CODELASTRO. A STEM Project for Code Learning with Astronomical Ideas

H Cachetas[1], VM Martins[2], MFM Costa[3], JP Vieira[4]
[1]AE André Soares in Braga, Portugal
[3]Centro de Física das Universidades do Minho e do Porto, Portugal
[1,2,4]Casa da Ciência de Braga, Portugal
henrique.cachetas@gmail.com

Abstract. Learning how to use and to be able to program computers and other computer based or artificial intelligence devices is becoming of high importance in today's modern societies. Micro:bit is a device that can be used to introduce children and students to this world. Micro:bit is a pocket programmable device that allows children and young people to learn how to program in a simple, playful and creative way. On the other hand, astronomy is an important science topic that appeals to the imagination and wonder of children, therefore contributing to their motivation and in promoting interest in science. Based on these two ideas, a STEM project named "Code Learning with Astronomical Ideas", codename CODELASTRO, was developed and is going to be presented herein. The core project idea was to create a code learning-program based on micro:bit and astronomical topics. The micro:bit integrated sensors allow the development of STEM projects, in which children and young people, almost autonomously and in groups, can learn by going through all phases of design, development, coding, testing and execution. Exploring the relationship of programming and astronomy gave us an opportunity to teach how science can be challenging and attractive to young people. The initial target audience of the CODELASTRO project was students and teachers from developing nations, Mozambique and East Timor in particular. It was proved that this project can be successfully implemented in such countries with limited resources and facing developmental challenges.

Keywords. STEM Projects, Astronomy, Coding.

1. Introduction

The CODELASTRO project is a code learning program based on micro:bit and astronomical topics. The project team, based at Braga Ciência Viva Center's (CCVB) in the north of Portugal, has partnered with schools in Mozambique and Timor-Leste to respond to a funding application of Office of Astronomy for Development (OAD), a joint project of the International Astronomical Union (IAU) and the South African National Research Foundation (NRF) with the support of the Department of Science and Innovation (DSI).[1] These funded projects use astronomy, space/related topics to tackle challenges in their communities and regions in order to promote sustainable development and create a better society. CODELASTRO implementation plan began by creating partnerships with schools that do not have access to this type of technological education, due to lack of means, which permitted a selection of children and young people in order to reduce inequalities. The chosen schools were Portuguese School of Mozambique and Portuguese School of East Timor, two Portuguese speaking countries. This kind of projets are an excellent way of exploring and investigate the metacognitive approach of science, Inquiry Based Science Education (IBSE). This approach gives students a metacognitive understanding of procedures leading to discussion, communication and argumentation among peers. The creation of ideas and solid cognitive structures gives to this project a way of linking science to society. [2] For young students and teachers, graphical coding is far easier and more adequate than dealing directly with coding languages. The use of the Makecode graphical programming tool appeals to the learners with its commands, simple or more advanced, represented by graphical blocks that the user will place on the programming board connected in a proper sequential way. Micro:bits were sent to schools in Mozambique and Timor-Leste (Fig.1). Then the team held online meetings with teachers in the partner schools to determine the structure and contents of a workshop for code learning based on astronomical ideas. This resulted in a learning guide to help teachers and students learn about the project.

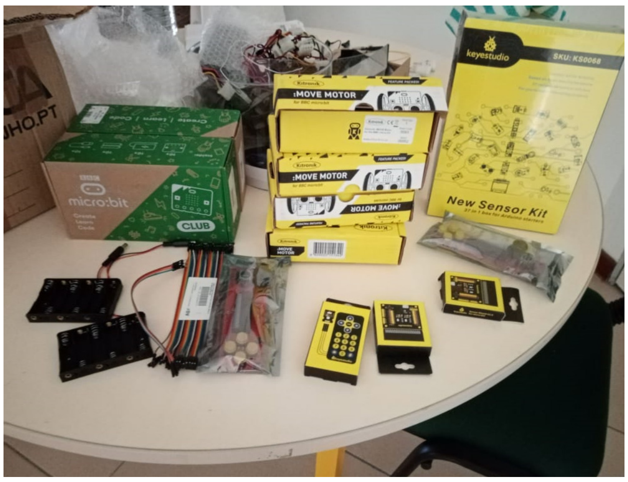


Figure 1. Set of materials sent to East Timor and Mozambique

The replication of the project was made possible through online training of teachers in those schools through 7 webinars with hands-on activities for learning coding with astronomical ideas (Fig.2). Afterwards, those teachers implemented this learning guide with students of their own schools (Fig.3).

Figure 2. School teachers in Mozambique and East Timor (on screen) learning about Micro:bit and Astronomical Ideas

Figure 3. Children learning about Micro:bit and Astronomical Ideas

The resulting guide and training sessions aimed at the following:

A Mission to Mars was decomposed in several stages in order to be replicated in a simplified way through micro:bit coding and its sensors. Project based learning included the following projects:

- Stage 1 – Astronaut training, reflexes training and reaction time;
- Stage 2 – Rocket launch, acceleration measurement;
- Stage 3 – Space trip, cabinet pressure, monitoring temperature and radiation, sound warnings;
- Stage 4 – Landing, distance to ground, contact;
- Stage 5 – Robot in Mars, robot control, temperature in Mars, climate station.

We can synthetize the project results as follows:

- 7 online training sessions which included 12 professors of different disciplines.
- 10 presencial sessions at Braga Ciência Viva Center's in order to develop and test our methodology and create the learning roadmap, which involved 150 students.
- Sending electronic materials necessary for this project in East Timor and Mozambique.
- Elaboration of a guide for code learning with astronomical ideas, which is available online.
- 1 dissimination activity, both face-to-face and online, at our local University for a public of 20 international teachers from Greece, Estonia, Kenya, Malta and Portugal.

2. Educational Robotics

Educational Robotics is included in a comprehensive spectrum of activities of the so-called Physical Computing, defined as the creative development of interactive objects or systems using programmable hardware. [3]

With this type of teaching processes, it has been demonstrated that learning happens more effectively in a context in which the learner is consciously involved in the construction of a real, visible and tangible object. Likewise, physical devices naturally support an exploratory diy approach, in which students learn to build on existing knowledge by following a step-by-step problem-solving pedagogy in a collaborative and active way [4]..

From the student's point of view, physical computing can become more positive than a more traditional approach in front of the screen. Students are naturally interested in building real devices with tangible physical functions and results, stimulating their creativity. It has also been reported that female children and young people, following this teaching method, are more interested in programming than traditionally. [5]

The benefits of educational robotics can be summarized as follows: [6]

- Motivation: the motivation of students increases, including those from adverse social contexts, because the learning experience and the result is tangible and not just virtual. This is especially true when programming tasks result in a practical and meaningful product.
- Tangibility: the physical nature of the devices helps to create connections in a natural way. Identifying and correcting physical system errors helps students to better understand the concepts of programming and the software development process. The fact that the result of the students' work can be seen and tested leads to a concrete understanding of the content transmitted.
- Collaboration: working with devices often leads to the need to work in groups, each student assuming a different role: design of functions, hardware interface, algorithm and user interaction. Students in each group can collaborate (or compete) due to the physical nature of challenges and tasks.
- Creativity: students relate in a practical way to devices and tasks, awakening their creativity and applying it to what they are building, thus promoting interest and autonomy in tasks.

3. About the Micro:bit

Micro:bit [7] was launched as part of the BBC's Make it Digital educational project [8], which Microsoft joined to create the programming environment, Makecode [9], which only requires a browser and internet access. The Micro:bit board is designed to be visually appealing, tangible, economically accessible, easy to use, interactive and extensible. Its integrated sensors allow the development of scientific, technological, engineering, artistic or mathematical (STEAM) projects, in which students can learn and lead all phases of design, development, programming and execution. Micro:bit's electronic expandability also allows the connection of external sensors, as well as connection to other devices and robots. The Micro:bit system includes: buttons A and B are binary sensors that only assume two possible states: pressed and not pressed. Buttons A, B and A+B can be programmed independently; the golden logo is a touch sensor that acts as the touch screen of a mobile phone, measuring small changes in electricity. It can be used to launch different actions such as when touching the sensor or when we take our finger off it; the accelerometer sensor values acceleration/inclination on the three axes x, y, and z (Fig.4), returning values between 0 and 1023 for accelerations between -2g and +2g. The x-axis values the lateral slope (left-right), the y-axis values the longitudinal slope (front-back), and the z-axis sees horizontal rotating movements. With these sensors (Fig.5) it is possible to detect movements such as shaking, shocks, measuring steps, etc; the magnetometer sensor measures the strength of the magnetic field on each of the three axes. It is useful for serving as a compass, metal detector or magnetic field generated by a magnet; the temperature sensor is that of the ARM processor itself, so the returned value is slightly above room temperature. For this reason, it is necessary to introduce a temperature correction in the code through a simple mathematical operation; the micro:bit light sensor is embedded in the LEDs themselves, which allows we to measure the light intensity in the surrounding environment, returning values between 0 (total darkness) and 255 (direct sunlight). The detectable light spectrum is wider than visible light, so it can also detect infrared light, for example from a television controller. By coupling the micro:bit to a robot, we can use this sensor to create a light tracking function. Projects can be made that take advantage of the fact that light is only a small part of the light spectrum, creating analogies with other radiation; In addition to being possible a sound output through the pins, the micro:bit has incorporated a magnetic PCB speaker to make itself be heard. Likewise, the MEMs microphone provides a sound input, with an LED in front of the micro:bit to indicate when the microphone is turned on.

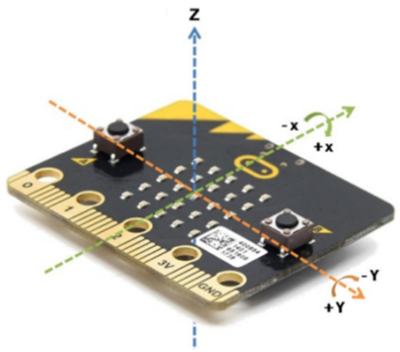


Figure 4. The axis of micro:bit gyroscope

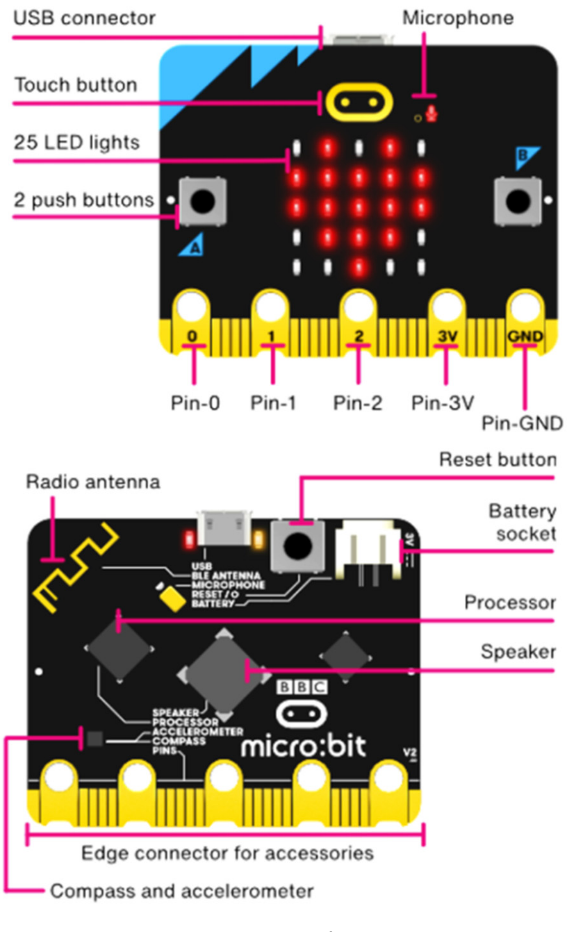


Figure 5. Features of the micro:bit

4. Learning roadmap

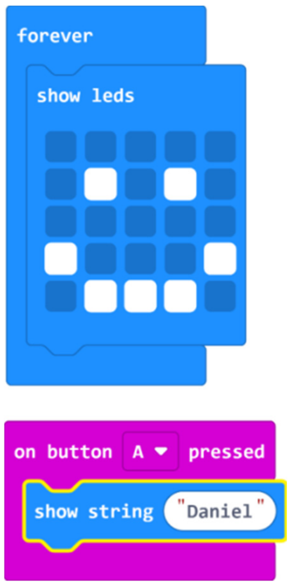
In this project, the most used pedagogical method to teach programming and micro:bit robotics with children and young people, as well as with teachers, was project-based learning. Instead of long theoretical exposures about the board and its programming resources, all explanations are transmitted throughout the implementation of practical projects, in which students are oriented to a concrete and attainable goal, applying the necessary tools. With the development of each project, doubts arise about how to solve the problems presented, before which space should be given to collaborative discussion and the creativity of each one to solve them. Achieving concrete and visible objectives quickly and easily, at least in the first projects, greatly increases enthusiasm and concentration on subsequent projects. An interesting strategy was to carry out a project that includes several subprojects. The project titled "Mission to Mars" includes smaller subprojects: launch (countdown and accelerometer), radiation exposure (luminosity sensor), communication (bluetooth messaging), and temperature difference between Earth and Mars (temperature sensor).

4.1. Stage 1 - Astronaut Training

Astronauts live in extreme conditions and their bodies experience many effects that can potentially diminish their normal abilities, so their training should include exposure to situations similar to those they will encounter in space. On a manned mission to Mars, astronauts should have diet changes, movement restrictions, changes in their sleep patterns, breathe artificial atmosphere and be subject to extreme pressure changes.

4.1.1. Electronic "badge"

With this project students can take a first contact with the micro:bit board and with the makecode programming environment. In order to create a badge through which they can show their names, various functions of the micro:bit can be explored, as is the case of the matrix of LEDs, the presentation of texts, icons or the use of buttons. This code (Fig.6) includes the function of "show LED´s", which can be activated one by one according to the desired drawing. This function has been added within the "forever" command, which indicates that the code will run indefinitely until other commands come to intervene. With the input function "On button A pressed", the student's name will be shown, after which the drawing will appear again on the LEDs.

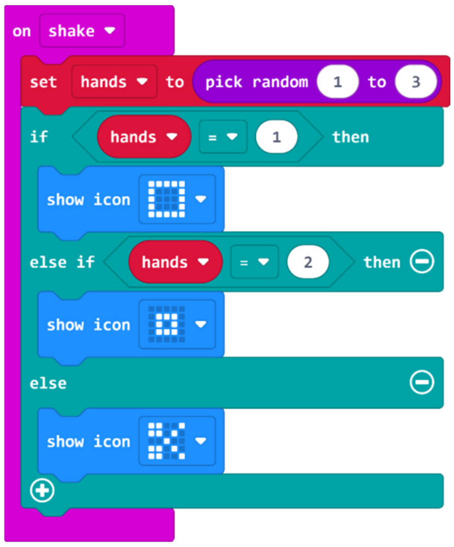


```
forever
    show leds

on button A pressed
    show string "Daniel"
```

Figure 6. Code for LEDs and butons

4.1.2. Rock Paper Scissors

Here we introduce a playful aspect by creating a game well known to everyone that is the Rock Paper Scissors. This game can serve, for example, to decide which "astronaut" will go into space. We use a new input method, the "shake" function, which will trigger the game.



```
on shake
    set hands to pick random 1 to 3
    if hands = 1 then
        show icon
    else if hands = 2 then
        show icon
    else
        show icon
```

Figure 7. Code for “Rock Paper Scisors”

For this it is necessary to define a random variable between 1 and 3, here called "hand", followed by a conditional function. If the hand is equal to 1, the rock-shaped LED´s are shown. If 2, the paper form. And if 3, the form of scissors. In this way it is possible to explain what variables are (very important aspect in programming), random functions and conditional functions. It can also be explained how the "shake" function works, which uses the integrated accelerometer sensor (Fig.7).

4.1.3. Trainning reflexes

Having quick reflexes is vital for astronauts as they can deal with unpredictable incidents or projectiles at high speed. Astronaut reaction times can be improved through training, or they can get worse in case of distraction or by the effect of drugs or alcohol.

4.1.4. Catching the Ball

Continuing with a playful design, this is a simple code that uses basic game functions included in makecode to simulate the training of reflexes. The game consists of a ball that must be caught when it is in the center. An LED moves horizontally back and forth and the A button should be pressed when the LED is in the center of the matrix, adding up a point. Otherwise the game is over and the final score is shown.

With this simple game it is possible to resume the theme of variables and conditional functions (Fig.8).

In addition to these, the functions of games such as sprite (red dot), "move by", "if on the edge, jump", as well as scoring and "game over", are also used. Changing the pause number changes the sprite speed and the degree of difficulty according to the astronaut's reflexes.

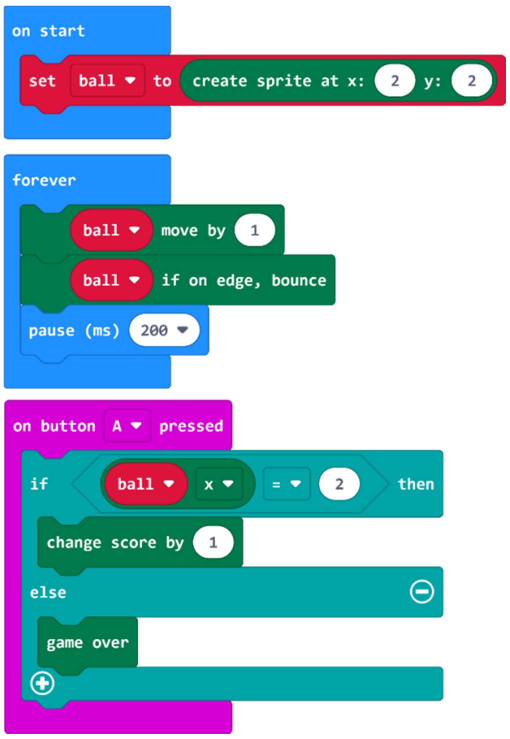
4.1.5. Measure reaction time

By elaborating a slightly more complete project, we can measure the reaction time by taking a series of measurements and finding their average value. The effects of distraction on reaction times can also be explored.

Although not mandatory, we can use an outside LCD module to show the reaction time values. These values could, alternatively, be presented in the micro:bit LED matrix itself.

The LCD module must connect to a shield card or expansion card as shown in Fig. 9. In the case of the LCD, the red wire must be connected to the 5V connection of the shield card. Once the connections are made, a Makecode extension is added to control the

LCD, searching in the field of extensions for "LCD" or "LCD1602".



```
on start
    set ball ▾ to create sprite at x: 2 y: 2

forever
    ball ▾ move by 1
    ball ▾ if on edge, bounce
    pause (ms) 200 ▾

on button A ▾ pressed
    if ball ▾ x ▾ = ▾ 2 then
        change score by 1
    else
        game over
```

Figure 8. Code for “Catching the ball”

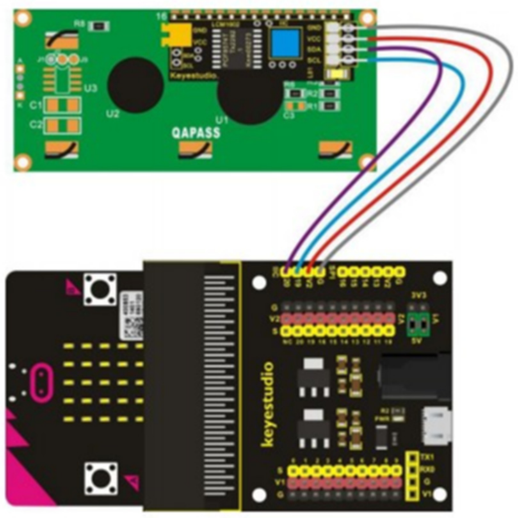
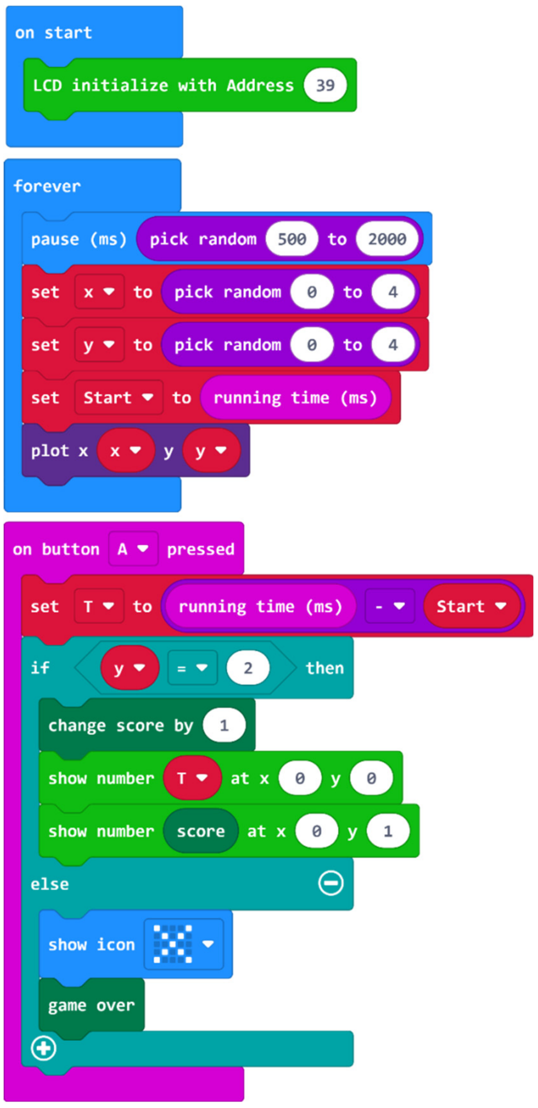


Figure 9. LCD connectiob to shield

A simple way to program the LCD in Makecode to measure reaction time is shown Fig.10.

To frame the use of this code, we can remember that reflexes are rapid responses of human beings to the environment. These reflexes involve the nerves that connect the spinal cord to the muscles, which give rise to the response. Some reflexes involve decision-making by the brain, which implies a reduction in response time. For this reason, reaction times can be improved with training, or else be impaired by distractions or mental changes caused by alcohol, for example.



```
on start
    LCD initialize with Address 39

forever
    pause (ms) pick random 500 to 2000
    set x ▾ to pick random 0 to 4
    set y ▾ to pick random 0 to 4
    set Start ▾ to running time (ms)
    plot x x ▾ y y ▾

on button A ▾ pressed
    set T ▾ to running time (ms) - ▾ Start ▾
    if y ▾ = ▾ 2 then
        change score by 1
        show number T ▾ at x 0 y 0
        show number score at x 0 y 1
    else
        show icon ▾
        game over
```

Figure 10. Code for “reaction time”

4.1.6. Step counting

Astronaut preparation requires numerous exercises to check their physical ability and train mobility in the harsh environment of space. We will train the astronaut to slow down his steps, which could jeopardize the success of his mission. This project aims to convey that the micro:bit can collect numerical data from acceleration, and apply a limit to sensor data to enable events when the number of steps is increased (Fig.11). We could resort to the "On shake" function that recognizes the typical movement of the human being when walking. However, this function may not return the most correct data, since it is possible that not all steps are recognized. Thus, we used the numerical data of the accelerometer, accepting as steps all accelerations with a value greater than 1500. We may need to change this number to make the step count more accurate. This modification on the activation threshold is called "calibration".

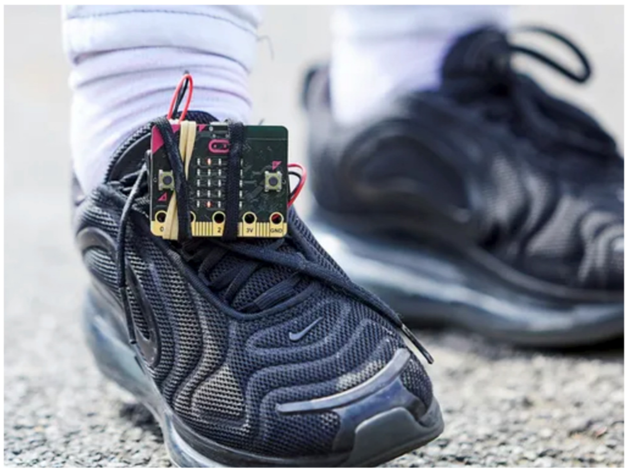
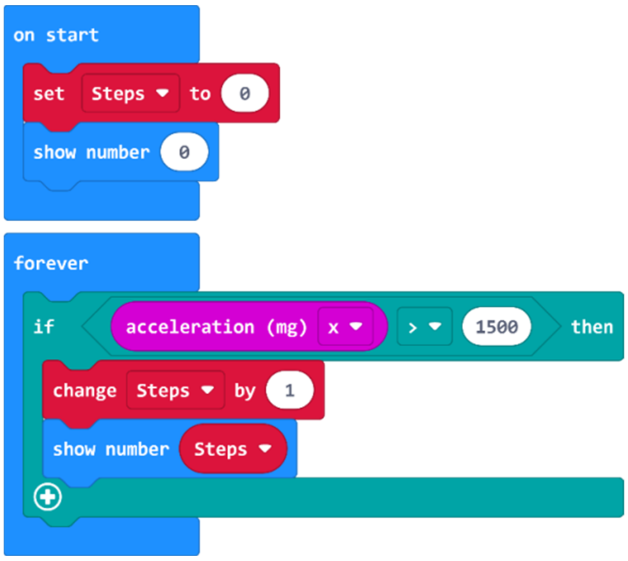


Figure 11. Micro:bit attached to shoe [10]

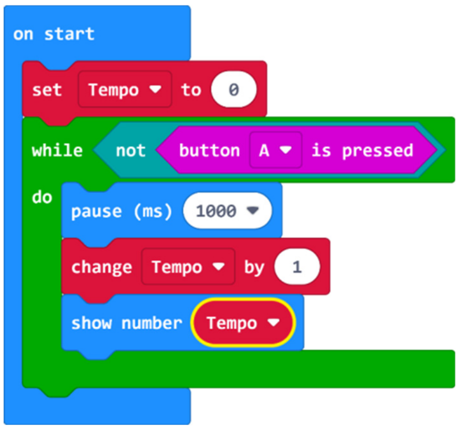
It should be noted that when the micro:bit is not moving, it gives a reading of about 1000, which is caused by the earth's gravity exerting this force on the micro:bit. By placing the micro:bit into the shoes, astronauts can train walking very slowly, placing a limit on how many steps they can take, non-stop, in 20 seconds, for example (Fig.12). To this end, a second micro:bit may be programmed to count the seconds (Fig.13).



```
on start
    set Steps ▾ to 0
    show number 0

forever
    if acceleration (mg) x ▾ > ▾ 1500 then
        change Steps ▾ by 1
        show number Steps ▾
    ⊕
```

Figure 12. Code for step counting

As a design improvement, in order to save the micro:bit battery, we can prevent the LEDs from being permanently connected to show the steps already taken. Thus, we can program so that only when the A button is pressed appear the steps already taken. Alternatively to walking slowly, it can also be recorded the number of quick steps each astronaut can do in the same amount of time.



```
on start
    set Tempo ▾ to 0
    while not button A ▾ is pressed
    do
        pause (ms) 1000 ▾
        change Tempo ▾ by 1
        show number Tempo ▾
```

Figure 13. Code for timer

4.2. Stage 2 - Rocket launch

The launch phase requires extreme care and several series of tests to verify that everything is in a condition for the rocket to take off. The countdown, i.e. the decrescente count for NASA launches, typically begins two days before launch. It is necessary to carry out electronic checks on the systems, verify that there are no leaks in the fuel tanks, and ensure that the mechanical systems are working properly. [11]

4.2.1. Gas and pressure detection

We can simulate a gas detector to check for leaks using a smoke sensor or gas sensor. Similarly, we can use a pressure sensor to simulate checking the pressure inside the fuel tanks, or inside the chambers where the astronauts are. The various fuel tanks can contain liquid oxygen, liquid hydrogen and kerosene, among others, which once mixed or activated generate the combustion that will boost the rocket. A small leak can compromise the entire mission, so it's essential to detect them before launch. These gas leaks can also occur in our daily life as well as inside the astronaut cabin. If a toxic or flammable gas is released, a major risk to the health of the people occurs.

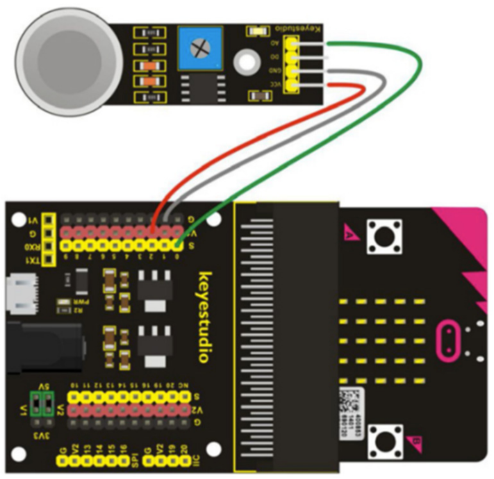


Figure 14. Analog gas sensor connection scheme

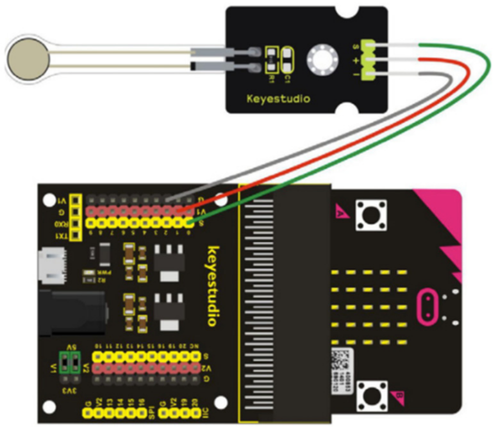


Figure 15. Analog pressure sensor connection scheme.

For this project we will use the analog gas and pressure sensors, whose connection to the micro:bit is carried out through the shield plate (Fig.14 and Fig.15). We display the connections of each sensor separately, although they can also be connected together, taking into account the programming for analog reading on the respective connection pin to each of the sensors. In order to be able to observe the evolution of sensor readings on the computer, the micro:bit with the makecode must be paired. After transferring the code to the micro:bit, the console can be opened to see the collected data (Fig.16). A file with data can also be downloaded to be analyzed later, or added to excel to make graphics.

If, in addition, we want to add a warning sound indicating the presence of flammable gases, we may use a passive buzzer module by programming as shown in Fig. 17. The code is programmed for the gas sensor to be read on pin 1 and the pressure on pin 2. Whenever the sensor signals the presence of gas (values less than 1000, in this case), a sound will be emitted through the buzzer, connected on pin 0.

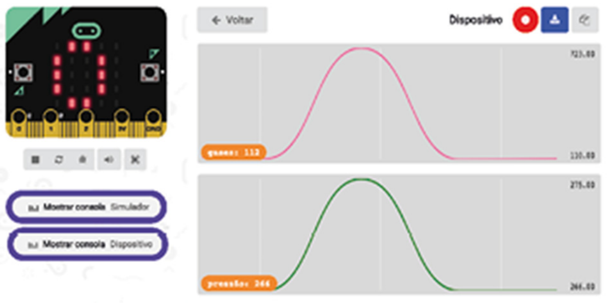
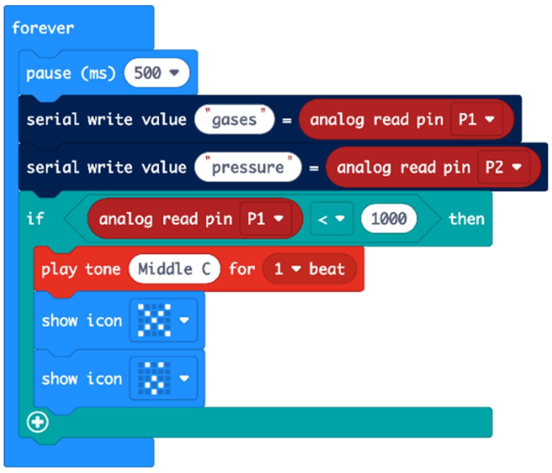


Figure 16. Reading of sensors data



```
forever
  pause (ms) 500
  serial write value "gases" = analog read pin P1
  serial write value "pressure" = analog read pin P2
  if analog read pin P1 < 1000 then
    play tone Middle C for 1 beat
    show icon
    show icon
```

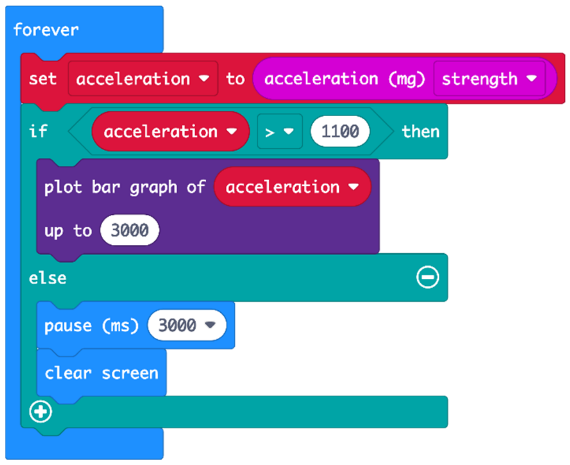
Figure 17. A code to read touch sensors

4.2.2. Acceleration

Understanding the difference between speed (measured in meters per second) and acceleration (measured in meters per second squared) is one of the essential learnings in physics curriculum. However, there is a common confusion between these two concepts, as to the terminology or units used.

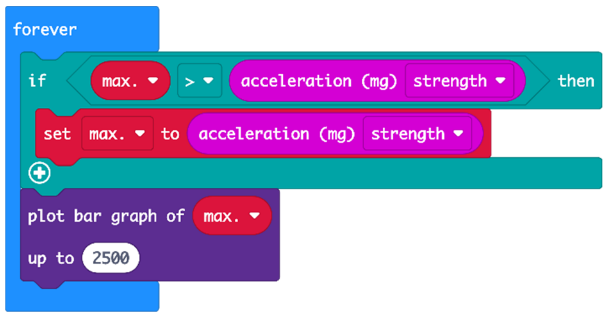
One of the concepts to have in mind is that rockets require a large amount of fuel to reach the escape velocity of the Earth's gravitational field and be launched into space. However, they do not require fuel when they are traveling at a constant speed, as there are no forces of resistance to movement in space. In order to understand the difference between speed and acceleration, we can program the microbit to turn on the LEDs when it is accelerating, keeping them off when the speed is constant. With this code (Fig.18), students can be asked to move the micro:bit without the LEDs on, since for this they can not apply too much force. If the LEDs light up, it means that the acceleration has passed the established limit.

We can also draw a code to show a graph of the acceleration obtained (during rocket launch), recording the maximum of the accelerometer sensor reading (Fig.19).



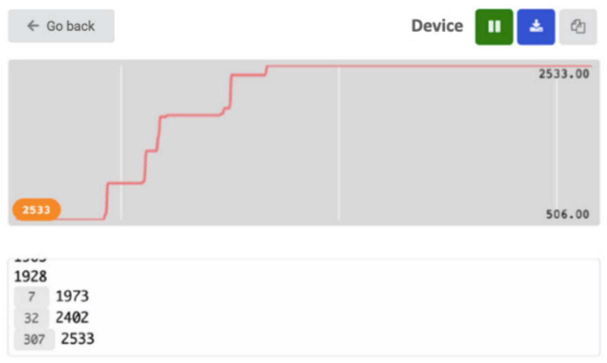
```
forever
    set acceleration ▾ to acceleration (mg) strength ▾
    if acceleration ▾ > ▾ 1100 then
        plot bar graph of acceleration ▾
        up to 3000
    else
        pause (ms) 3000 ▾
        clear screen
```

Figure 18. Code to light the LEDs when the acceleration is greater than the 1100 limit, turning off when it is lower



```
forever
    if max. ▾ > ▾ acceleration (mg) strength ▾ then
        set max. ▾ to acceleration (mg) strength ▾
    plot bar graph of max. ▾
    up to 2500
```

Figure 19. Code to show the graphics with the maximum acceleration value obtained by the micro: bit



```
1928
  7   1973
 32   2402
307   2533
```

Figure 20. Console with graphics of real-time micro:bit accelerometer sensor values

In order to follow the concrete values of the sensor reading, we can open the console in Makecode to view, record or download the measured acceleration values (Fig.20).

If students move the micro:bit too fast, the LEDs will light up, but they won't be able to keep it that way for long. It is interesting to note that for LEDs to stay connected, a considerable amount of constant force is required. This may be related to the force required to launch a rocket into space, which must be maintained for as long as necessary until it reaches the required altitude and speed.

4.3. Stage 3 – Travel

During the trip some permanent care should be maintained for the safety of the rocket and astronauts. We will use three sensors to analyze important factors: radiation exposure, pressure and temperature inside the astronaut cabin.

4.3.1. Conditions in space

Even at relatively low altitudes on the Earth's atmosphere, conditions are hostile to the human body. The altitude at which atmospheric pressure equals the vapor pressure of water at the temperature of the human body is called the Armstrong Line. It is located at an altitude of about 19.14 km. Above this line, fluids in the human body, especially from the lungs, can begin to boil (evaporate). Therefore, above this altitude human survival requires a space suit or a pressurized capsule. There is also a thermodynamic relationship between pressure and temperature. Since temperature is the result of particle movement, the heat or cold senses will be the result of the interaction of these particles with our skin. When the pressure increases, it also increases the number of particles by volume quantity, and therefore increases the interaction between the particles. For this reason, the higher the air pressure around us, the higher the temperature. In space, however, there is no air, that is, there are no moving particles and therefore there is no temperature. We can not say that the temperature of the space is high or low, it simply does not exist. When exposed to the vacuum of space, the astronaut will not gain or lose temperature through convection or conduction (only existing in the presence of air), but will be exposed to radiation temperature transmission. Direct exposure to sunlight, without the protection of the atmosphere or an appropriate suit can cause serious damage. Exposure to high-energy radiation and ionizing cosmic rays present in space can also have serious consequences on astronauts' health. On a mission to Mars, whose round trip can last three years, a large fraction of the astronaut's body cells can be traversed and potentially damaged by high-energy nuclei. The energy of such particles can be significantly decreased or blocked by a shield on the walls of the spacecraft.

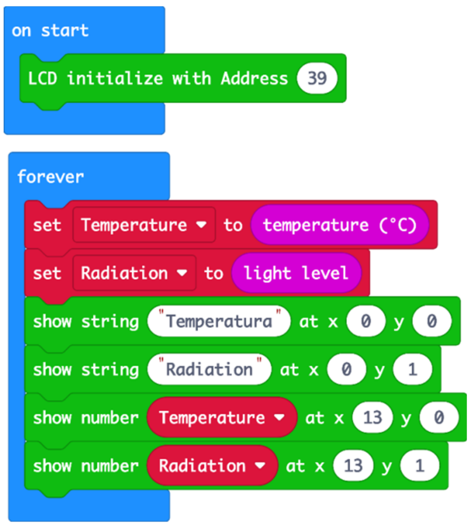
4.3.2. Temperature

The micro:bit board has its own temperature sensor, associated with the processor temperature. However, for the same reason, the temperature obtained is not exact, since the temperature of the processor itself varies the reading, diverting it from the surrounding temperature value. To obtain a value closer to reality, we must calibrate the reading subtracting the difference, comparing with the temperature measured by a more rigorous thermometer. Despite this, the difference is only about 3 degrees, so it can be overlooked without affecting the purpose of the project.

4.3.3. Radiation

Likewise, the board also has a light sensor, which is measured through the LEDs themselves. By making the relation of visible light with electromagnetic radiation, this project can be explored to explain what radiation is, the existence of a visible and invisible spectrum, as well as the extreme radiations that can exist in space without the protection of the atmosphere (Fig.21). With this code, the temperature and brightness of the environment will be shown in real time.

Changes in this environment can be explored by increasing the temperature (approaching a heat source, or exposing it to the sun, for example), as well as changing the amount of light (representing radiation) in the environment. These changes can be logged over time and used to perform charts and tables with the recorded values. The connection to the LCD is performed in the same way as explained earlier (section 4.1.5.).



```
on start
  LCD initialize with Address 39

forever
  set Temperature to temperature (°C)
  set Radiation to light level
  show string "Temperatura" at x 0 y 0
  show string "Radiation" at x 0 y 1
  show number Temperature at x 13 y 0
  show number Radiation at x 13 y 1
```

Figure 21. Code to show temperature and brightness readings on the external LCD

4.3.4. Cabin pressure

Pressure can only be measured through external sensors, since the micro:bit does not contain this built-in sensor (Fig.22).

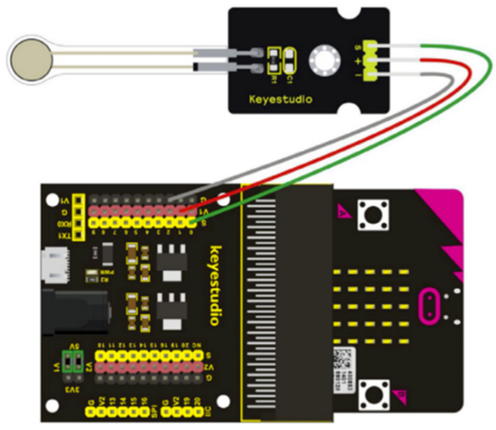
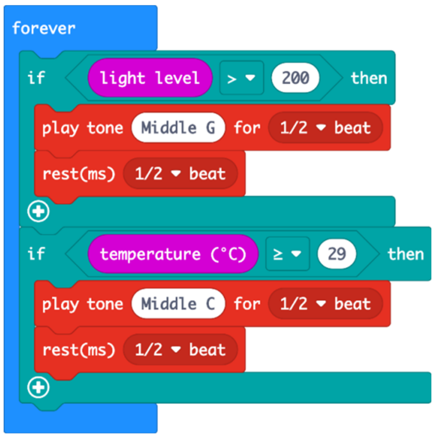


Figure 22. Pressure sensor connection scheme



```
forever
  if light level > 200 then
    play tone Middle G for 1/2 beat
    rest(ms) 1/2 beat
  if temperature (°C) ≥ 29 then
    play tone Middle C for 1/2 beat
    rest(ms) 1/2 beat
```

Figure 23. Code to beep when temperature and radiation limits are exceeded

If we want to connect multiple sensors at the same time, the connections must be made using different pins for each connection, with the consequent change in the makecode codes. The result of this programming can be tested by placing the sensor inside a balloon, which can be pressed to change the air pressure inside.

4.3.5. Audible warnings

To complete previous projects, we can set an audible warning whenever a temperature or radiation value is found above a previously established limit. With this code (Fig.23), the micro:bit (V2) will emit a warning sound whenever the radiation passes 200 and the temperature of 29ºC. In addition to an audible warning, an action could also be triggered through an external actuator. For example, an engine could be activated that would simulate the closing of an isolation gate of a section of the ship.

4.4. Stage 4 – Landing

The atmosphere entry, descent and landing is the shortest and most intense phase of a mission to Mars. On the Perseverance mission [12], the journey from the top of the atmosphere to the planet's surface lasted seven minutes, during which hundreds of critical events must be executed perfectly for a safe landing. To replicate the descent and landing, we can use several external sensors, including the distance and contact sensors.

4.4.1. Distance

As the rover descends through mars' atmosphere, it is necessary to know how far it is from the ground. The ultrasonic sensor can simulate the calculation of this distance, through a method in everything similar to that of a sonar. It includes an ultrasonic transmitter, a receiver and a control circuit. The module will emit ultrasound waves that will be reflected when finding an object, sended back to the receiver as an echo signal. By calculating the signal return time, we can determine the distance to the object.

In order to use this module, it is necessary to install an extension of the makecode, which can be found by searching "sonar" in the extensions. This link uses pin 1 for trigger and pin 2 for echo, but other pins could be used by making their code change. Using the "Sonar" (Fig.24) option of the added extension, we can configure the sensor to return the distance to the object as shown in Fig. 25.

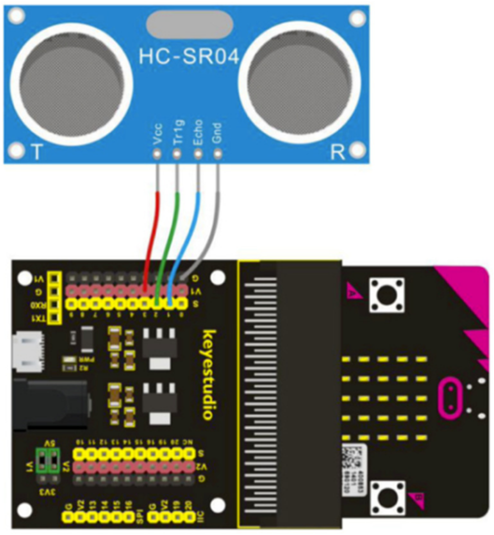
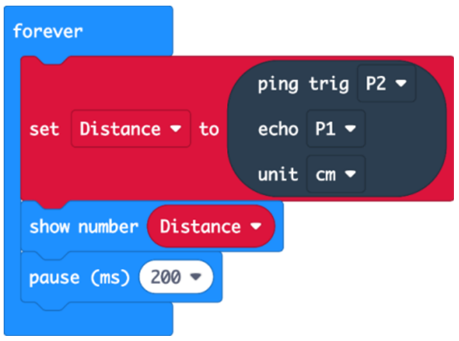


Figure 24. Ultrasonic sensor connection scheme



```
forever
    set Distance to ping trig P2
                         echo P1
                         unit cm
    show number Distance
    pause (ms) 200
```

Figure 25. Code to get distance to ultrasonic sensor

This code will cause the distance to appear in centimeters on the micro:bit LEDs, but could also be programmed to show the distance value on an external screen, as previously programmed. This sensor could also be programmed together with a beep.

4.4.2. Contact

Contact with the ground on another planet is the most exciting moment of a space trip, as a good landing means the arrival at the ultimate goal of the journey. Contact is, after all, a controlled collision at a low speed, in order to avoid damaging the rover. That is why the sensor we're going to use is the collision sensor. The micro:bit, through its internal accelerometer sensor, has the possibility to measure the force of a collision, however its sensitivity is too imprecise for it to be used reliably. The collision sensor, on the other hand, only allows we to understand whether or not there was contact, not measuring the force of that impact. This sensor, also known as an electronic switch, is a button-like module, which gives rise to an on-off digital signal. We will program this sensor in conjunction with an external buzzer so as to produce a contact signal as soon as the collision sensor is activated. We must note that in the connection diagram (Fig.26), the collision sensor is connected to pin 0, while the buzzer is activated via pin 7 (Fig.27). The collision sensor gives the value of 0 each time it is pressed, and 1 when it is not. We can also notice that the sensor has an LED that lights up whenever there is a collision.

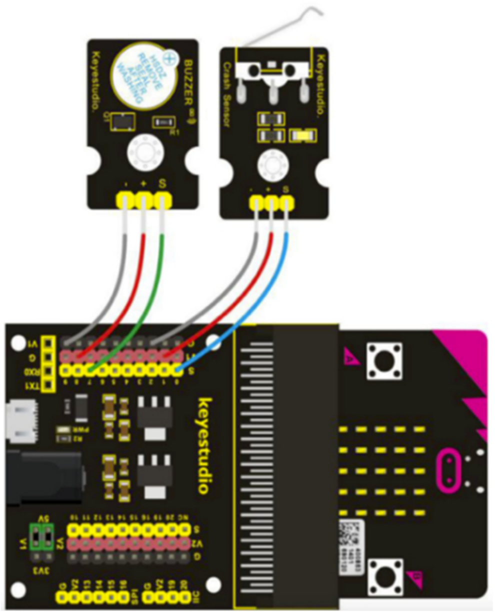
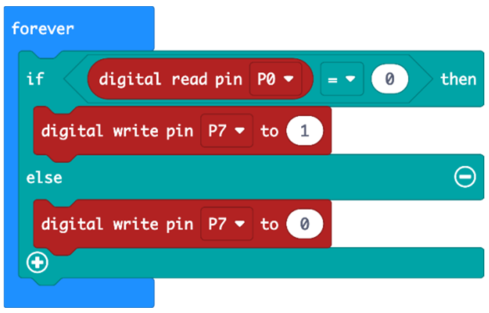


Figure 26. Conection schemes for buzzer and electronic switch



```
forever
  if  digital read pin P0 = 0  then
    digital write pin P7 to 1
  else
    digital write pin P7 to 0
```

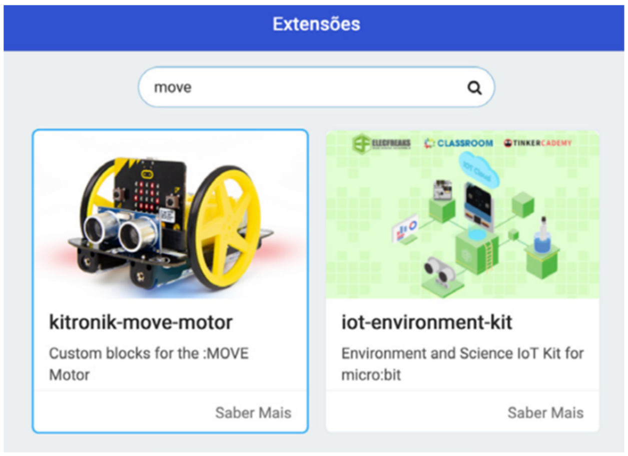
Figure 27. Code to get a beep when there is a collision

4.5. Stage 5 – Robot on Mars

After landing on Mars, the last phase of the mission begins, with a robot having to move and collect data from the Martian environment. Mars is being explored for eight missions currently: six in orbit — Mars Odyssey, Mars Express, Mars Reconnaissance Orbiter, Mars Atmosphere and Volatile Evolution Missile - MAVEN, Mars Orbiter Mission and ExoMars Trace Gas Orbiter — and three on the surface — Curiosity, Perseverance and the Chinese rover Zhurong. Among the deactivated missions that are on the Martian surface are the Spirit probe and several other probes and rovers, such as the Phoenix, which completed its mission in 2008, and Opportunity. Projects for simulating a rover on Mars may inspire students to research some of the missions mentioned above. This study may also include knowledge of the planet's physical characteristics, such as its atmosphere and climate, surface geology, soil and hydrology, craters, and volcanic activity.

4.5.1. Robot Control

We use Kitronik's "MOVE" robot [13] to replicate the rover on Mars. To do this, it is necessary to install the corresponding extension, searching for the word "move" (Fig.28). After installing the extension, new options will be available to control the lights, motors, sensors and actuators of the robot (Fig.29). In order to learn a basic method of controlling the engines, we will program a simple effect in which the robot moves in various directions.



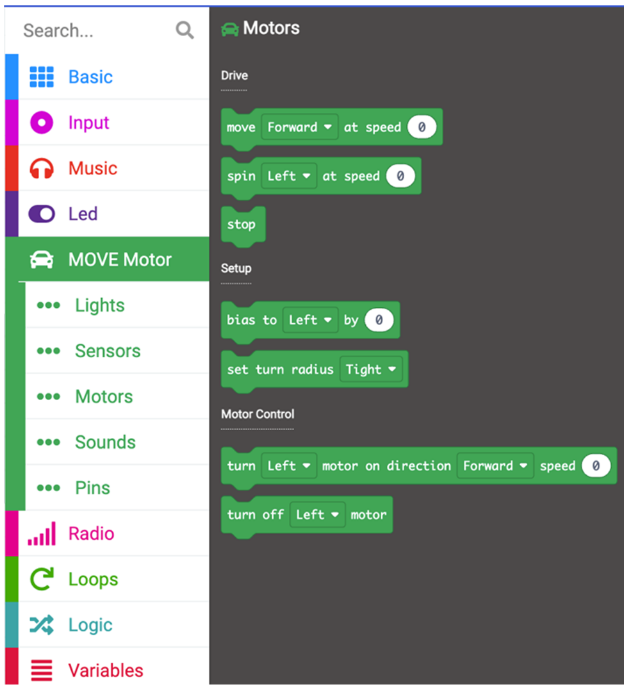
Extensões

move

kitronik-move-motor
Custom blocks for the :MOVE Motor
Saber Mais

iot-environment-kit
Environment and Science IoT Kit for micro:bit
Saber Mais

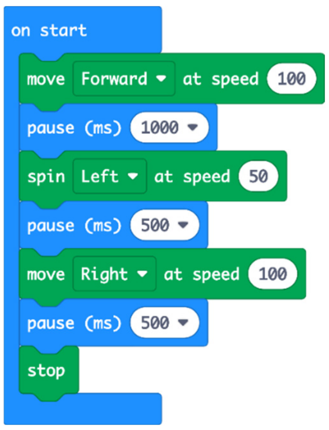
Figure 28. Adding :Move Motor extension



```
move Forward ▾ at speed 0
spin Left ▾ at speed 0
stop
bias to Left ▾ by 0
set turn radius Tight ▾
turn Left ▾ motor on direction Forward ▾ speed 0
turn off Left ▾ motor
```

Figure 29. Move Motor functions

Engines can be programmed synchronized or with differentiated speeds. When one wheel is programmed to rotate at a slower speed than the other, the robot will move in the direction of the slower wheel. The pause between the engine instructions corresponds to the time the wheels will operate. Using a code with this type of commands, a route can be programmed between two points, bypassing obstacles, simulating the trajectory that a rover should make on Mars (Fig.30).

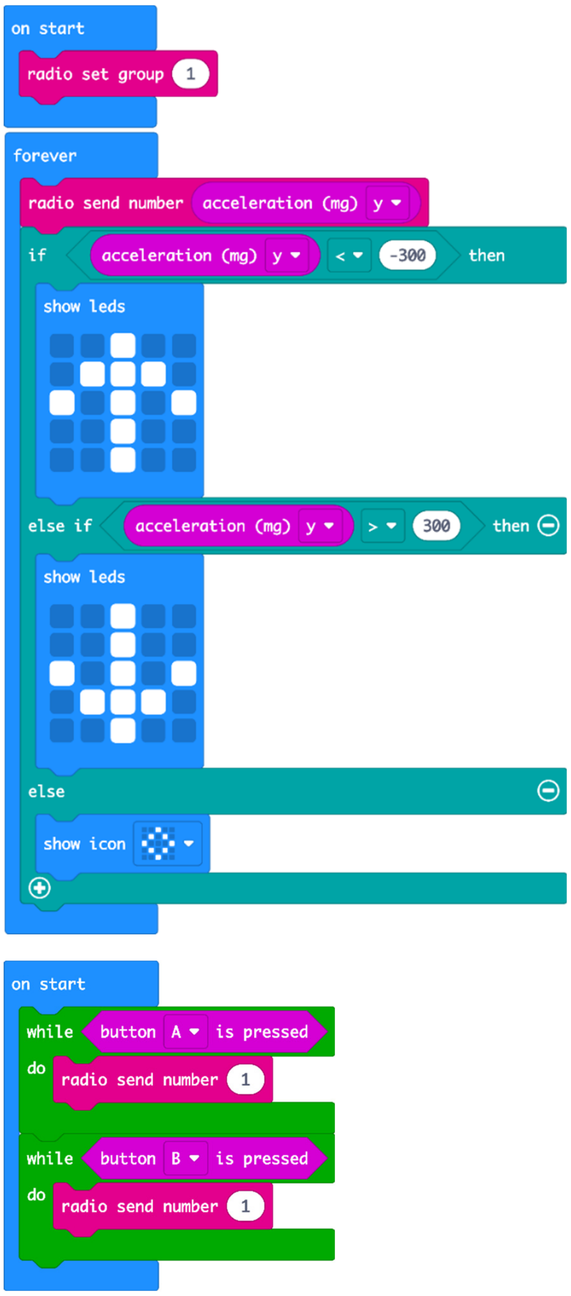


```
on start
    move Forward ▾ at speed 100
    pause (ms) 1000 ▾
    spin Left ▾ at speed 50
    pause (ms) 500 ▾
    move Right ▾ at speed 100
    pause (ms) 500 ▾
    stop
```

Figure 30. Code example for :Move Motor

4.5.2. Remote control

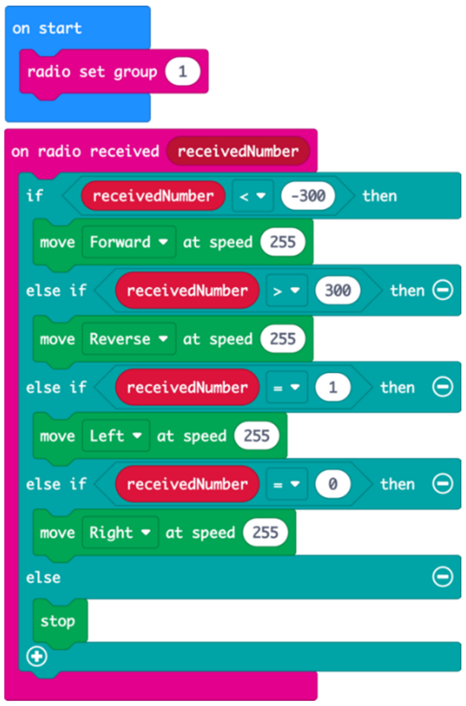
The robot, in addition to being programmed for predefined routes, can also be controlled remotely. However, it must be remembered that all communication between Earth and Mars must take into account the distance between the two planets.



```
on start
    radio set group 1

forever
    radio send number acceleration (mg) y ▾
    if acceleration (mg) y ▾ < ▾ -300 then
        show leds
    else if acceleration (mg) y ▾ > ▾ 300 then ⊖
        show leds
    else ⊖
        show icon ▾
    ⊕

on start
    while button A ▾ is pressed
    do radio send number 1
    while button B ▾ is pressed
    do radio send number 1
```

Figure 31. Command code for controlling the robot remotely

Depending on the relative position in their orbits, the signal sent from Earth may take 15 minutes to reach the red planet. In any case, for learning purposes, we can develop a distance control code. In the case of radio communication, after defining the radio group (the frequency), we will use the accelerometer sensor to control the robot by tilting the micro:bit (Fig.31). The acceleration on the Y axis is determined by the back and forth tilt of the micro:bit, which is why we use this movement to move the robot in these directions. We want the robot to only move with a tilt value greater than 300, so that the controller is not too sensitive. For left and right shift, we use buttons A and B, sending a numeric signal for each direction. The micro:bit that will be coupled to the robot will make the movements depending on the data received. With a received number greater than 300 or lower than -300, the robot will walk back and forth respectively (negative Y values correspond to a forward slope). With the number 1 (sent by pressing button A) the robot moves to the left, and with the number 2 (pressing button B) to the right. With all the other numbers, the robot will stand still (Fig.32).



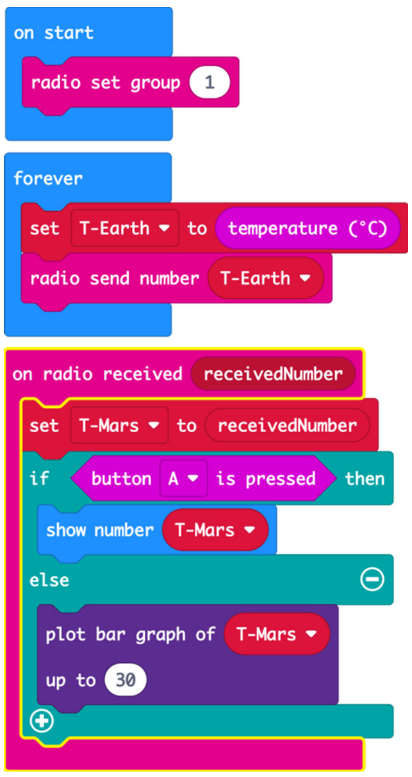
```
on start
    radio set group 1

on radio received receivedNumber
    if receivedNumber < -300 then
        move Forward at speed 255
    else if receivedNumber > 300 then
        move Reverse at speed 255
    else if receivedNumber = 1 then
        move Left at speed 255
    else if receivedNumber = 0 then
        move Right at speed 255
    else
        stop
```

Figure 32. Code for Micro:bit attached to robot

4.5.3. Temperature on Mars

In order to simulate a temperature sensor installed on Mars that sends its value to Earth and vice versa, we should note that the average temperature on Mars is about -60ºC, while the average temperature of the Earth is about 15ºC.

This means that the temperature difference between the two planets is 75ºC. If we want the micro:bit corresponding to Earth to show the equivalent temperature of Mars, we can subtract 75 at the temperature received. For the use of the radio, it is necessary to define an equal radio group for the transmitter and receiver (groups can be defined between the values 0 and 255). The micro:bit temperature allocated on Earth is stored in a T-Earth variable, the value of which is sent by radio. When this micro:bit receives a numeric value (sent by the micro:bit allocated on Mars), that value is saved in the T-Mars variable. This value is continuously shown on a temperature chart through the LEDs. When the A button is pressed, the LEDs show the temperature value of Mars. The other micro:bit (equivalent to that on Mars) will have to have the variables reversed, as it will receive by radio the temperature of the Earth. This same code (Fig.33) can be used to continuously send the value of other variables, such as the inclination and orientation of the accelerometer and compass (simulating the position and direction of the rover), or luminosity (analogous to radiation).



```
on start
    radio set group 1

forever
    set T-Earth to temperature (°C)
    radio send number T-Earth

on radio received receivedNumber
    set T-Mars to receivedNumber
    if button A is pressed then
        show number T-Mars
    else
        plot bar graph of T-Mars
        up to 30
```

Figure 33. Code sending Earth's and receiving Mars Temperature

4.5.4. Clima station

For the installation of a climacteric station for continuous data collection, we will use a Kitronic real time clock (RTC) & Klimate card (Fig.34). The environmental sensor of this plate allows measuring temperature, barometric pressure and humidity, associating this data with the current date and time. This card already has a 3V battery holder to power a clock in real time while the micro:bit is not connected. The RTC card and micro:bit can be powered at the same time via a USB connection or through the terminal block.

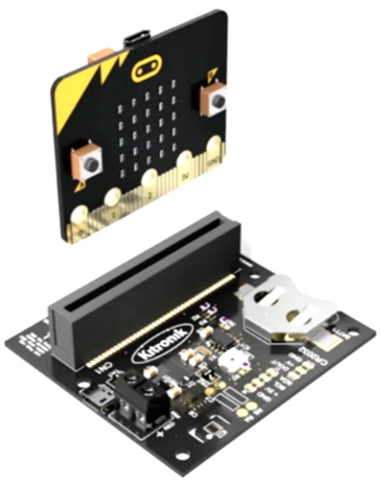
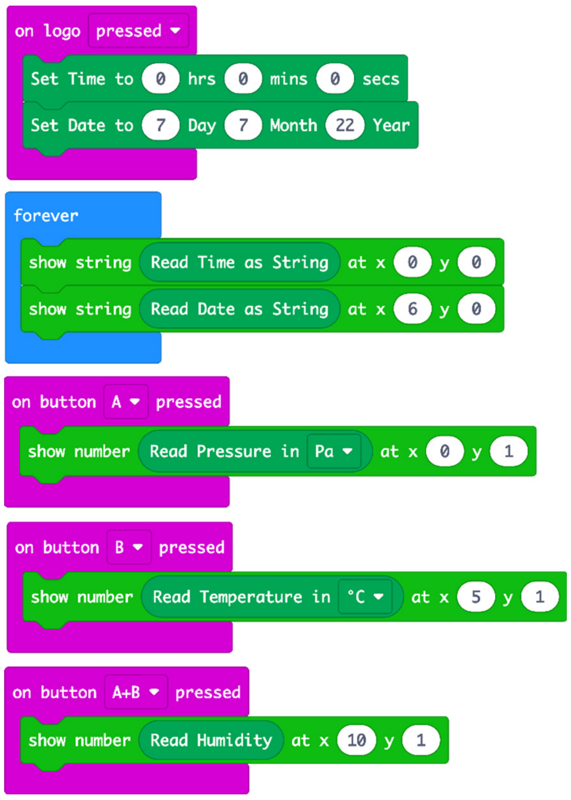


Figure 34. RTC board



```
on logo pressed
    Set Time to 0 hrs 0 mins 0 secs
    Set Date to 7 Day 7 Month 22 Year

forever
    show string Read Time as String at x 0 y 0
    show string Read Date as String at x 6 y 0

on button A pressed
    show number Read Pressure in Pa at x 0 y 1

on button B pressed
    show number Read Temperature in °C at x 5 y 1

on button A+B pressed
    show number Read Humidity at x 10 y 1
```

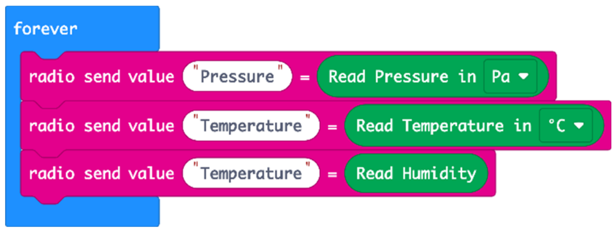
Figure 36. Code to show time, date, pressure, temperature and humidity

In order to define the beginning of time, we can create a code so that the time starts from scratch when we press the micro:bit Logo.

With this code, the micro:bit will permanently show the time and date until you press the A, B and A+B buttons to show atmospheric pressure, temperature and humidity, respectively (Fig.35).

For the use of this board, it is necessary to add the corresponding extension by searching for "kitronik klimate". To use the date and time functions, you must install the "kitronic-rtc" extension in Makecode.

In order to communicate this information to another micro:bit in real time, we just have to use the radio function (Fig.36), sending the name and values of each measurement.



```
forever
    radio send value "Pressure" = Read Pressure in Pa
    radio send value "Temperature" = Read Temperature in °C
    radio send value "Temperature" = Read Humidity
```

Figure 37. Code to send the measured values by radio

5. Conclusion

The CODESLASTRO project created coding competences and promoted the creation of opportunities, sensitising the school community to overcome the injustices imposed by geographical imposition, creating an enterprising and innovative project.

The authors concluded that in the initial phase of knowledge construction, autonomy and use of the programming language as a metacognitive tool and sharing, discussion, structuring, planning and interconnection with astronomy processes were predominant.

The CODESLASTRO project intended to be a contribution to alert to the importance of developing skills inherent to the concept of astronomy and programming and to demonstrate the need for its integration in the educational system, as it leads young people to relate and bring their learning closer to reality. On the other hand, it was intended to promote the formation of actors who are aware of the difficulties and opportunities associated to scientific and tecnological advances, capable of having an active and critical role on their own future, with a civic and intervening conscience and a decisive role in the construction of a fairer society for all.

Regarding the use of programming as an educational resource to develop projects in collaborative work and with the project work methodology, this educational tool, with great potential, stimulates and develops collaborativism, and is also an excellent vehicle to develop the project work methodology.

Activities involving programming and robotics, in project-based learning, allow students to develop skills for the XXI century. They can also be used to stimulate the interest and skills in STEM (Science, Technology, Engineering and Mathematics). The use of these technological tools in the learning environment, particularly in interdisciplinary processes, can be very useful and should be taken into account also in teacher training. It would be an added value if these students and teachers had the opportunity to further explore these scientific approaches.

6. Acknowledgments

This work was supported by the Portuguese Foundation for Science and Technology (FCT) in the framework of the Strategic Funding UIDB/04650/2020.

7. References

[1] Office of Astronomy for Development website: https://www.astro4dev.org

[2] Martins V, Resende A, Mateus V, Costa MFM. Robotics and Entrepreneurship for a Better Society. Opening Doors to Mobility. Hands-on Science. Hands-on The Heart of Science Education, Costa MFM, Dorrío BV, Trna J, Trnova E (Eds.), 82-90, Masaryk University, Czech Republic, 2016.

[3] Przybylla M, Romeike R. Key Competences with Physical Computing. KEYCIT 2014: key competencies in informatics and ICT, 7:351, 2015.

[4] Horn MS, Crouser RJ, Bers MU. Tangible interaction and learning: the case for a hybrid approach. Personal and Ubiquitous Computing, 16 (4):379–389, Apr. 2012.

[5] Hodges S, Scott J, Sentence S, Miller C, Villar N, Schwiderski-Grosche S, Hammil K, Johnston S. .NET Gadgeteer: a new platform for K- 12 computer science education. In Proceedings of the 44th ACM technical symposium on Computer science education, pages 391–396. ACM, 2013.

[6] Sentance S, Waite J, Hodges S, MacLeod, E, Yeomans LE (2017). "Creating Cool Stuff" - Pupils' experience of the BBC micro:bit. In Proceedings of the 48th ACM Technical Symposium on Computer Science Education: SIGCSE 2017.

[7] https://microbit.org

[8] https://www.bbc.co.uk/programmes/p02kbpx4

[9] http://makecode.microbit.org

[10] https://microbit.org/pt-br/projects/make-it-code-it/step-counter/

[11] A summary of all phases of NASA's countdown for the Space Shuttle launch can be viewed on the page: https://en.wikipedia.org/wiki/Space_Shuttle_launch_countdown

[12] https://mars.nasa.gov/mars2020/

[13] https://kitronik.co.uk/products/5683-kitronik-move-motor-for-the-bbc-micro-bit