to give the sessions some structure and to introduce some of the more challenging topics and concepts. As the teacher is doing minimal delivery of the content they are free to circulate amongst the learners and troubleshoot where needed, this allows weaker learners to be better supported and more able learners to be pushed harder when appropriate. There are stretch activities in every project that extend the knowledge and application of the skills for these learners.

Whilst circulating amongst learners teachers should be mindful of the "Key concepts" and "Key words" and should quiz learners at appropriate times on their knowledge of the concepts by getting them to explain the code/blocks they are using and also the thinking behind their designs for the making activities. Learners also may need reminding of the success criteria.

To aid the teacher in delivering the lessons, detailed plans are provided that include:

- The big picture – a brief overview of why this lesson is relevant
- Learning objectives
- How to engage learners
- Assessment for learning – expected, good and exceptional progress statements
- The key concepts of the lesson
- Key words – to be used for questioning during the lesson
- Differentiation advice
- Resources – what hardware and software are required
- Lesson flow – this is the sequence of events for the lesson
- Making – a description of any practical activities
- Questions – some questions to interrogate learners understand of the core concepts

A lesson plan template is in Appendix 1.

Making

The making element of all these projects is what makes these engaging STEM activities as they draw together learning about computer science, maths and engineering (designing and making) throughout the activities. Whilst the making element allows for effective differentiation and group work all learners should be given the opportunity to both code and make as both activities are rewarding. The design and iterative improvement of the physical parts of the projects represent the engineering element and learners should be encouraged to design, prototype and refine their work. The designs can be presented in many ways but should show the product, describe its purpose and function and how it has designed to suite its use with the project. It is vital that learners are made to design the physical products before starting to make them. Once the product is made it should be tested, evaluated and then improved.

Fail early, fail often

STEM is a mix of different disciplines and learners may initially struggle with the content. This is something to be celebrated as failure is an opportunity to learn and to improve. This mindset of failing early and failing often breeds resilience in learners and is an important mindset to cultivate in the STEM field. When learners are stuck they should be encouraged to seek assistance from the project resources, then the internet, then their peers and finally their teacher.

Test everything

Testing is an important aspect of iterative design. This is true of programs and of physical products. Learners should be encouraged to constantly test their products and improve them in an iterative fashion.

Evaluation and Success criteria

Each project has a list of “Success criteria” at the beginning of the project and these are referred to throughout the project. It is important to emphasise the importance of this criteria as this keeps the learners focussed and forms a checklist of what needs to be achieved. This also allows their work to be evaluated against the criteria. This is important as a STEM skill as projects and products always have criteria that needs to be met to be successful.

Teacher CPD

This type of delivery and assessment needs specific interventions. Teachers need to intervene only when necessary and should only do enough to allow the learners to progress independently. Teachers should use questioning to determine the motivation and purpose of the learner’s actions and decisions and teachers should allow learners to make mistakes and then help them troubleshoot only if required. This methodology will be further explored in the teacher CPD course that will be developed alongside this course.

Assessment

The assessment will consist mainly (80%) of a mix of block based and Python PRIMM questions (see Appendix 2) as well as a number of multiple choice (MCQ) questions covering the theory topics. The assessment will also determine the learner’s confidence in IOT product design concepts to give a baseline to compare to the final assessment to determine efficacy using both quantitive and qualitative data. This can be used to iteratively improve the course and to establish impact.

The assessment will consist of interactive MCQs on the ASP VLE and will last no more than 1 hour. It will consist of roughly equal weighting of topics from the course.

Formative assessment

The teachers leading the project based sessions will be continuously monitoring progress and will give awards at the end of the project for various categories such as:

- Best teamwork
- Best explanation of a topic (based on teachers questioning throughout the day)
- Most sophisticated programmed solution
- Best engineered solution
- Best designed solution
- Most imaginative solution
- Perseverance award

Formative assessment will be recorded and will make up 20% of the learner's final grade.

The learners will be expected to upload their work digitally to the VLE every lesson. This can be used by the teacher to formatively assess the learners progress.

Assessment objectives

AO1	Demonstrate knowledge and understanding of technology
AO2	Apply knowledge and understanding of technology
AO3	Analyse and evaluate problems
AO4	Demonstrate application of knowledge and understanding to solve problems

Grade scale and descriptors

Pass	• Learners will have demonstrated limited knowledge and understanding of the concepts and principals involved in the course. • Learners will have applied the principals and concepts using some analytical and evaluative thinking and practice to solve a problem. • Learners will have demonstrated some ability to apply knowledge and understanding to solve problems. • Learners will have collaborated with their peers and demonstrated some communication and teamwork.
Intermediate	• Learners will have demonstrated mostly accurate knowledge and understanding of the concepts and principals involved in the course. • Learners will have appropriately applied the principals and concepts using analytical and evaluative thinking and practice to solve a range of problems. • Learners will have demonstrated their ability to apply knowledge and understanding to solve problems. • Learners will have collaborated reasonably successfully with their peers and demonstrated mostly effective communication as well as effective teamwork.
Higher	• Learners will have demonstrated relevant and comprehensive knowledge and understanding of the concepts and principals involved in the course. • Learners will have effectively applied the principals and concepts using sustained analytical and evaluative thinking and practice to solve a range of problems. • Learners will have successfully demonstrated their ability to apply knowledge and understanding to solve substantial problems in an efficient manner. • Learners will have collaborated successfully with their peers and demonstrated effective communication as well as efficient and effective teamwork.

Appendix 1 – Lesson plan template

Lesson plan example

The big picture – why is this relevant?	Learning objectives:
•	•
Engagement – How can I engage learners?	Assessment for learning
•	**Expected progress:** **Good progress:** **Exceptional progress:** •
Key concepts:	Key words:
•	•
Differentiation:	Resources:
	•
Lesson flow	
•	
Making	
•	

Appendix 2 – PRIMM: A scaffolded approach to teaching programming

PRIMM is a teaching methodology based on educational research that scaffolds programming activities to maximise engagement and understanding of the core programming concepts. The best approach to ensuring all learners fully understand and are able to apply their understand is to utilise a blended approach to delivery, combining group discussion, paired work, scaffolded practical tasks and creative, engaging tasks and challenges.

PRIMM stands for the following:

- Predict
- Run
- Investigate
- Modify
- Make

Scaffolding

Learners typically need a lot of support to understand programming concepts, especially at the beginning. There are many techniques and concepts that whilst individually are straight forward to understand and apply in a given context, applying the same technique in an unfamiliar context is challenging. This means that strategies are needed that scaffold the learning as well as encouraging discussion about what is occurring and why. The following teaching approaches can be employed to support learners when working with code.

Stage	**Activities**	**Resources**
Predict	Review code in pairs and predict the output Group discussion on larger examples of code Black box flow charts where learners predict what code is in the box Predict the purpose of a function based on the input/output	Pre prepared code examples
Run	Run any code examples provided	Pre prepared code examples
Investigate	Comment up some provided code Find a specific technique in the code and highlight it Trace tables Turn a flowchart into code and vice versa	Intentionally buggy code Pre prepared code examples
Modify	Bug fix some broken code Fix a logic error Improve an algorithm Add functionality to a program Modify a programme to be functional Re-factor an inefficient program	Pre prepared code examples
Make	Create a program to a brief Add functionality to a program Create a function based on arguments from other functions	Challenge briefs Pre prepared code examples