Open Coding and Robotics curricula and educator training

Version 1.2

Prepared by Andrew Moore of Neil Butcher and Associates

Cairo office 2022

Contents

Open licence

Glossary

Breadboard

Coding

Robotics

Open education

Open Educational Resources

Open Source

Potentiometer

PIR Sensor - Passive infrared sensor.

Python

STEAM

Ultrasonic Sensor Distance Measuring Module

UNESCO

Introduction

Since 2018 the UNESCO Cairo office has been piloting the development of coding and robotic skills amongst youth in Egypt. This work has to date revolved around the use of proprietary training materials and platforms. The associated costs of the training materials, physical components and intellectual property are restricting wider dissemination of the training initiative. As UNESCO currently champions open education solutions to improve access to quality education, the Cairo office is investigating how the use of Open Educational Resources (OER), Open-Source tools and low cost components might improve access to coding and robotics education.

Prioritising coding and robotics education is a new trend in developing world nations. The conundrum coding and robotics curriculum developers need to solve is what to include and what to leave out in this vast field. There are numerous programming languages. There are many assembly options and platforms. Essential, however, is the need to integrate theory and physical resources in such a way as their mutual dependence becomes apparent. It can be argued the successful integration of theory and practice in popular proprietary solutions, such as Arduino and Lego, is what has made them attractive to educators and students alike.

This curriculum aims to identify the scope of an introduction to the subjects of Coding and Robotics aimed at youth who have not had exposure before to coding and robotics. The curriculum does not claim to offer comprehensive coverage of the subject. Its focus is specific, foundational, and would benefit by being incorporated into, or linked to, national STEAM curricula, rather than being used in isolation. However, it does provide a solid grounding in both the knowledge sets and skills needed to code and build programmable devices. The associated learning guide that accompanies this curriculum provides an integrated, cost effective, learning pathway designed to achieve the curriculum objectives.

The UNESCO Egyptian projects identified that reskilling educators with coding and robotics knowledge and skills is a prerequisite for a successful programme. This curriculum document also identifies open, free (or at least cost-effective), training options to support the open coding and robotics curricula.

This document proposes a coding robotics curriculum with different approaches for different age groups centred around available OER and open-source and/or free tools and cost effective training options for educators.

Principles

A good coding and robotics curriculum should …

- Offer all students an opportunity to learn with minimal barriers to entry, particularly cost.
- Provide students with knowledge and a skill set that can lead directly to employment opportunities.
- Provide a learning sequence that becomes ever more sophisticated yet aligned to student maturation.
- Ensure teachers are skilled and knowledgeable about the latest technologies and programming languages.

To achieve these ends this coding and robotics curriculum document describes a learning approach that…

- Uses open educational resources (OER) to provide free access to quality learning resources, tutorials and reference materials.
- Uses open-source and/or free coding and robotics tools and platforms to provide cost-effective access support the writing of code.
- Progresses students from easy-to-use block-based coding languages to more sophisticated line-based coding.
- provides students, after completing the learning, experience with using a high level coding language that can lead to employment opportunities.
- Uses components, including microcontrollers and other electrical and physical parts, that can be accessed easily and cheaply.
- Identifies where teachers might look for cost effective professional development opportunities to reskill and grow.

Desired learner competencies

Teaching coding and robotics has the broad aim of guiding and preparing learners to solve problems, think critically, work collaboratively and creatively, function in a digital and information-driven world, apply digital and ICT skills and to transfer these skills to solve everyday problems and its possibilities[1].

Through coding and robotics learners are exposed to a range of knowledge, skills and values that strengthen their:

- aesthetic, creative skills and cognitive development, knowledge through engaging with design processes
- logical, computational and critical thinking skills
- knowledge of digital technologies and the acquisition of skills to use them.
- understanding of the relationship between people and the environment, awareness of social relationships, and science subjects.
- Interpersonal and social skills when working collaboratively with peers.

Curriculum design principles

In designing a curriculum to achieve these broad competencies mentioned above, a good coding and robotics curriculum should …

- Offer all students an opportunity to learn with minimal barriers to entry, particularly cost.
- Provide students with knowledge and a skill set that can lead directly to employment opportunities.
- Provide a learning sequence that becomes ever more sophisticated yet aligned to student maturation.
- Ensure teachers are skilled and knowledgeable about the latest technologies and programming languages.

To achieve these ends this coding and robotics curriculum document describes a learning approach that…

- Uses open educational resources (OER) to provide free access to quality learning resources, tutorials and reference materials.

[1] As described in the South African, Curriculum and Assessment Policy Statement Grades 7-9 for Coding and Robotics. (see 2.2)

- Uses open-source and/or free coding and robotics tools and platforms to provide cost-effective access support the writing of code.
- Progresses students from easy-to-use block-based coding languages to more sophisticated line-based coding.
- provides students, after completing the learning, experience with using a high level coding language that can lead to employment opportunities.
- Uses components, including microcontrollers and other electrical and physical parts, that can be accessed easily and cheaply.
- Identifies where teachers might look for cost effective professional development opportunities to reskill and grow.

Integration of coding and physical resources

This curriculum, and the associated learning guide, are centred around the following open and/or cost effective coding and robotics resources. The selection was made to ensure that the coding fully integrated with the development of physical devices. It is important that both are integrated and support each other. However, coding tools and physical components can be swapped out as per educators experience and/or access to resources. It needs to be re-emphasized that an integrated approach must be maintained.

A] Coding resources

Languages

- Scratch
- Python language (MicroPython)

Coding platforms

- Pipermake
- Trinket
- Thonny

B] Physical resources

Computers and Internet

- A personal digital device – e.g. PC, laptop or tablet.
- An internet connection
- A micro-USB connector cable

Microcontroller and components

- Raspberry Pi Pico Basic Kit - with Pico and pre-soldered header
    - 1 x Breadboard 16.5x5.5cm (830 Holes)
    - 1 x Buzzer-PCB Mount
    - 1 x Finger Adjust Preset Potentiometer 10K
    - 2 x LED 5mm Red
    - 2 x LED 5mm Green
    - 2 x LED 5mm Yellow
    - 2 x LED Super Bright 5mm Blue
    - 2 x PIR Sensor
    - 3 x 6x6x1 Push Button 4Pin
    - 5 x Resistor 0.25W 5% (330R)

  - 10 x Male to Female Jumper Wire
  - 20 x Male to Male Jumper Wire
  - 1 x RPi Pico with pre-soldered header
- 2 x TT-02 DIY Car Model TT Motor w/ Wheel
- 1 x HC-SR04 Ultrasonic Sensor Distance Measuring Module

Justification

As of the compiling of this document in July 2022 the most cost effective combination of coding and robotics platforms, resources and tools were centred around the Raspberry Pi Pico microcontroller and the use of MicroPython coding language. However, the field is dynamic and constantly changing and so this relationship between hardware and coding platforms needs to be revisited regularly. The offering of other popular platforms should be monitored, including but not limited to, Audrino, Lego, BBC Micro:Bit.

The Raspberry Pi Pico microcontroller was selected because…

- It is very cheap compared to alternatives
- It was released in 2021 with a powerful chip the RP2040[2]
- As it is a microcontroller, unlike single boards, and consumes less current
- It is breadboard friendly
- It officially supports MicroPython and C++ but also CircuitPython and block-based programming is possible with Pico Piper
- It can also be integrated into the Arduino ecosystem and benefit from additional libraries for modules and sensors
- As of 2022 the Pico is supported by multiple open and/or free, IDE and OER to support teaching and learning.

For a list if the pros and cons of the Raspberry Pi Pico see Getting Started With Raspberry Pi Pico - Electronic Resources - Electronics-Lab.com Community

Cost analysis (as of July 2022)

Item	Cost
Scratch	Free (CC BY-SA 2.0) https://scratch.mit.edu/ Offline version
MicroPython	Free (MIT Licence) http://micropython.org/download/
Pipermake	Free MicroPython/RP Pico tutorials https://make.playpiper.com/
Trinket Code	Free basic coding platform https://trinket.io/
Thonny IDE	Free coding platform https://thonny.org/
Raspberry Pi Pico / wifi version	$4.00 / $6.00 https://www.raspberrypi.com/products/raspberry-pi-pico/

[2] Dual-Core ARM Cortex M0+ processor

Textbook Malfacree, G & Everard, B. (2021). *Get Started with MicroPython on Raspberry Pi Pico*.	Free digital PDF (CC BY_NC_SA 3.0) on dbooks.com
RP Pico Basic Kit (with Pico, pre-soldered pins, and project components as described above, except wheels)	$11.00
Motorised wheelset (2xTT-02 DIY Car Model TT Motor w/ Wheel and 1 caster wheel)	$9.00
AA 12V Battery pack and 4 x AA batteries	$1.00 + $4.50 = $5.50
HC-SR04 Ultrasonic Sensor Distance Measuring Module	$1.50

Note: It is possible to access the resources required for the Coding and Robotics Curriculum (12–17 years), described below, for $27.00 per student. In addition, however, students will also need access to a digital device and connectivity to complete the curriculum activities.

Scope of the Curriculum

This curriculum is aimed at teachers and students who need to identify where coding and robotics might be incorporated into an existing curriculum or added on as an extracurricular. In addition, teachers will need to adapt it as necessary to align with the particular context and environment they find themselves, and their learners, studying in.

It is divided into two parts. Curriculum A is aimed at young students anywhere between 8-12 years of age and Curriculum B for students 12-17 years and older of age. Both parts are divided into two separate sections, 1] Coding and 2] Robotics/Physical programming. Each section contains approximately 20 hours of study.

A] Curriculum statements for 8–12 years		
1	Coding	20 hours
2	Robotics	15 hours
B] Curriculum statements for 12–17 years or older		
1	Coding	17.5 hours
2	Robotics	20.5 hours
	Total	73 hours

However, the coding and robotic sections are not exclusive as coding and robotics sections are meant to merge and extend each other. Below is a breakdown of the 4 sections identifying what should be covered and what open-source tools and OER are available to support the acquisition of each specific competency.

See the accompanying *Coding and Robotics Learner's Guide* for a suggested learning pathway that integrates the free and open resources into a logical sequence for learners.

A] Coding and Robotics curriculum statements for 8–12 years

1] Coding – Introduction to programming with block-based programming (Scratch)

(See the accompanying Learner's Guide to see how the learning is sequenced and the curriculum competencies integrated with each other)

A1	Recommended time: 20 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and/or Open Educational Resources (OER)
A1.1	**Orientation to the programming interface** (240 minutes)	• Can access and interact with the Scratch interface. • Uses built in libraries such as 'Looks' and 'Sound' • Can join text • Uses simple loops to create movement	• MIT Scratch (Online Interface)	• Space Talk project on RPF • Catch the bus project on RPF • Surprise animation project on RPF • Find the bug project on RPF • Chatbot (Join text/strings) project on RPF
A1.2	**Debugging** (60 minutes)	• Can identify where there are errors in the program and fix them	• MIT Scratch (Online Interface)	• Surprise animation project on RPF
A1.3	**Variables** (120 minutes)	• Can create, name and delete variables. • Can store data in a variable • Can create different types of variables (text and number)	• MIT Scratch (Online Interface)	• Drum star (variables) project on RPF • Catch the dots (variables) project on RPF
A1.4	**Mathematical operators and conditional statements** (130 minutes)	• Can use the 'Greater Than' and 'Less Than', AND, OR, NOT symbols appropriately	• MIT Scratch (Online Interface)	• Numerical Expression Evaluation with Basic Operations | CK-12 Foundation (ck12.org) • Order of Operations, Part One • Puzzle room project on RPF
A1.5	**Control Blocks I** (240 minutes)	• Can use the following blocks appropriately, Join (operator), Ask	• MIT Scratch (Online Interface)	• Grow a dragonfly project on RPF • Boat race (goto x,y) project on RPF • Beat the goalie (touching) project on RPF

A1	Recommended time: 20 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and/or Open Educational Resources (OER)
		(sensing), Touching, If Then, Go to (x,y), broadcasting.		
1.6	**Control Blocks II** (180 minutes)	• Can use the following blocks appropriately, If Then Else, Random, Repeat Until, Broadcasting, Wait, Loops, Repeat and nested if statements	• MIT Scratch (Online Interface)	• Broadcasting spells (broadcasts) on RPF • Dodgeball (if, then, else) project on RPF • Grow a dragonfly (Random) project on RPF • Puzzle room project on RPF • Drum star project on RPF • Lost in space (Loops) project on RPF • Ghostbusters (Random) project in RPF
1.7	**Lists** (180 minutes)	• Can create a list • Delete a list, • Store data in a list and • delete items from a list	• MIT Scratch (Online Interface)	• Memory (Lists) project on RPF • Create your own world (Lists) project on RPF • Username generator (Lists) project on RPF

2] Robotics (Physical Programming)

This section will require students to have access to a physical programming kit, ideally the Raspberry Pi Pico Basic Kit (approx. US$12.00[3]) that uses the MicroPython language. [Alternatives include Arduino Nano v3 Development Kit (approximately $18.00) or the BBC Micro:bit v2.2 Starter kit (approx. $46.00)] Also see the accompanying learning guide to see how the learning is sequenced and the curriculum competencies integrated with each other.

A2	Recommended time: 15 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free or Open Educational Resources (OER)
A2.1	**Introduction to the field of robotics** (30 Minutes)	• Can identify the characteristics of a programmed device		• Types of Robots, Explained for Beginners with Tips, History, Learning, Resources (CC BY) Types of Robots, Explained for Beginn…
A2.2	Introduction to physical computing	• Can identify a microcontroller and its components	• Raspberry Pi Pico • Piper Make (playpiper.com)	• Getting Started Educator Resource (playpiper.in) on Playpiper • Silly Stories (playpiper.in) on Playpiper

[3] Prices quoted here are based on availability in South Africa. These will vary from region to region.

A2	Recommended time: 15 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free or Open Educational Resources (OER)
	(180 minutes)	● Describe inputs, processing, and outputs ● Can navigate and use the block-based programming user interface ● Can communicate via code to the microcontroller		● Guess My Number (playpiper.in) on Playpiper
A2.3	**Physical programming components using block-based code I** (240 minutes)	● Can use code to interact with Light Emitting Diodes (LEDs), buttons.	● Raspberry Pi Pico ● Piper Make (playpiper.com)	● Blink Educator Resource – Piper (playpiper.in) uses LED ● Traffic Light Educator Resource – Piper (playpiper.in) uses multiple LEDs ● Reaction Game – Piper (playpiper.in) uses buttons and LEDs ● Tally Educator Resource – Piper (playpiper.in) uses buttons
A2.4	**Physical programming components using block-based code II** (180 Minutes)	● Can use code to interact with buzzers, ultrasonic and infrared sensors,	● Raspberry Pi Pico ● Piper Make (playpiper.com)	● Security Zone – Piper (playpiper.in) uses ultrasonic sensors and buzzers ● Ultrasonic Drum – Piper (playpiper.in) ● uses ultrasonic sensors and buzzers ● Educator Resource Servo Bug – Piper (playpiper.in) Uses a servo
A2.5	**Final Project** (270 minutes)			

B] Coding and Robotics curriculum statements (12-17 years or older)

1] Coding – Introduction to programming with Python

(See the accompanying Learner's Guide to see how the learning is sequenced and the curriculum competencies integrated with each other)

B1	Recommended time 17.5 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and Open Educational Resources (OER)
B1.1	**Introduction to line-based coding - Computers, algorithms and programs** (60 minutes)	• Can state the differences between block-based and line-based coding and the potential advantages of each.		• CSER. From visual to general programming From visual programming to general-purp… (CC BY) • What is coding? What is Coding? • Why Python? Why Python is a Great First Language – Trinket Blog
B1.2	**Orientation to the Integrated Development Environment (IDE) for Python** (180 minutes)	• Can install and/or access an appropriate IDE to support the writing of code using the Python language • Can distinguish between algorithms, programmes, variables and assignments	• Thonny, Python IDE for beginners (to be used if you are also using Raspberry Pi hardware) • Trinket (A browser-based IDE for Python and Python 3)	• See Chapter 2 - Programming in Python - Thonny activities (Get Started with MicroPython on Raspberry Pi Pico.pdf - Free download books)
B1.3	**Basic operators and functions** (180 minutes)	• Can devise mathematical operations using the order of operations • Can use mathematical functions such as square root, exponents, round and random)	• Thonny, Python IDE for beginners • Trinket	**Operators** • Wikipedia. Order of operations - Simple English Wikipedia, the free encyclopedia • Order of Operations – Programming Fundamentals (rebus.community) • Basic Operators - Learn Python - Free Interactive Python Tutorial • Python Operators - After Hours Programming **What is a function?**

B1	Recommended time 17.5 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and Open Educational Resources (OER)
				• CSER. Functions. Introduction to functions (CC BY) **Create your own functions tutorial** • Python 3 - Functions • Python Full Course 🐍 -Learn to code … Built-in functions • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed2c90a0001dc1a42868d71020925 • Abstracting Code with Functions (cmu.edu) • Functions – Programming Fundamentals (rebus.community) • Python Full Course 🐍 -Learn to code … • Python Full Course 🐍 -Learn to code …
1.4	**Variables, types and lists** (180 minutes)	• Can evaluate expressions and execute basic statements (also known as commands) using a Python interpreter. Can distinguish between different types of variables: Strings, integers, float and Boolean • Can create, store and retrieve data in lists.	• Thonny, Python IDE for beginners • Trinket	• https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed2c70a0001dc1101665b8433ae51 • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed2c80a0001dc12364ce530bc276b • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3440a0001dc483b2d42ff1c71a3 • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3450a0001dc164d372dbcaf9791 • Variables and Assignment Statements (cmu.edu) • Variables and Types - Learn Python - Free Interactive Python Tutorial

B1	Recommended time 17.5 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and Open Educational Resources (OER)
				• String Formatting - Learn Python - Free Interactive Python Tutorial
1.5	**Iterations / loops** (180 minutes)	• Can write a 'for' loop • Can write a 'while' loop • Can nest a loop inside loops.	• Thonny, Python IDE for beginners • Trinket	Loops • Loops - Learn Python - Free Interactive Python Tutorial • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3120a0001dc4d389b08514f9dc2 • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3130a0001dc6dc054a844f38a2c • Loops – Programming Fundamentals (rebus.community) 'For' Loops • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3200a0001dc7bc24ce10956015a • Python Full Course 🐍 -Learn to code tod… 'While' Loops • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3340a0001dc65a298efeafff951 • Python Full Course 🐍 -Learn to code tod… Nested • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3570a0001dc26cda8d0d6a0aff0 • Python Full Course 🐍 -Learn to code tod…
1.6	**Decisions / conditional statements**	• Design and code algorithms involving decisions	• Thonny, Python IDE for beginners	Conditions

B1	Recommended time 17.5 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and Open Educational Resources (OER)
	(180 minutes)	(Boolean-valued expressions)	• Trinket	• Conditions - Learn Python - Free Interactive Python Tutorial • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3650a0001dc25cd55019a9c9bad • Python Full Course 🐍 -Learn to code tod… Boolean • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3670a0001dc230feaae13856529 If..else • https://oli.cmu.edu/jcourse/workbook/activity/page?context=9deed3680a0001dc25b2081650e5c099
1.7	**Data Flow Diagrams (DFD)** (60 minutes)	• Can decipher and create data flow charts that include Start, Stop, Process, Flow, Decisions using logic gates (AND, OR, NOT)	• diagrams.net (A tool to create flow diagrams with a library of conventional symbols) • NCH https://www.nchsoftware.com/chart/index.html • Creately • https://creately.com/lp/data-flow-diagram-software-online/	• CSER. An Overview of Flowcharts An overview of flowcharts (CC BY) • Flowcharts – Programming Fundamentals (rebus.community)
1.8	**Coding, robotics and artificial intelligence** (30 minutes)	• Can appreciate the intersection between coding, robotics and artificial intelligence (AI)		• What is AI and Robotics? What is Artificial Intelligence (A.I.) and Ro… • Three big ethical concerns for AI

B1	Recommended time 17.5 hours	Competencies: Students ...	Open-source and/or free tools for student use.	Free and Open Educational Resources (OER)
		● Is aware of some of the ethical issues of developing AI and Robotics		● The three big ethical concerns with artific…

2] Robotics (Physical Programming)

This section will require students to have access to a physical programming kit, ideally the Raspberry Pi Pico Basic Kit (approx. US$11.00[4]) that uses the MicroPython language. [Alternatives include Arduino Nano v3 Development Kit (approximately $18.00) or the BBC Micro:bit v2.2 Starter kits (approx. $46.00).] Also see the accompanying learning guide to see how the learning is sequenced and the curriculum competencies integrated with each other.

B2	Recommended time: 20.5 hours	Competencies: Students ...	Free or open-source tools for student use.	Free or Open Educational Resources (OER) study materials
B2.1	**What is Robotics?** (30 Minutes)	● Can describe the field of robotics research and development.		● https://youtu.be/uv6SNObmGhw
B2.2	**Microcontroller boards, chips, pins and the IDE** (180 minutes)	● Can describe components on a single board computer and has connected the IDE to communicate with the board.	● Raspberry Pi Pico or alternative ● Thonny, Python IDE for beginners	● GSMPRPP – Chapters 1-3 ● LED Firefly project on RPF
B2.3	Common electrical components (180 minutes)	● Can differentiate and describe the function of various common electronic components: Push button switches, light emitting diodes, resisters, I2C displays, piezoelectric buzzers, potentiometers,	● Raspberry Pi Pico or alternative	● GSMPRPP – Chapter 3

[4] Prices quoted here are based on availability in South Africa as of June 2022. These will vary from region to region and over time.

B2	Recommended time: 20.5 hours	Competencies: Students ...	Free or open-source tools for student use.	Free or Open Educational Resources (OER) study materials
		passive infra-red sensors (PIR) as well as common physical computing tools such as a breadboard and jumper wires.		
B2.4	**Circuit Construction** (60 minutes)	• Can design a simple circuit using conventional symbols • Understand the implications of suppling power to a physical device	• PhET Circuit Construction Kit: DC	• PhET Circuit Construction Kit Walk through
B2.5	**Programming physical** components I (180 minutes)	• Can create a programme to control electrical components. (LED, switches and buzzers) • Can use programming libraries (for MicroPython particularly, machine, utime, urandom and _thread)	• Raspberry Pi Pico using LEDs, switches and piezoelectric buzzers	• GSMPRPP – Chapter 4-6 (Traffic Light and Reaction Game projects)
B2.6	**Programming physical components II** (180 minutes)	• Can create a programme to control electrical components. (Sensors, LED, and buzzers) • Can use programming libraries	• Raspberry Pi Pico using PIR sensors, LEDs and piezoelectric buzzers	• GSMPRPP – Chapter 7 (Burglar alarm project) • Sound machine (Buzzer) project on RPL
B2.7	**Programming physical components III** (180 minutes)	• Can create a programme to control electrical components. (Potentiometers, LEDs) • Can use programming libraries	• Raspberry Pi Pico using sensors, LEDs and potentiometers	• GSMPRPP – Chapter 8 (Temperature gauge project) • Beating heart (potentiometer) project on RPF

B2	Recommended time: 20.5 hours	Competencies: Students ...	Free or open-source tools for student use.	Free or Open Educational Resources (OER) study materials
B2.8	**Programming physical components IV** (120 minutes)	• Can use a file system to store and retrieve data (data logging) for later use	• Raspberry Pi Pico	• GSMPRPP – Chapter 9 (Data logging project)
B2.9	**Programming physical components V** (120 minutes)	• Can create a programme to control electrical components to communicate using I2C (LCD Display) • Can use programming libraries	• Raspberry Pi Pico using LCD display	• GSMPRPP – Chapter 10 (Digital communication protocols project)
B2.10	**Project: Create a robot** (300 minutes)	• Design and create their own sensory gadget ideally a device what moves like a robot	• Raspberry Pi Pico	• Projects | Computer coding for kids and teens | Raspberry Pi • How To Build A Simple Raspberry Pi Pico Robot | Tom's Hardware (tomshardware.com) • Build a Motion-Activated Guard Robot (BlueBot Project #1) | Science Project (sciencebuddies.org) Additional inspiration resources Raspberry PI Pico - jpralves.net 10 Projects for a Raspberry Pi Pico 21 Best Raspberry Pi Pico Projects You Must Build in 2022

Teacher training options / Overlap with the UNESCO ICT CFT

Teaching coding and robotics aligns well with the UNESCO ICT competency framework for teachers (CFT). The ICT CFT identifies technology skills all teachers should aspire to. Version 3 of the framework specifically identifies 'coding' as a significant technological innovation that will shape our futures, and hence contemporary education should ensure both educators and students are aware of its role.

> "Underlying all computer programmes are algorithms, which specify how a task is to be done. Algorithmic thinking – also called computational thinking – underlies computer science, and there has been a growing movement on algorithmic thinking in schools. Coding is taught so that students are exposed to the skills needed to develop computer applications. Just as students learn to write to be able to organize, express and share ideas, learning to code teaches students how to organize, express and share ideas in new ways, in a new medium". (p18).

Aligned ICT CFT Competencies

The following UNESCO ICT CFT specific objectives demonstrate how the teaching of coding and robotics align well with, Knowledge Creation, the top level of the competency framework.

> KC.2.a – [Teachers can…] Analyse the curriculum standards to identify opportunities where students can master Knowledge Society skills and complex cognitive skills, considering learning styles, abilities and sociolinguistic skills. (The 'example activities' section specifically mentions coding: 'If students are learning to code, identify where coding projects would link the complex cognitive skills with knowledge society skills.')
> KC.2.c - [Teachers can…] Guide students to make appropriate ICT choices to achieve curriculum standards that support reasoning, planning, reflection and knowledge building.
> KC.3.a. - [Teachers can…] Explicitly model their own reasoning, problem-solving and knowledge creation while teaching students.
> KC.6.c. - [Teachers can…] Continually evaluate and reflect on professional practice to promote innovation and improvement.

Specific Educator Training opportunities that align with this curriculum's objectives

Examples of teacher training courses to support Coding and Robotics include

- *Introduction to programming with Scratch* on FutureLearn offered by TeachComputing & Raspberry Pi Foundation (8 hours – Online – Free)
- *Scratch to Python: Moving from Block- to Text-based Programming* on FutureLearn offered by Raspberry Pi (4 weeks - $44.00 or 7 days free)
- *Teaching Physical Computing with Raspberry Pi and Python* on FutureLearn offered by Raspberry Pi (6 weeks - $44.00 or 7 days free)
- Programming 101: An Introduction to Python for Educators on FutureLearn offered by Raspberry Pi (8 notional hours. - $44.00 or 4 weeks free)
- *Learn Python Basics* on OpenClassrooms (6 hours – Free - CC BY-SA)
- *Learn Programming with Python* on OpenClassrooms (12 hours – Free - CC-BY-SA)
- *Introduction to Programming with Python* on FutureLearn offered by FutureLearn. (4 weeks - $59.00 or 4 weeks free)
- *Principles of Computation with Python — Open & Free* on Open Learning Initiative (Free)

Open Education Resources and Open-Source Tools

Key Learning resources

- Busbee, KL, & Braunschweig, D. (nd). *Programming Fundamentals: A modular Structured Approach (2nd Edition)*. Available on PressBooks here. (CC BY-SA)
- Halfacree, G & Everard, B. (2021). *Get Started with MicroPython on Raspberry Pi Pico*. Published by Raspberry Pi Press. Available online here. CC BY-NC-SA 3.0)
- Hessel, T. (2020). *PhET Circuit Construction Kit HTML5 Walkthrough*. Available on YouTube. CC BY
- Open Learning Initiative. (nd). *Principles of Computing - Introduction to programming*. Carnegie Mellon University. Available online here. CC BY-NC-SA
- Raspberry Pi Foundation. (2022). *Sensory Gadget*. Available online here. (CC BY-SA 4.0)
- Suzuki, J. (2019). *Order of operations Part I* on YouTube here (CC BY)

Additional Online projects

- Code Club World (Raspberry Pi Foundation) for Scratch/Python https://codeclubworld.org/
- João Alves Raspberry PI Pico Projects - jpralves.net (CC BY-NC-ND)
- Les Pounder How To Build A Simple Raspberry Pi Pico Robot | Tom's Hardware (tomshardware.com)