



Project Number: 101006468

Project Acronym: PAFSE

Project title: Partnerships for Science Education

EDUCATIONAL SCENARIOS

UMINHO UNIVERSITY (UM)



Universidade do Minho

JULY 2023



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101006468.

1. Specifications for an educational scenario on the topic of “Low-code development environments – Level 1 (Basic)”

Title of Educational Scenario: Connecting students to IT using low-code development environments to promote public health and digital literacy – Level 1 (Basic)

Topic in School Curriculum: Block programming / Health subject by teachers choice

School Subject: ICT classes/Biology/F.Q classes/Health Education classes (Interdisciplinarity¹)

Main resource: [MIT App Inventor](#)

Grade level: 7th grade (+/- 12-13 years old students)

Context and its relevance to public health education

In a world of immediacy, anything less than digital handiness results in lost opportunities and innovation disregard. In fact, technology is enriching humans’ lives, improving access to information, and revolutionizing how people teach, learn and work in the 21st century. Thus, learning how to code is a contribute to the process of developing problem-solving skills central to success in STEM (Science, Technology, Engineering, Mathematics) curricula and careers. It can also aid students in the development of innovative solutions that benefit the health of their community.

Block-based coding or programming is based on a drag-and-drop learning environment, where programmers use coding instruction, called “blocks”, to construct animated stories, games and other types of multimedia content. It’s an entry-level activity, where students can gain a foundation in computational thinking through visuals as opposed to coding that is based in text, making it more interesting and viable to use as an educational resource.

The educational scenario assists (mainly) ICT teachers in exploring how low-code environments can positively impact education and increase digital and public health literacy. The learning experience supports youths in understanding how STEM may contribute to create new and revolutionizing solutions to the healthcare market, as well as stimulate their creativity, decision-making and problem-solving competences and enhance their technological and communication skills. In the teaching-learning sequence, hygiene habits (e.g.: oral health, sleeping habits) are explored in connection with appropriate tools for multimodal content creation (quizzes, infographics, presentations, etc.) that promote appropriate exposition of this relevant public health topic to other members of the society.

¹ Integrating knowledge and methods from different disciplines, using a real synthesis of approaches.

Estimated Duration

Variable, depending on ICT teachers weekly schedule (normally 40 min. per week)

Estimate (based on pilot experience): At least 4 classes of 40 min. (lesson 1- 4) and 3 sessions of 40-45 minutes for supplementary learning activities and school project (session 4 – session 7)

Classroom organization requirements

Classroom ergonomics:

- Create a space that is adaptable to the learning experience instead of having the learning experience adapted to the space (Bayse, 2015).
- Focused on the training of required skills and collaborative work.
- Students are encouraged to be creative and active “prosumers” (producers+consumers).
- Teachers are required to provide high support to students, without compromising students autonomy.

For the learning-through-teacher lessons, students will work alone/in groups (preferably) and should have access to:

- An ICT classroom with regular functioning computers.
 - [Setup - MIT App Inventor](#);
 - [System Requirements - MIT App Inventor](#);
 - [App tester - MIT App Inventor](#);
 - [Pre-setup \(Tech and Networking Specialists\) - MIT App Inventor](#).
- An internet connection.
- A Gmail account (to log in in MIT App Inventor);
 - [Accounts and devices - MIT App Inventor](#).
- Any android device.

To carry out the research project, students will work in groups and the same equipment is required, as well as an open, curious, and creative mind.

Observations:

- No prior downloading of software is required.
- Students are welcome to use their own computers.
- Each student should have their own email account.
- App Inventor offers the ability to develop using the Android emulator that shows up in a window on the computer screen if the students don't have an android device.

However, using the emulator isn't as good as a physical device, because students can't carry their apps around with them and some features might not be present.

- The navigator “Internet Explorer” is not supported.
- MIT App Inventor works as a cloud, therefore everything is stored online.

Prerequisite knowledge and skills

- Basic IT and ICT notions.

General content glossary

- **IT.** **IT** (Information Technology) is the study, design, development, application, implementation, support, or management of computer-based information systems. (Source: [Code Academy](#))
- **ICT.** Information and communication technologies (**ICT**) is defined as a diverse set of technological tools and resources used to transmit, store, create, share or exchange information. These technological tools and resources include computers, the Internet (websites, blogs and emails), live broadcasting technologies (radio, television and webcasting), recorded broadcasting technologies (podcasting, audio and video players, and storage devices) and telephony (fixed or mobile, satellite, visio/videoconferencing, etc.). (Source: [UNESCO](#))
- **Low-code.** A **low-code** platform allows app development through the use of a graphical user interface (GUI) rather than traditional hand-coding. In other words, it is a type of visual software development environment that allows developers to drag and drop application components, connect them together and create mobile or web apps with little to no code. (Source: [Techtarget](#))
- **Block coding.** **Block coding** is a process used in computer programming where text-based software codes change to a visual block format to create animated games, characters, and even stories. With block coding, kids can learn the basics and foundational concepts through visuals instead of text-based coding. (Source: [Codingal](#))
- **Algorithm.** An **algorithm** is a detailed step-by-step instruction set or formula for solving a problem or completing a task. In computing, programmers write algorithms that instruct the computer how to perform a task. When you think of an algorithm in the most general way (not just in regards to computing), algorithms are everywhere. A recipe for making food is an algorithm, the method you use to solve addition or long division problems is an algorithm, and the process of folding a shirt or a pair of pants is an algorithm. (Source: [Tynker - Coding for Kids](#))

- **Programming language.** A **programming language** is a set of commands, instructions, and other syntax use to create a software program. In other words, it is a language that allows a programmer to tell the computer what to do in a variety of circumstances. Languages that programmers use to write code are called "high-level languages." This code can be compiled into a "low-level language," which is recognized directly by the computer hardware. (Source: [Techterms](#); [Ageuk](#))
- **Event-driven programming.** **Event-driven programming** is a programming paradigm in which the flow of program execution is determined by *events* - for example a user action such as a mouse click, key press, or a message from the operating system or another program. An event-driven application is designed to detect events as they occur, and then deal with them using an appropriate *event-handling procedure*. (Source: [Technologyuk](#))
- **MIT App Inventor.** **MIT App Inventor** is an intuitive, visual programming environment that allows everyone – even children – to build fully functional apps for Android phones, iPhones, and Android/iOS tablets. It is an open-source tool that aims to make programming and app building accessible to a wide variety of audiences (educators; researchers; government; etc.) Initially developed by Professor Hal Abelson and his team, App Inventor is managed by members of MIT's Center for Mobile Learning. (Source: [MIT App Inventor](#))
- **IDE.** An **IDE**, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program and develop programs more efficiently. IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging. (Source: [Code Academy](#))
- **User Interface.** The **user interface** (UI) is the look and feel of an operating system. A good interface puts the user first, making commands and access to apps easy to discover. For the programmer, understanding how the interface works and what impact it has on application design is extremely useful. (Source: [O'Reilly](#))
- **Conditional blocks.** Conditionals refer to expressions or statements that evaluate to true or false. If the condition is "true", a particular section of text will be inserted into the message. If the condition is "false", the text will not be inserted. An "ELSE" clause can be included as part of the conditional statement so that a different section of text will be inserted into the message when the condition is "false". (Source: [Isoft](#))
- **Loops** are a way to tell a computer to do something many times in a row. Computers are really good at doing things over and over again, and doing them fast. (Source: [technovationchallenge](#))

- **Lists** - a way to organize multiple pieces of data in App Inventor (Source: [technovationchallenge](#))
- **Index** - a number that tells you where a piece of data is in a list (Source: [technovationchallenge](#))

Pedagogical glossary

- **Constructivism.** Jean Piaget presented the theory of **constructivism**, asserting that knowledge is not simply transmitted from teacher to student, but actively constructed in the mind of the learner. Learners don't receive ideas; rather they create them from their own base of knowledge. Some characteristics of constructivist learning are that it:
 - ⇒ fosters critical thinking;
 - ⇒ creates motivated and independent learners;
 - ⇒ has lessons that include guided discovery, whereby the teachers acts as a guide to the learner, helping to point out inconsistencies in students' thinking. Students build their understanding by resolving these conflicts;
 - ⇒ includes a minimal amount of direct instruction. (Source: [MIT App Inventor](#))
- **Constructionism.** Building from the idea of **constructivism**, Seymour Papert presented his theory of constructionism which suggests that new ideas are most likely to be created when learners are actively engaged in building some type of external artifact that they can reflect upon and share with others. Elements of a constructionist learning environment include:
 - ⇒ a teacher who acts as a facilitator;
 - ⇒ learners who investigate, create, and solve problems;
 - ⇒ learner collaboration;
 - ⇒ learners engaging in authentic tasks;
 - ⇒ opportunity for feedback and multiple opportunities for revision. (Source: [MIT App Inventor](#))
- **Problem-Based Learning.** **Problem-based learning** is one type of constructivist learning theory that can be applied in a classroom setting. It is a method which allows students to learn about a subject by exposing them to multiple problems, so they will be able to construct their understanding of the subject through these problems. Problem-based learning typically:
 - ⇒ begins with problem for students to solve or learn about;
 - ⇒ includes problems that are somewhat ambiguous to mirror the complexity of real life;
 - ⇒ uses an inquiry model;
 - ⇒ requires students to present a conclusion of the problem solving process, but does not necessarily require them to create a product as a result;

⇒ is driven by defined problems. (Source: [MIT App Inventor](#))

- **Project-Based Learning.** **Project-based learning** encompasses Papert's theory of constructionism where students build an artifact as part of the learning process. Project-based learning typically:
 - ⇒ begins with an end product in mind;
 - ⇒ includes production of an artifact, which typically raises one or more problems for students to solve;
 - ⇒ asks students to use or present the product they have created;
 - ⇒ is driven by the end product;
 - ⇒ stresses that content knowledge and skills acquired during the production process are critical to success. (Source: [MIT App Inventor](#))
- **Computational thinking.** The term **Computational Thinking (CT)**, coined by Jeannette Wing in 2006, describes solving problems, designing systems, and understanding human behavior based on the principles of computer science. CT includes analyzing and organizing data, automated problem solving and using it to solve similar problems. Nowadays, it has become necessary to solve complex technological problems. If sufficient background knowledge is available and the necessary new knowledge is acquired through critical thinking, CT may help to solve the problem. It is actually a hybrid of several other modes of thinking, like abstract, logical, algorithmic, constructive and modelling thinking, which summarizes all previous modes for solving the corresponding problem. (Source: [IGI](#))
- **Brainstorming.** Brainstorming is a group problem-solving technique that involves the spontaneous contribution of ideas from all members of the group. (Source: [Merriam-Webster](#))
- **Collaborative learning.** A **collaborative** (or cooperative) **learning** approach involves students working together on activities or learning tasks in a group small enough to ensure that everyone participates. Students in the group may work on separate tasks contributing to a common overall outcome or work together on a shared task. This is distinct from unstructured group work. Some collaborative learning approaches put mixed ability pairs, groups or teams together to work in competition with each other in order to drive more effective collaboration. (Source: [Evidence For Learning](#))
- **Gamification.** **Gamification** of education is a developing approach for increasing learners' motivation and engagement by incorporating game design elements in educational environments. It is often described as the use of game design elements in non-game contexts" (Deterding, Dixon, Khaled, & Nacke, 2011), "the phenomenon of creating gameful experiences" (Hamari, Koivisto, & Sarsa, 2014), or "the process of making activities more game-like" (Werbach, 2014). (Source: [Springer](#))

- **Learning through Storytelling.** **Storytelling** is the vivid description of ideas, beliefs, personal experiences, and life-lessons through stories or narratives that evoke powerful emotions and insights. It represents the use of stories or narratives as a communication tool to value, share, and capitalize on the knowledge of individuals. (Source: [Springer](#))
- **STEM.** **STEM** (Science, Technology, Engineering, and Math) is an integrated, interdisciplinary, and student-centered approach to learning that encourages critical thinking, creativity, collaboration, and design thinking across multiple disciplines. An important role of STEM education is to help students develop skills that will empower them later on in the workplace. This includes helping students develop skills that foster:
 - ⇒ Critical thinking;
 - ⇒ Flexible thinking;
 - ⇒ Data-driven analytical inquiry;
 - ⇒ Design (interdisciplinary) thinking;
 - ⇒ Social responsibility;
 - ⇒ Productivity;
 - ⇒ Leadership;
 - ⇒ Teamwork;
 - ⇒ Collaboration;
 - ⇒ Communication. (Source: [Techopedia](#))
- **Multimodality.** To understand multimodal learning, you first have to know the different modalities and their characteristics.
 - ⇒ Modes are channels of information. They include:
 - Speech
 - Audio
 - Written and print
 - Illustrations

An example is that people learn from images by reacting to visual cues such as photos and graphs. People can also learn from kinesthetics by reacting to tactile cues such as actions and movement. Multimodal learning is teaching a concept using more than one mode (visual, auditory, reading, writing, and kinaesthetic methods). By engaging the mind in multiple learning styles at the same time, learners experience a diverse learning style that collectively suits all of them. Thus it is meant to improve the quality of teaching by matching content delivery with the best mode of learning from the student. (Source: [eLearning Industry](#))

Other connected terms (pedagogical eclecticism):

- Adaptative teaching.
- Personal inquiry.
- Dynamic assessment.
- Crossover learning.
- Navigating knowledge.
- Learning through argumentation.
- Learning from animations.
- Learning to learn.
- Event-based learning.
- Learning for the future.
- Immersive learning.
- Open pedagogy.

Indicative literature:

- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads*, 2(1), 48–54. <https://doi.org/10.1145/1929887.1929905>
- Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(2), 45–73. <https://doi.org/10.1145/273133.274308>
- Bialik, M., & Fadel, C. (2015). *Skills for the twenty-first century: What should students learn?*. Boston: Center for Curriculum Redesign
- Gómez, L. A., & Fischer, C. O. (2017). *Android + Arduino usando o MIT AppInventor: Integrando Android, Arduino e o MIT App Inventor, para implementar a Internet das Coisas (IoT) (Portuguese Edition)*. Novas Edições Acadêmicas.
- Hill, D. R. & Brunvand, S. (2018). *Gamification: Taking Your Teaching to the Next Level: A Guide for Gamifying your Classroom*. In A. Ottenbreit-Leftwich & R. Kimmons, *The K-12 Educational Technology Handbook*. EdTech Books. Retrieved from <https://edtechbooks.org/k12handbook/g>
- Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, 52, 200–210. <https://doi.org/10.1016/j.chb.2015.05.047>

- Romero, M., Davidson, A.-L., Cucinelli, G., Ouellet, H., & Arthur, K.. (2016). Learning to code: from procedural puzzle-based games to creative programming. *Revista del Congrés Internacional de Docència Universitària i Innovació (CIDUI)* (online journal), 3, 1–8. <http://www.cidui.org/revistacidui/index.php/cidui/article/download/944/909>
- Serrat, O. (2017). Storytelling. In: *Knowledge Solutions*. Springer, Singapore. https://doi.org/10.1007/978-981-10-0983-9_91
- Walter, D. (2014). *Learning MIT App Inventor: A Hands-On Guide to Building Your Own Android Apps*. Addison-Wesley Professional.
- Wing, J. (2006). Computational thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. <https://doi.org/10.1145/1118178.1118215>
- Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2017). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education*, 26(4), 1–20. <https://doi.org/10.1080/08993408.2016.1257418>

Competences / Learning Goals

Key Competences

STEM / Personal, social, and learning to learn, literacy, citizenship, responsibility.

Knowledge

Main concept: The role of low-code programming in public health issues

Programming concepts:

- Basic programming expressions and statements.

ICT concepts:

- Programming solutions development and application; low-code development environments; low-code development in public health (*ex: hygiene habits*).

Knowledge - outcome assessment:

1. Understands basic concepts of computational thinking/science.
2. Identifies the principles of low code and block-programming.
3. Recognizes the importance of content creation-involved concepts.

4. Recognizes the meaning and basic use of common statements and expressions common in programming languages.
5. Identifies basic MIT App inventor notions.
6. Justifies why low code is crucial to the future.

Skills (abilities/competences)

General:

- “Creactical skills” (Ohler, 2013) / 21st century key skills:
 - Communication: digital communication; digital literacy; traditional literacy; health literacy; public speaking; argue capability; learn to learn.
 - Collaboration: working in groups towards a goal/to solve a problem/answer a question; collaboration.
 - Critical thinking: perform reasoning and analysis to draw conclusions based on simple systems; decision-making process; problem-solving process; project-based thinking.
 - Creativity: involves initiative, entrepreneurship, taking risks and learning from risks.

Specific:

- Developing, enhancing, and practicing computational thinking and technology-based projects.
- Finding, analyzing, and interpreting multimodal content to map the principles of low-code programming.
- Smoothly expanding the [21st century competences](#).

Skills – outcome assessment:

1. Recognizes basic and appropriate proficiencies necessary for block programming.
2. Can explain pros of adopting multimodal and gamification strategies in education and health.
3. Can partially transform creative ideas into programmable concepts.
4. Uses creativity to explore several basic programming statements and expressions.
5. Feels able to explain the benefits of using low-code development environments for real life problems connected with lifestyles.

Affective/Attitudes/ Behaviour (beliefs)

- Adopting a citizen developer role in society, as well as having social and personal responsibility.
- Pursuing the adoption of critical thinking and problem-solving attitude as an individual.
- Engaging in other basic programming challenges/courses to further develop his/her interest in STEM.

- Adopting attitudes that mitigate public health risks.

Attitudes and behavior - outcome assessment:

1. Believes that technological competence is essential for citizenship and can lead to positive outcomes in the community.
2. Believes that low-code development environments can be vital to solve real-life problems connected with public health and contribute for innovative solutions (since they are fast, cheap, adaptable, and flexible).
3. Believes that health is the most important constituent of life.
4. Believes that digital literacy is vital for any profession in the future.
5. Is committed to improve his/her digital literacy and programming knowledge, as well as influence “STEM adoption” in his/her living environments.
6. Intends to use technology in his/her routine and has a positive attitude towards it.

Learning goals and outcomes

- Uses low-code environments and content creation tools as a creative extension to express ideas and knowledge.
- Correctly uses online tools to create multimodal content.
- Understands the core components of programming and uses them to apply on other block-structured programming applications.
- Clearly exposes the main principles of good hygiene habits, how they influence daily routines and health outcomes.

Assessment methods

- Outcome assessment
 - Quantitative – A questionnaire (in digital format)
<https://ec.europa.eu/eusurvey/runner/STEM-Tecnologia>
<https://drive.google.com/file/d/1HuvJTCmGhPlf32A8hPPWdNueDBO73261/view?usp=sharing>
 - Qualitative - students project: a. basic app/quizz building activity b. additional multimodal resources regarding public health and technological principles.
- Process assessment – *assessment of the teaching-learning sequence* – observation grid: reaching the target audience, and extent; implementation of the scenario as planned; run of the learning scenario as expected/organizational issues to be solved; duration of the teaching-learning sequence; number of people exposed; score for likeability – students (“how fun was it to do”/ how fun would be to do again/ how could it be better).

Content (relevant to learning goals & research topics)

STEM content

- Technical literacy.
- Basic programming expressions.
- Programming oriented math.
- Digital Literacy.
- Communicating science: Healthy lifestyles – hygiene habits.

Non-STEM content

- Teamwork
- Metacognition
- Proactivity
- Multimodality
- Autonomy

Digital Learning Objects (DLOs) and Digital Educational Resources (DERs)

New (developed by PAFSE team):

Given the versatile nature of MIT APP Inventor and the imperative need to let teachers choose their own way of teaching their class, suited to student's needs, the PAFSE team, after discussion with pilot schools, opted by providing general animated, intuitive, and interactive *Noocs*² (*Nano Open Online Courses*), used in the Pilot workshop (all available in a e-me hive: <https://files.e-me4all.eu/s/GpWkoyobw2DYdAp>)

1. Animated, intuitive, and interactive *Noocs*³ (*Nano Open Online Course*) using [genially](#)⁴ concerning MIT App Inventor (Basic) and concepts to communicate science through technological content.
2. Informational multimodal resources (images, videos, sound, etc.) with content, challenges and solutions regarding basic low-code programming.
3. MIT AppInventor app(s) created by the team.

² NOOC are "nano" learning experiences that are specific, targeted to a certain skill and or competency, and can be disseminated in smaller, isolated ways.

³ NOOC are "nano" learning experiences that are specific, targeted to a certain skill and or competency, and can be disseminated in smaller, isolated ways.

⁴ Genially is the world-leader in interactive visual communication using low/no-code. It is an all-in-one online tool to create stunning presentations, interactive images, infographics, gamification, quizzes, breakouts, portfolios, etc. and enrich them with interactivity and animation effects in seconds.

4. Questionnaires – quantitative and qualitative assessment of learnings - Student Interest and Choice in Science, Technology, Engineering and Mathematics (STEM Survey (adapted from Roller et al. 2018 and Faber et al, 2013); Student Interest and Choice in Technology; Informed Consent – Students; Informed Consent – Legal Representatives; Process Assessment – Observation grids, weekly meetings; Photos, Videos, Tasks; Final projects.

Available resources (link):

E-me hive with all resources: <https://files.e-me4all.eu/s/GpWkoyobw2DYdAp>

From other sources/high-quality platforms⁵:

Tutorials and examples to aid teachers prepare and train the students:

- MIT AppInventor:
 - [Teach your students](#)
 - [Setting up your classroom for teaching App Inventor 2](#)
 - [Teaching an app inventor course](#)
 - [Hello Codi! \(app example\)](#)
 - [The MIT App Inventor library: documentation & support](#)
 - [MIT App Inventor - tutorials](#)
 - [MIT App Inventor - Beginner videos](#)
 - [MIT App Inventor - Nooc](#)
- Block-Programming:
 - [Block-based Programming in Computer Science Education](#)
 - [Block coding 101](#)
- Hygiene habits ⁶(personal, environmental, regarding food, mental, COVID-19 related, etc.):
 - [Programa escolar da Colgate](#)
 - [Hygiene for teens: why good habits are important](#)
 - [The importance of teen hygiene](#)
 - [Adolescent hygiene basics](#)
 - [Food hygiene](#)
 - [Show me the science - why wash your hands?](#)
 - [Several types of hygiene](#)
 - [Children's oral health](#)

⁵ The majority will be included in the developed DER.

⁶ The topic is merely a suggestion, teachers are free to explore any health-related topic.

- [Sleeping hygiene](#)
- [Types of hygiene](#)
- [Mental hygiene](#)
- [COVID-19 hygiene](#)
- [How to protect yourself from COVID-19](#)

Observation:

- The PAFSE team provides examples of high-quality platforms that can be used by students to develop their app in the research project. However, teachers are encouraged to choose the resources they see fit, or even leave it up to the students, instigating their creativity and research capacity, since it is such a "mundane" topic. In case of the schools that have science/health teachers involved (in addition to/instead of ICT teachers), this is even more encouraged.

Teaching-learning activities

Principal target:

ICT classes /Biology classes /F.Q classes/Health Education classes (depending on the institution)

7h grade (+/- 12-13 years old students)

At least 4 classes of 40 min. (lesson 1- 4)


ICT teachers integrate other colleagues in the enactment of the scenario, as it aims to be interdisciplinary. The scenario provides the necessary tools for students to explore desirable behavior in an individual and public health perspective.

General note:

- Even if students stumble into some moderate-advanced programming, this educational scenario focuses on the applicability, on learning by doing. Thus, as the education provider, do not worry if they struggle for a bit and you have to step in from time to time, let them explore and learn as much from the platform as possible.
- All the activities/theoretical aspects used in lessons will be available in the DER so you can access any information there.
- The lessons will have "[Bloom's taxonomy](#)" as a background, which pillars are the following:

#bitesizePD

Bloom's Digital Taxonomy



Bloom's taxonomy	Bloom's modified taxonomy	Bloom's extended digital taxonomy	Functional Levels	Activities with digital tools	
		Sharing	Publicly sharing, publishing, broadcasting	Contributing to open social networks, publishing, broadcasting, networking	Higher Order Thinking Skills ↑
Evaluation	Creating	Creating	Designing, constructing, planning, producing, inventing, devising, making	Programming, filming, animating, blogging, video blogging, mixing, re-mixing, wiki-ing, videocasting, podcasting, directing	
Synthesis	Evaluating	Evaluating	Checking, hypothesising, critiquing, experimenting, judging, testing, detecting, monitoring	Blog commenting, reviewing, posting, moderating, collaborating, refactoring, testing	
Analysis	Analyzing	Conceptualizing	Comparing, organising, deconstructing, attributing, outlining, finding, structuring, integrating	Hacking, mashing, linking, validating, reverse engineering, cracking	
Application	Applying	Applying	Implementing, carrying out, using, executing	Running, loading, playing, operating, uploading, sharing with group, editing	
Comprehension	Understanding	Connecting	Interpreting, summarizing, inferring, paraphrasing, classifying, comparing, explaining, exemplifying	Boolean searches, advanced searches, blog journaling, tweeting, categorizing, tagging, commenting, annotating, subscribing	
Knowledge	Remembering	Doing	Recognizing, listing, describing, identifying, retrieving, naming, locating, finding	Bullet pointing, highlighting, bookmarking, group networking, shared bookmarking, searching	Lower Order Thinking Skills ↓



Bloom's Digital Taxonomy by [Fractus Learning](#) is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](#).

Lesson 1: Introduction to key concepts

Learning objective: In the starting session, students will be exposed to various theoretical concepts (discussed in the workshop) they will need to use MIT App Inventor successfully and attain the scenario goals. The activities performed during this first lesson aim to engage students in these topics and explore some preliminary ideas.

Topics to be explored: Content Creation; Storytelling; Gamification; Programming; Computational science.

The teaching-learning script starts with the division of the class in small groups (4-5 people or as the teacher sees fit), in order to stimulate collaboration and capacity of exposing and discussing ideas, which will lead to an icebreaking moment, assessing the preconceptions and misconceptions of the students on the topic of "technology".

⇒ Brainstorming on the questions (No research)

- How important is technology?
- What can we use the computer for?
- Do you use a lot of apps? Do they help you daily?

The group organizes the main ideas to present to the class. In parallel, the teacher writes, on the board, the main ideas of each group concerning each question.

⇒ Taking as a starting point the answers obtained during the icebreaker moment, students will be guided through three questions:

- What is an app?
- What are the key programming/computing principles?
- What do we need to create an app?

⇒ At this point, teachers will discuss these questions in a manner students understand:

- The importance of technology and digital literacy to learn about specific topics.
 - Show different and simple examples of technology outcomes.
 - Give examples of fun digital resources they use daily in all mundane activities.
- The core concepts of programming and programming languages. E.g:
 - Coding means to write code, or to write instructions for a computer.
 - Programming, similarly, means to write code or instructions.
 - Debugging means to check code for mistakes and try to fix errors.
- The main principle of gamification: storytelling (how to make a storyboard).
 - Activity 1: Propose to students the elaboration of a simple storyboard of their weekly morning routine (it doesn't have to be real, let them be creative!) using this simple [template](#) (print it).
 - Pair them up and have them swap the papers and telling the story to their colleague.
- What is MIT App inventor and what will they use it for.
 - Simple run-through the platform.
 - How to access it.
 - Quick setup.
 - The teacher should write simple steps on the board and provide students with a tutorial that they consult at any time.
 - Activity: Each student should successfully access their account in the computer.
 - Explain on what they will work on and attribute and elaborate on the topic “hygiene habits”.

⇒ Questionnaires – quantitative and qualitative assessment of learnings - Student Interest and Choice in Science, Technology, Engineering and Mathematics (STEM Survey (adapted from Roller et al. 2018 and Faber et al, 2013); Student Interest and Choice in Technology

Lesson 2: Discovering coding

Learning objective: In less than an hour, they will have become familiar with block sequences, conditional connections and loops. They may forget them afterwards, but they will have seen and used these three basic concepts in solving a concrete problem.

Topics to be explored: Block-programming principles.

Lesson 2 starts by asking, once again, for students to divide into groups. Collaborative work is highly appreciated since they will play with code, it's way more fun with friends. They can help each other, as well as discuss what they are about to see.

⇒ **Activity 1:** Ask students to open this [website](#), watch the video and complete the tasks (phases of a game).

These games work very well to animate a workshop. Each group is focused on the game, and the passage of levels.

- Help them in the beginning with instructions, clues, triggering and show – along the way – that the blocks are highlighted at the time of execution.
- Tell them to setup the page to PT-PT since it's available, both video and game.

In case there are some fast and very curious minds, you can give them [another example](#), while the group awaits for the others.

⇒ Facilitate a "Turn and Talk" Ask students to share their game, app or final product with other group for feedback. Ask a few groups to share out their experience:

- What did you learn about how apps and games work today?
- How do you feel having had the opportunity to study computer science?

⇒ By the end of the exercise, participants will have seen the three essential control structures of any program. Discuss with them what a script or program is: an ordered sequence of instructions, some of which allow you to control what will happen next: loops and conditional connections.

- You can also point out: that each script begins with an event: Which one? When they pressed the start button. That the progress of the program may depend on the environment, for example the presence of a wall in a labyrinth or lava in Minecraft.

⇒ Celebrate! Give them some [certificates](#) to keep it fun and make them proud of their work.

⇒ Ask them to summarize what they learned today, how they felt, or what they experienced.

⇒ Challenge students to share some aspect of their Hour of Code experience on social media using #hourofcode as a way to lend their voices to this worldwide movement. Students can share their game, images, videos or just their thoughts.

⇒ Give them the used [website](#) in case they got interested and want to explore at home.

Lesson 3: Digging into MIT App Inventor

Learning objective: Students should be able to get familiarized with the MIT App Inventor User Interface and basic components.

Topics to be explored: Consolidate the previously experimented concepts - how the components and blocks work and interact; events and a definition of event-based programming – and learn new ones trace existing code to understand functionality; utilize MIT App Inventor interface to modify existing code; demonstrate ability to include conditionals, lists and iteration.

⇒ The teacher will rapidly explore the User interface of the app in front of the students, hand them this quick [guide](#) and ask if they remember the conclusions from last lesson, then explaining that:

- The implementation of a program is done in two parts:
 - The creation of the user interface and the choice of resources that will be used in the application,
 - then block programming that uses the components (events, properties, procedures) associated with the components defined in the first part.

- App Inventor is often called events-based programming. What this means is that the apps run and function based on reactions to events, like on the Minecraft game. When they click a button, start the app, shake the phone, swipe the phone, enter into a textbox. These are all events. Apps and App Inventor are event driven which means that events need to happen to cause something else to happen. Shaking the phone will cause the phone to play a sound and getting a text message will cause the phone to vibrate. Events cause or drive actions.
- App Architecture includes events but also includes components, event handlers, event types, behavior, and object-oriented programming. App Architecture is extremely important to understanding what an app needs to be built and run.

⇒ Activity 1: Let's keep them learning by doing – Creating the “Making Magic” app, a simple app where a rabbit magically comes out of a top hat.

- Get every student to log onto their account, as they were taught before, and teach them how to make the app (All the info you need is in [this simple 6 minute video](#), give it a try!)
 - [Pdf version](#)
 - [Media library](#)

Note: Do not do part 2 as it contains advanced components.

- Get this opportunity to succinctly discuss how to export/Test apps
- [Assessment activity](#) so they ponder on what they learned so far.

⇒ Now that that is settled, let's step up a little bit more and start introducing conditionals, lists and iterations.

- Talk about Programming Your App to Make Decisions: Using [Conditional Blocks](#)
 - Conditionals refer to expressions or statements that evaluate to true or false.
 - App Inventor provides two types of conditional blocks: if and ifelse, both of which are found in the Control drawer of the Built-In palette.
 - You can plug any Boolean expression into the test slot of these blocks. A Boolean expression is a mathematical equation that returns a result of either true or false. The expression tests the value of properties and variables using relational and logical operators such as the ones shown in the figure below:
 - For both if and ifelse, the blocks you put within the then-do slot will only be executed if the test is true. For an if block, if the test is false, the app moves on to the blocks below it. If the ifelse test is false, the blocks within the else-do slot are performed.
 - Talk about [lists](#)
 - Apps contain data or information. Data is raw facts, information is data processed into usable items. This data can be anything from your location to a high score. All apps need to have ways to store this data. One of these ways is by using lists.
 - A list in Computer Science is essentially what you think it would be: a number of connected items or names written consecutively. You may have a list of names of the contacts in your phone, a list of email addresses from a conference, a list of homework assignments for the week. Apps also use lists and App Inventor makes it easy to do so.
 - Talk about [iteration](#)
 - Think about a screensaver that shows a collection of images. These images are stored in a list. To display all of them one at a time, one after the other, this process is called iteration.
- ⇒ Activity 2: Get every student to complete these [tasks](#) to practice the concepts.
- Have a quick group discussion about the results.
- ⇒ Activity 3: The source code, [HelloAnimal](#), displays a random image and plays a corresponding animal noise for every time the button is clicked. Currently, there are only two images in this app.
- The assignment is to add two additional images and sounds to the app.
 - They will also need to modify the blocks in the Blocks Editor to work for two additional images and sounds.
 - Remember them that if they wanted, they could completely change the theme of the app.
 - Have a quick group discussion about the results.

- ⇒ Assign homework (if possible due to equipment and wi-fi connection, it can be an activity to lesson 4): tell the students to build the app “[Magic 8 ball](#)” (only parts 1 and 2, given that part 3 uses sensors and that’s way too advanced for now!)
- ⇒ **Simpler Alternative to all activities (tested in the Pilot):** Let’s keep them learning by doing
 - Creating the “survey” app, available in “Sessão nº 2_12dejaneiro”:<https://files.e-me4all.eu/s/GpWkoyobw2DYdAp>

Lesson 4 and further: Practice makes perfect

Learning objective: During this lesson and as wrap-up moment on the “app creation” world, students are invited to challenge themselves and take it to the next level. Basic concepts were explored. This is the moment to explore.

Topics to be explored: Intermediate block-programming principles.

- ⇒ Homework of lesson 3: Students present the results of their homework and the entire class discuss the findings.
- ⇒ Activity 1: You will tell the students that, in groups, they will create an app, called “[Paintpot](#)”(only part 1) in [5 steps](#):
 - Step 1: When user touches, draw a circle. When user drags, draw a line.
 - Step 2: Add menu items that let user draw in different colors
 - Step 3: Add menu items that let user draw different size circles.
 - Step 4: Let the user use camera to set background picture
 - Step 5: Add feedback for the user interface
- ⇒ **Alternatives:** [Spaceinvaders](#); [Presidentsquiz](#); [Molemash](#)

Let them be as independent as possible but provide high support when needed.

The first group to get it moving has an advantage in the final research project and can leave the class early. That will motivate them to do best.

Lesson 5-forward: The final countdown

This is the **School Project** described below.

Learning objective: In these final sessions, a recap regarding the major practiced concepts concerning block programming is made, as well as of the used methodologies. In addition, students are prepared to the student’s research project.

Topics to be explored: Mental mapping, Pitch skills

- ⇒ Present the concept of a mental map by showing one representing the lessons.
- ⇒ Activity 1: Ask students to elaborate one, on paper, describing what they learnt.

- ⇒ Aiming to prepare the students for their final project, a presentation called “O meu Pitch em 5 p’s” (My pitch in 5 p’s) will be given, teaching students how to publicly present projects.
- How to communicate science/health using these 5 pillars:
 - “Priorizar” (Prioritize).
 - “Pesquisar” (Research).
 - “Planificar” (Plan)
 - Personalizar (Personalize)
 - Produzir (Produce)
 - Provision of storyboard templates to prepare the pitch.
 - Explanation of what will be evaluated.

These are moments of creative freedom, where students apply the programming concepts learned during previous sessions in a project linked with hygiene habits. There is freedom to work with the MIT AppInventor platform or use paper modeling, with the aim of presenting the project at the end of the sessions, or in another event created for that purpose.

The projects will cover all the tools and themes explored. Students may choose, within the topic of hygiene habits, which sub-topics they want to explore and focus on. Regarding programming skills, they should follow a simple framework to create their own apps, so that they can be as independent as possible.

Supplementary learning resources and educational activities

1. Regarding the school Research project:
 - Production of multimodal content – Students can transfer and use the knowledge acquired to other classes.
 - Public health reflection.
 - Competition – reward of the best app ideas.

School Research Project (Ciência Viva)

Topics

Importance of digital literacy and real-life implications.

Basic technical features and principles of programming solutions development.

Possible applications of mobile applications in public health (e.g., in the promotion of hygiene habits).

Research management, design and administration

Challenge: Content exposition creation on one topic involving (for example) “hygiene habits”, promoting not only technology, but also public health.

Method (summary): Lessons 5 to 7 will be dedicated to the school research project. Students are, as usual, organized in groups and each group addresses the practiced programming and technological concepts and connects them to health.

Development process:

The project is based on the use of technology to create scientific artefacts. The five-six sessions will be supervised by the teachers and developed by the students, with scheduled moments for checking the work development.

Groups of students will be instructed to create an interactive game or tool app that explores the topic of hygiene habits in some way, as well as some other resources (of their choice) that they see fit on the same topic:

During session 1, students are presented, not only, with software to use for content creation but also the norms to follow:

1. Each group should have, at least:
 - a. 1 sketch of an app (or the app itself) on the theme of healthy habits.
 - b. One other multimodal resource on the theme of healthy habits.
 - i. It can be a mental/conceptual map, a quizz, a presentation, an interactive resource, an infographic, a story, a video, etc.
 - c. A short portfolio with all the created resources.
2. Each group is required to:
 - a. brainstorm a project idea, develop drawing(s) of the app/game on paper, creating a [storyboard](#).
 - b. meet regularly with the teacher to discuss the feasibility of the app and if necessary make any changes to their plan.
 - c. Present a short (5-10 minutes) elevator pitch of your project idea to the class.
3. Create a portfolio (free structure, let the student be creative) write up of your project.
 - a. Some useful points:
 - i. Names of developers
 - ii. Name of your app and why
 - iii. Identification and justification of the other resources and
 - iv. What problem it solves and/or why it is important or useful

- v. People who would use your app (target audience or market) and why they would use it
- vi. Describe what each person on your team did
- vii. Describe how you made decisions together
- viii. Describe how effectively you worked together as a team

Finishing session 1, the groups are created and ideas start being discussed.

During session 2, the teacher will pass by the groups to collect ideas and topics and, if valid, students can start working on their project. The teacher will provide all the needed help, even if that means that he is an contributor to the project.

From session 3-forward, the students will actively work on their project and are encouraged to exchange ideas with other groups.

Teaching-learning process milestones:

1. Students will be able to propose basic programming solutions.
2. Students will be able to communicate the findings, motivations and limitations of various solutions considered in the work process.
3. Students will be able to identify and communicate the importance of digital literacy/end-user development in public health and citizenship.

Teaching-learning process for school project (summary):

1. Development of multimodal materials.
2. Basic Mobile Applications Sketches (or real for the most driven).
3. Presentation of all the resources created in the open schooling event, where students will be advocating better conditions for their community and show their relationship with public health and low-code environments.

Organization of the open schooling event:

1. Each project output (portfolio) is presented by the students in a community setting (e.g., exposition center, municipality, science fair) with appropriate/pre-prepared environment (computer and smartphone with the MIT App Inventor installed).
2. Students do a pitch on how mobile solutions can be used to address public health, like the case of good hygiene habits. Technical speeches to motivate peers to new technologies and technological environments. Students will also be advocating better conditions for their community and show their relationship with citizens health.

3. Students, parents, school community and relevant local stakeholders attend the event and recognize that mobile solutions can be used to address real life challenges, public health ones, and others. They also get high-level understanding on strategies to minimize the phenomena and how they may have an influence on the relevant settings (e.g., home, school, workplace, community).

Data Analysis and Reporting

Content Analysis.

Multimodal resources.

Portfolio development.

Target Audience for Recommendations

School community and local stakeholders: students, parents, municipalities, engineers, public health authorities, and local enterprises.

Public Debate and Recommendations (based on research results)

Presentation of the resources produced by students in a community setting and dissemination of evidence recommendations via social, community and conventional media.

Discussion and feedback.

Attribution of the prize of “best app ideas”.

Main partner responsible: UM (University of Minho)