Project number: 2023-1-PL01-KA220-SCH-000154043

IoT4Schools
"Bringing the Internet of Things in school education as a tool to address 21st century challenges"

Fitness tracker: creation of a pedometer

Teachers' guidelines

Authors: C.Papasarantou, R. Alimisi

Organization: EDUMOTIVA

Co-funded by the European Union

Table of contents

1 Introduction to the project

1.1 Scenario and Scope of the project

The aim of this project is to introduce students to the concept of IoT in the context of health monitoring and through the lens of physical activity. They will learn how to create their own pedometer (i.e., a fitness tracker device that detects vertical movement and measures the number of steps, thereby providing an approximation of the distance covered), and how to develop an application that receives data from the pedometer and displays it in real time. Specifically, using the BBC micro:bit microcontroller and the built-in accelerometer sensor, students will learn how to create and program their own pedometer that can monitor physical activity by counting steps and distance covered. Using the MIT App Inventor software (https://appinventor.mit.edu/), they will also learn how to design and program the application to receive and display the counted steps. The data will be transmitted via Bluetooth. The general concept of this project is illustrated in Figure 1. This project will help students to understand how different devices can exchange data remotely, while familiarizing them with the process of monitoring data in real-time and making decisions based on this data.

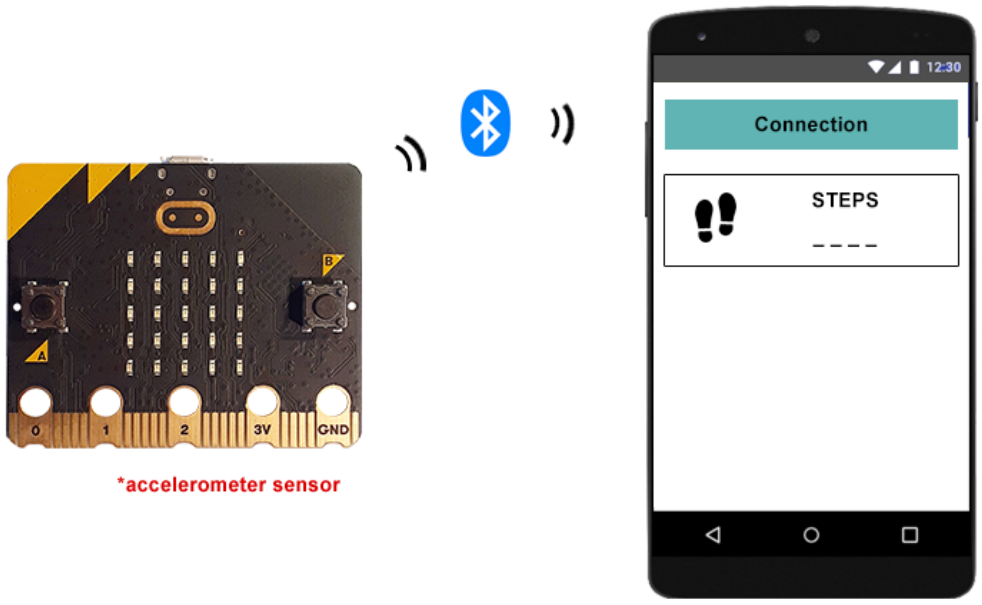


Figure 1: Graphical representation of the concept of the project

By addressing the concepts associated with the IoT and suggesting ways for integrating and applying these concepts in the context of school education, this project also promotes methods to help both teachers and students improve their digital skills and competences, thereby contributing to the digital transformation of school education. Furthermore, this project encourages interdisciplinarity and the integration of STEM principles through the implementation of practices from diverse fields, including Technology and Mathematics. Additionally, the developed educational resources (teacher's guidelines, student worksheets, etc.) provide instructional materials that can effectively support educators in the smooth implementation of the project within the classroom.

1.2 Learning objectives

Through this project the students will be able to:

- Build and program a device that can count steps
- Design and program an application that displays the number of steps
- Program the device to measure distance based on the counted steps.
- Learn how to use sensors, such as the accelerometer, to monitor data related to health (in terms of physical activity)
- Understand how IoT devices can collect and transmit data via Bluetooth
- Understand and explain how data can be monitored in real time
- Identify the advantages, disadvantages and risks using such devices and applications for making health-related decisions

1.3 Learning pathway – Stages of implementation

The project raises the challenge of health and the ways that IoT can be implemented to improve health-related issues through monitoring data relevant to the daily physical activity. To this end, the students will be encouraged to create their own pedometer by using the micro:bit board.

Here are some suggested stages to smoothly and effectively implement the pedometer project with your students:

Group formation: Divide your students in teams of two or three.

Brainstorming: Encourage each team to search more information about pedometers (i.e., how they work, what data a pedometer monitors, where this data is sent or stored, how this data can be used to monitor physical activity etc.). If the students already using pedometers encourage them to reflect on how they use this device (what data they mainly check, how often they monitor the progress of their physical activity, if and what decisions they make based on the monitored data etc.).

Discussion and assignment of the activity: Encourage each team to share their findings and ideas about the use of a pedometer in the plenary and have a general discussion based on the aspects they have highlighted. Following the discussion, introduce the specific aim of the project, namely the creation of a device that can count steps, and the development of an application that displays the counted steps. (*Note: It is recommended that the specific aim of the project is introduced after the brainstorming in order to encourage your students to consider the pedometer project in a broader context*).

Planning: Encourage each team to think about how they will construct the device (how it will be attached to the body, in which part of the body etc.), and how they will design the application (what the interface will be, what fields they will need to include etc.).

Creation: Using the student worksheets, encourage each team to create their own pedometer and develop their own application to display the counted steps. Depending on the skills of the students, you may wish to consider role allocation.

Testing - optimization: After completing the project, encourage your students to test their pedometers. You can suggest that they test another team's pedometer to see if there are any differences in the way the pedometers work. Based on the test results you can encourage each team to optimize their project.

Presentation - sharing: Encourage your students to present their projects in plenary and ask them to reflect on the whole experience. Encourage all teams to consider the impact of such devices on people's daily lives and the advantages and disadvantages of using pedometers to monitor physical activity and other health-related parameters.

1.4 Learning prerequisites

The students should be familiar with basic block-based programming methods and software. No other previous experience or learning background is required.

1.5 Hardware and software

Hardware:

- The BBC micro:bit microcontroller board
- External power source: 2AAA battery holder or a power bank of low voltage output (up to 5V)
- Heart rate sensor (optional/ only for Level 2)

Software:

- Microsoft Makecode block-based programming environment
- MIT App Inventor

1.6 Time plan

It is estimated that you will need 4 to 6 hours to complete the project

In particular, it is estimated that you will need:

- 30-40 minutes to introduce the project (including brainstorming and discussion)
- 30-40 minutes for planning and warm-up activity
- 1 to 2 hours to complete Level 1
- 1 hour for level 2
- 30 minutes for wrap-up and discussion

2 Implementation of the project

2.1 Level 1: Creating a pedometer that counts steps

2.1.1 Circuit making process

For the needs of this project, the built-in accelerometer will be used. Therefore, at the end of the project, all you need to do is connect an external power source, such as a 2AAA battery holder, to disconnect the micro:bit from the computer and turn it into a portable device.

Therefore, connect the micro:bit to the computer using a USB cable and start to program.

2.1.2 Programming

To ensure a smooth implementation of the project, it is recommended that you start with a warm-up activity, in which the students will create a pedometer that counts steps and displays the results on the micro:bit's LED screen.

Open the Microsoft Makecode block-based environment (https://makecode.microbit.org/) and create a new project.

Warm-up

Create a variable to store the counted steps. To do this, click on the **Variables** command group, and on the Make a Variable button (Figure 2a). Then, in the pop-up window (Figure 2b), type the name of the variable (i.e., steps). Two new commands will appear under the Make a Variable button: the "**set…to...**" and the "**change…by…**" commands.

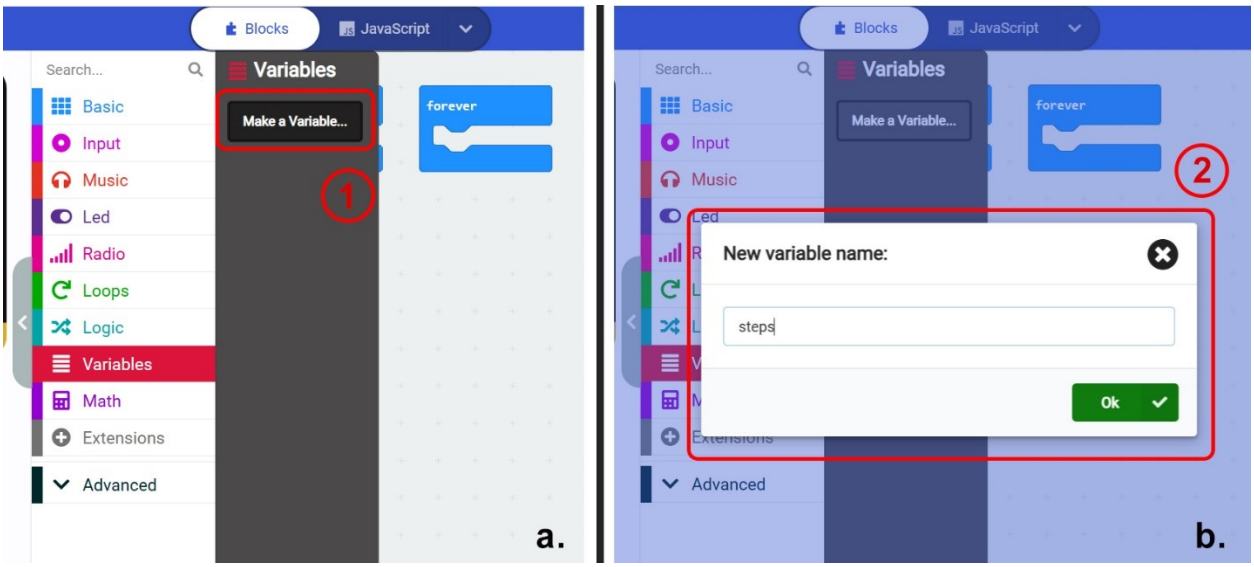
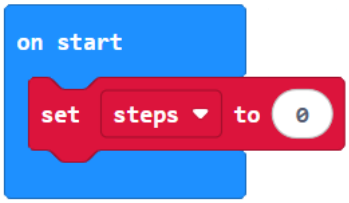


Figure 2: Creating a variable

In the "**On start**" command block, snap the "**set 'steps' to '0'**", to set the step counter to zero, each time the program starts to run.



Then drag and drop the "**On shake**" block command from the **Input** command group into the script editor area (Figure 3). This command activates an event every time the micro:bit is shaken.

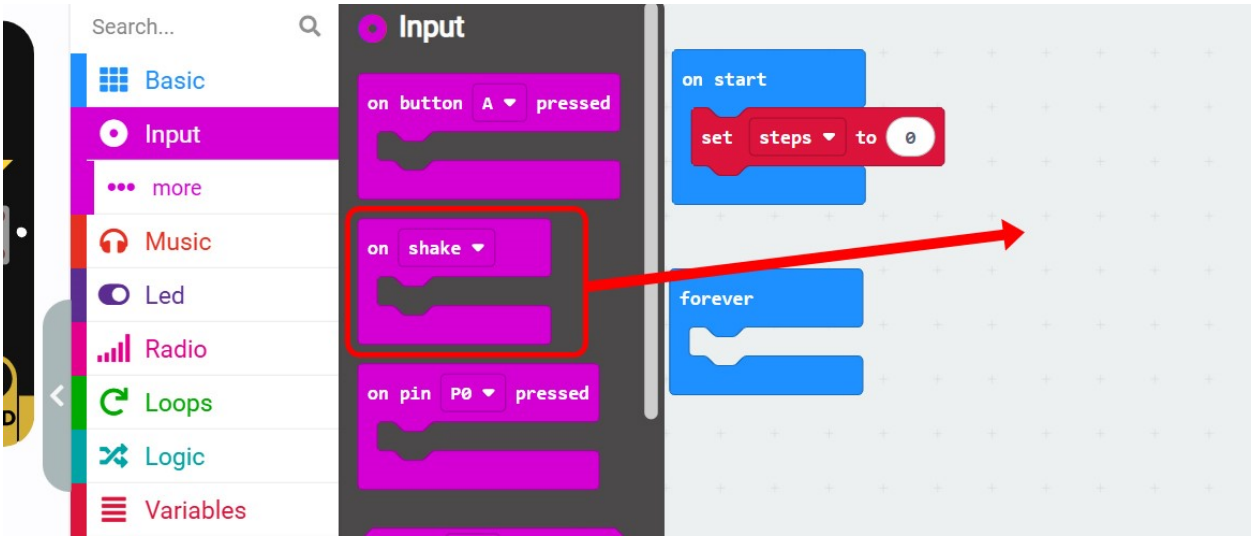


Figure 3: Finding and dragging the "On shake" command

In this command, snap the "**change 'step' by…**" command and set the value to 1. In this way, the value of the step variable will change by 1, each time the micro:bit is shaken (i.e., a movement along the vertical or horizontal axis is detected).

The pedometer is almost ready. The only thing left to do is to program the micro:bit to display the counted steps. Therefore, in the "**Forever**" command, snap the "**show number**" block command from the **Basic** menu, and place the "**step**" variable in the value field.

The final script/code should look like the following (Figure 4):

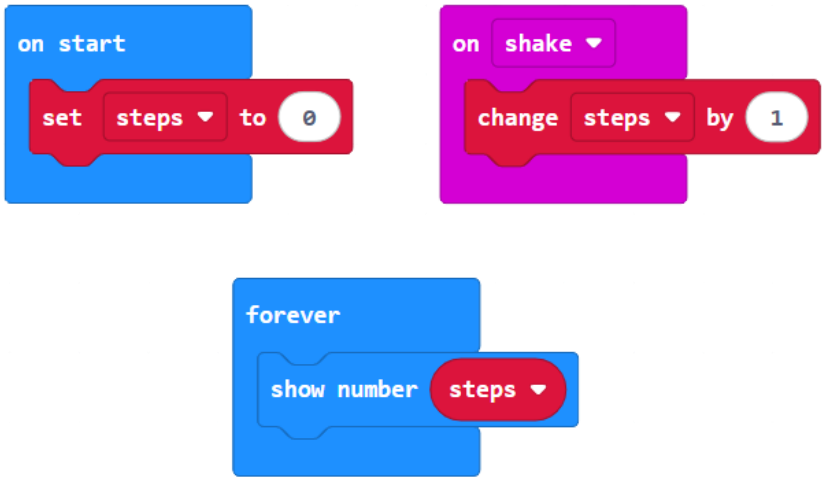


Figure 4: The script of the warm-up activity

Upload the script to the micro:bit and test how the pedometer works.

Towards an IoT solution

So far you have created a portable device that can count steps. However this data is not transferred or stored anywhere other than the micro:bit. In order to transfer this data and better monitor your steps, one solution is to create an application that receives the counted steps from the micro:bit at any given time, and displays them on your smart device. The micro:bit has a Bluetooth antenna, so it can connect to other Bluetooth devices. Therefore, you will create an application using the MIT App Inventor, to install on a smart device and exchange data with the micro:bit via Bluetooth. You will also need to make changes to the existing script of the micro:bit to enable the Bluetooth connection. For this reason, the programming solution is divided into two parts: the Makecode and the MIT App inventor programming part.

Makecode programming part

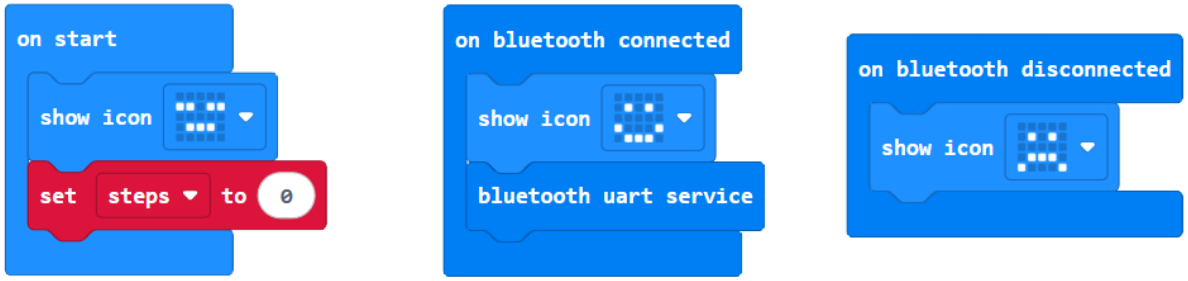
To enable the micro:bit to exchange data using the micro:bit antenna, you need to import the Bluetooth command group into your project (see how in the micro:bit technical guide file).

The script you are about to create will instruct the micro:bit to use the available Bluetooth services, when it is connected to another Bluetooth device, and to transmit the data of the counted steps when the Button A is pressed.

Tip: Before making any changes to the existing code/script, save the project under a new name

In the existing script, delete the "**forever**" block command. Then add the "**on Bluetooth connected**" block command from the **Bluetooth** command group, and snap the "**Bluetooth uart service**" command. You can also add a "**show icon**" command to display something like a smiley face (or whatever you want) on the LED screen, to indicate that the micro:bit has successfully connected to the other Bluetooth device.

For the same reason (i.e., to have a visual indication of the connectivity status of the micro:bit), you can add a sad icon inside the "**on Bluetooth disconnected**" block command and a sleepy face inside the "**on start**" block command.



The final step is to instruct micro:bit to send the counted steps when Button A is pressed. Inside the “on button A pressed” block command from the Input command group, snap the “Bluetooth uart write number…” command, and place the “steps” variable inside the value field. The final script will look like the following (Figure 5).

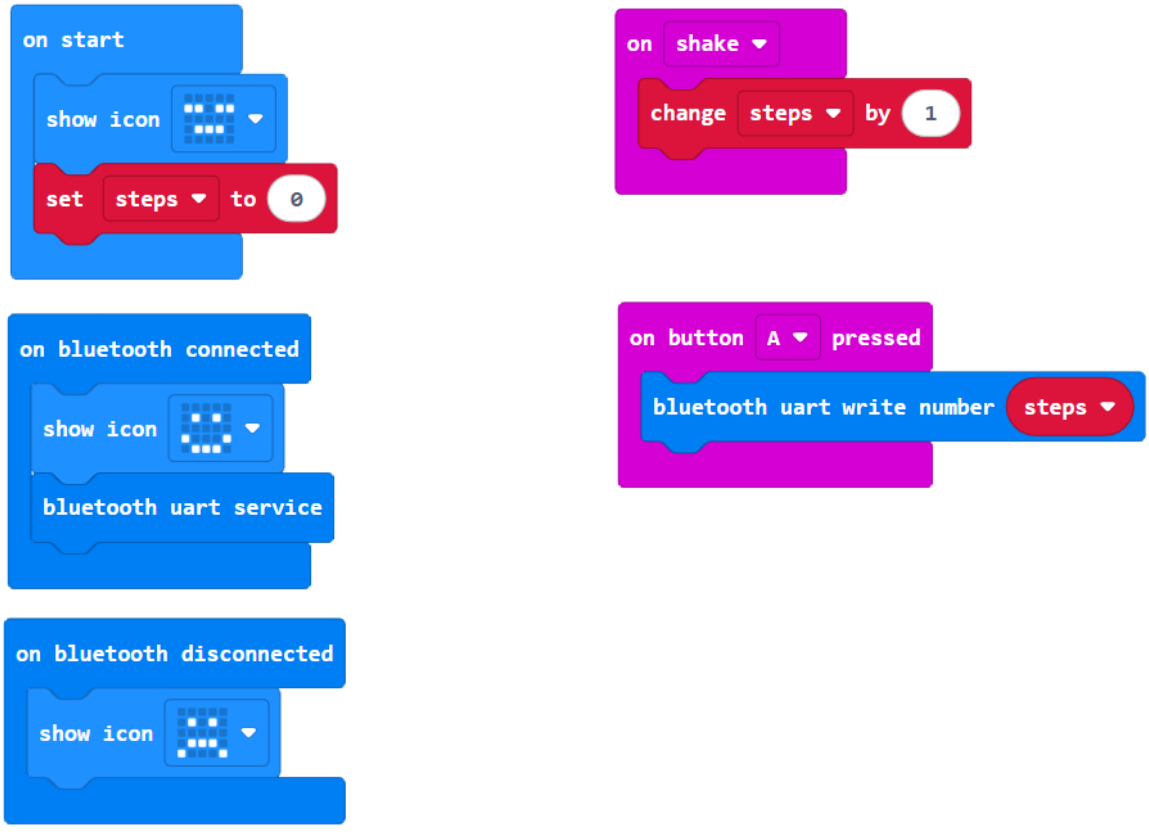


Figure 5: The final script of Level 1

You can know download the script to the micro:bit.

Important notes:

a. Before uploading the script make sure that “No Pairing Required: Anyone can connect via Bluetooth” is selected (Figure 6). You can find the pairing options by clicking on the “more” menu (i.e., the gear icon) and selecting “Project settings” from the floating menu.

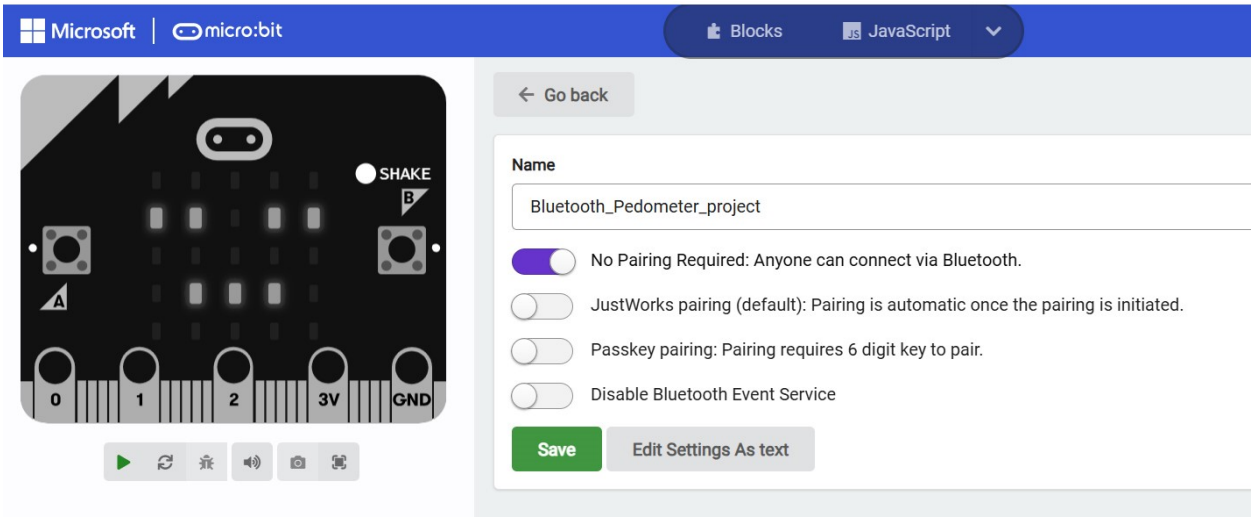


Figure 6: Pairing options

b. If the micro:bit cannot connect to the other Bluetooth device, try adding the word “Bluetooth” to the beginning of the project name, and reupload the script.

MIT App Inventor programming part

In addition to creating the pedometer device, you will also need to develop the application that will receive and display the counted steps. For this part, you will need to use the MIT App Inventor software (https://appinventor.mit.edu/). In general, the development of an application is divided into two parts: i) the design of the interface and ii) the programming of the included components. This document will focus on the programming part, as the design part can be time consuming and add to the time needed to implement the project in the classroom. Therefore, you can download and work on this file, which contains the application with the interface already designed.

However, depending on the available class time and the level of your students, you may wish to develop the application from scratch. In this case, here are detailed guidelines on how to design the interface of the application.

The following image (Figure 7) shows the components that should be programmed.

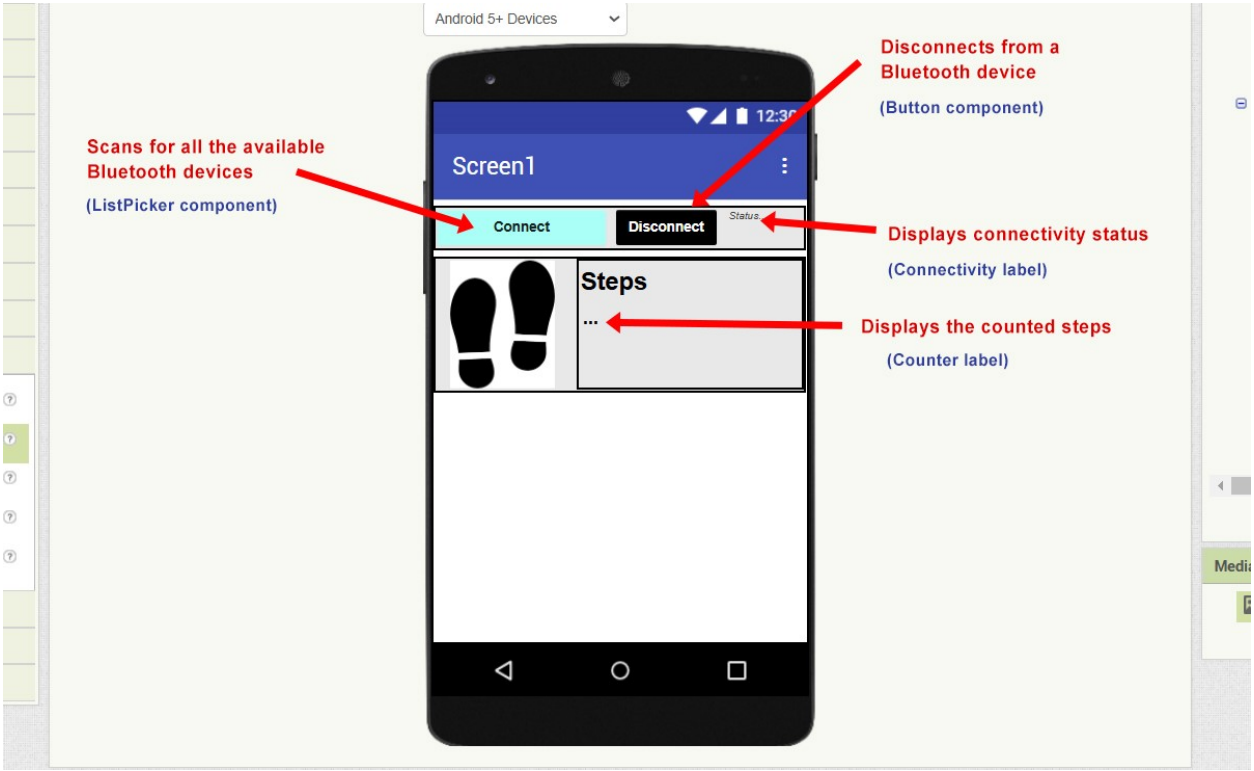


Figure 7: The components that should be programmed

Specifically, you need to program:

- the “Connect” button (which is a ListPicker component) to scan for all available Bluetooth LE devices, display them as a list, and allow the user to select the one they want to use (in our case, the micro:bit device).
- the Disconnect button (which is a button component) to disconnect the application from the connected Bluetooth device
- the “Status” (which is a label component) to change to “connected” or “disconnected” depending on the connectivity status.
- the “…” (which is also a label component) to display the counted steps, each time the Button A of the micro:bit is pressed.

Tip:

To program a component in MIT App Inventor, go to the Blocks menu and select the component you want to program from the Blocks list **(1)** (Figure 8). A menu, with all the available commands for programming each component, will appear **(2)**. Find the command you need, and drag it to the Viewer area, where you can assemble your script.

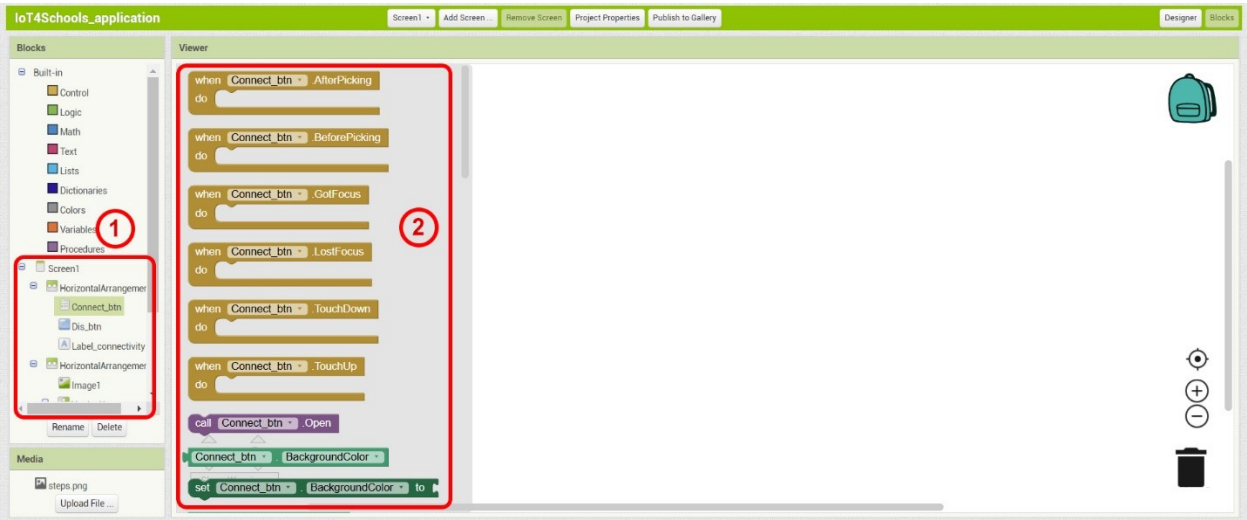


Figure 8: Finding the block of commands to program a component

Programming Connect and Disconnect buttons

Note: It is not compulsory to introduce this part of programming to your students. Depending on their level, you can decide whether to introduce it or directly provide them the programming solution.

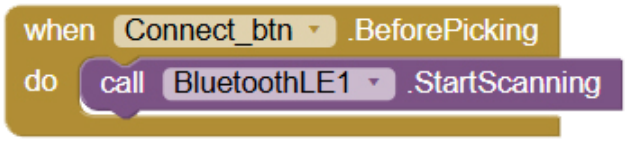
Connect button

The following diagram shows the general concept behind programming the Connect button (i.e., the ListPicker component).

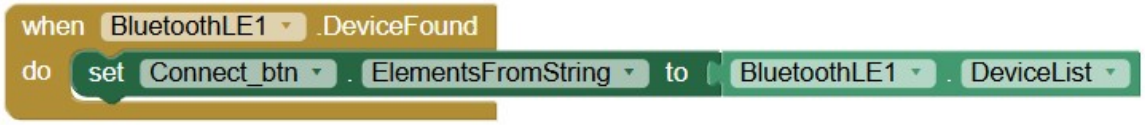
Based on this, you need to use:

- a command (BeforePicking) that will instruct the Bluetooth service to scan for available Bluetooth addresses
- a command (set) that will fill the list of the ListPicker component with the available Bluetooth addresses
- a command (AfterPicking) that will stop scanning, and connect the application to the selected Bluetooth address.

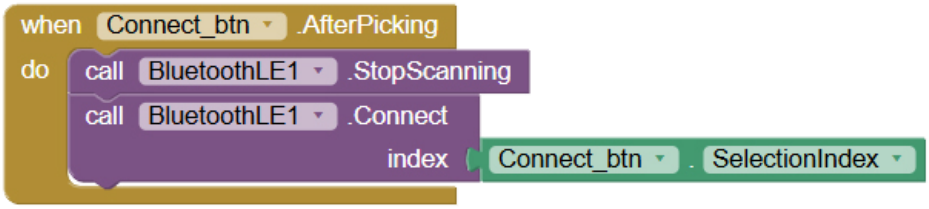
Therefore, select the “Connect_btn” component and from the floating menu, select the event command “**When Connect_btn BeforePicking**”. Then, select the BluetoothLE1 component, and from the floating menu, select the “**call BluetoothLE1. StartScanning**”, and place it inside the previous command.



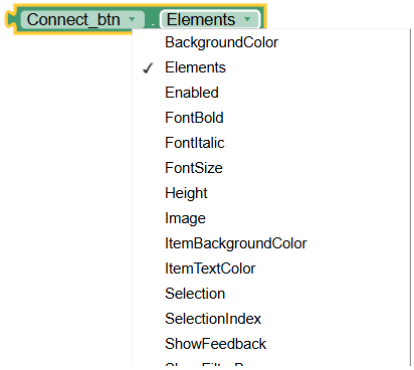
Then, select the BluetoothLE1 component and the “**When BluetoothLE1.DeviceFound**” command. Inside this command, place the “**set Connect_btn.ElementsFromString to**” command, located in the Connect_btn commands menu. Then, select the BluetoothLE1 again to find the “**BluetoothLE1.DeviceList**” command, and snap it at the end of the previous command.



Select the “Connect_btn” again and use the “**when Connect_btn.AfterPicking**” command. In this command, place the “**call BluetoothLE1.StopScanning**” and “**call BluetoothLE1.Connect index**” commands, both located in the BluetoothLE1 commands menu. Then, snap the “**Connect_btn.SelectionIndex**” command next to “**index**”.



Note: SelectionIndex is one of the available choices of the “**Connect_Btn Elements**” command.



The script for connecting your application to the micro:bit is now ready.

Disconnect button

The Disconnect button will terminate the connection to the selected Bluetooth device when pressed.

Therefore, select the “Dis_btn” component and drag the “**When Dis_btn.Click do**” command. Inside this command insert the “**call BluetoothLE1.Disconnect**” command, located in the BluetoothLE1 menu.

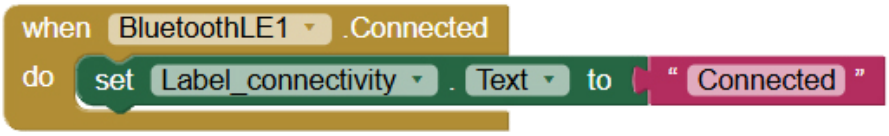
The script for the disconnect button is now ready.

In addition to the two buttons, there is also a label that will indicate whether or not the connection has been established. To do this, you need to create the following scripts:

Select the “Bluetooth” component and drag the “**When Bluetooth.Connected do**” command. Then select the “Label_connectivity” component and use the “**set Label_connectivity.Text to**” command. Then go to the text command menu, drag an empty text input command (Figure 9), and snap it next to the “**set Label_connectivity.Text to**” command. Inside this text input command type the word “Connected”.

Figure 9: Finding the text input command

Now, when the application is connected to the micro:bit, the word “Status” will change to “Connected”.



To change the word “Status” to “Disconnected”, when the connection is terminated, you need to add a similar line of code to the “When Dis_btn.Click” command.

In this way, when the Disconnect button is pressed, it will terminate the Bluetooth connection, and it will also change the “Status” to “Disconnected”.

Programming the application to receive messages from the pedometer device

The main objective, from a programming point of view, is for your students to learn how the designed application can communicate and exchange data with the micro:bit (i.e., the pedometer device). When programming the micro:bit you have instructed Button A to send a numerical value (in the case of this project, the number of counted steps) via Bluetooth when it is pressed. While connected to the pedometer device, our application can receive and display this numerical value. The label that will display this value is the "Counter". Therefore, you need to program the application to change the content of this label to the number of steps received. This can be done by programming the Microbit_Uart_Simple1 component.

Select the Microbit_Uart_Simple1 component and drag the "**when Microbit_Uart_Simple1.MessageReceived do**" command to the Viewer area. Then select the Counter component and drag the "**set Counter.Text to**" command, into the above command. Hover the cursor over the message field, and from the floating menu (Figure 10), drag and snap the "**get message**" command to the "**set Counter.Text to**" command.

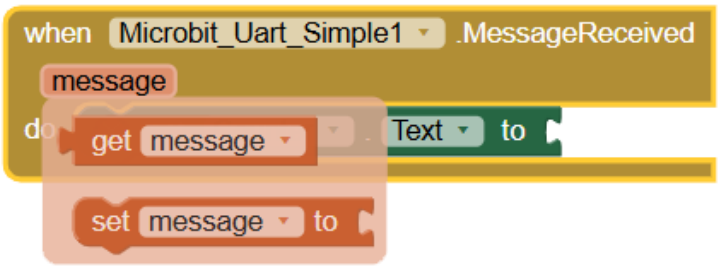
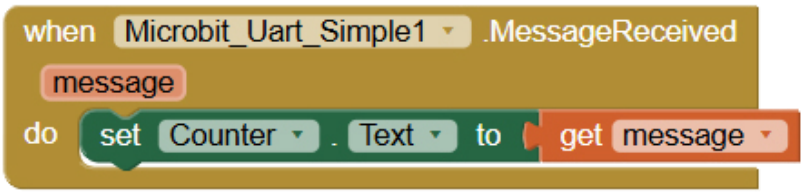
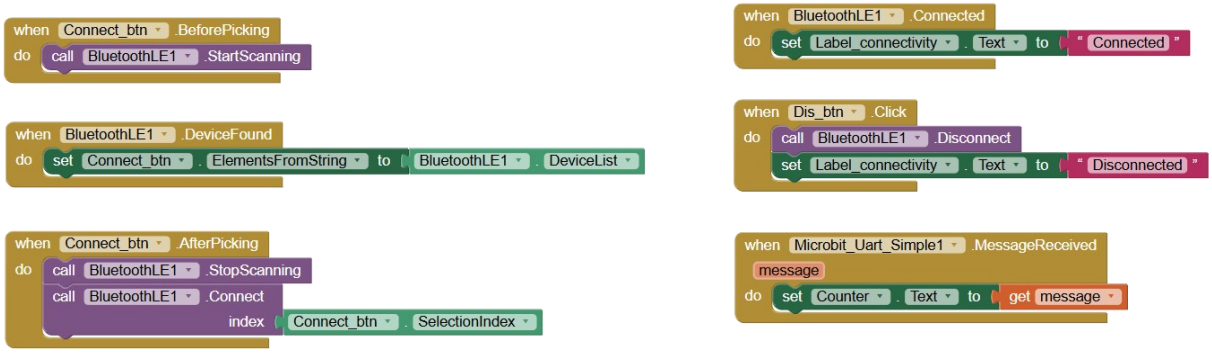


Figure 10: Finding the "get message" command

With this script, the application will display the number of steps counted each time the pedometer user presses Button A.



The following image shows the entire script of the application.



The application is now programmed and ready to be installed on a smart device such as a smart phone.

Building the application

To install the application on a smart device, you must first build it.

Click on the Build menu (Figure 11), and select Android App (apk) from the floating menu.

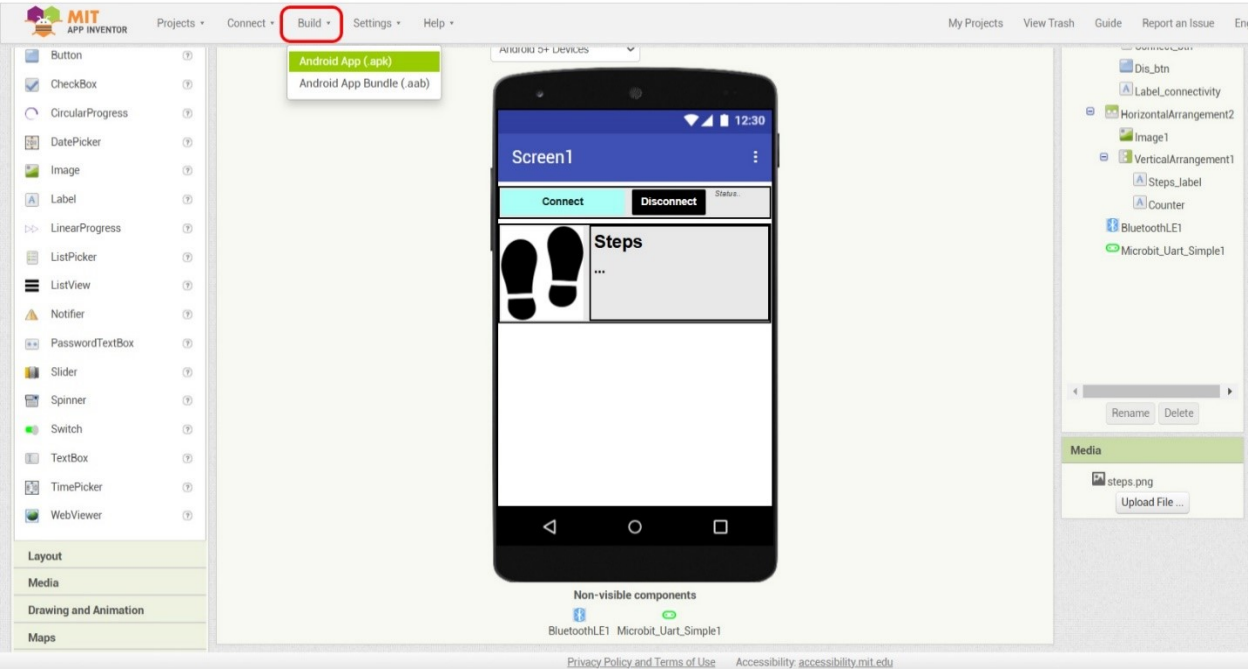


Figure 11: Building the application

This can take a few minutes. Once the process is complete, you can either download the .apk file, or scan it with your smart device, using the MIT AI2 Companion application (Figure 12).

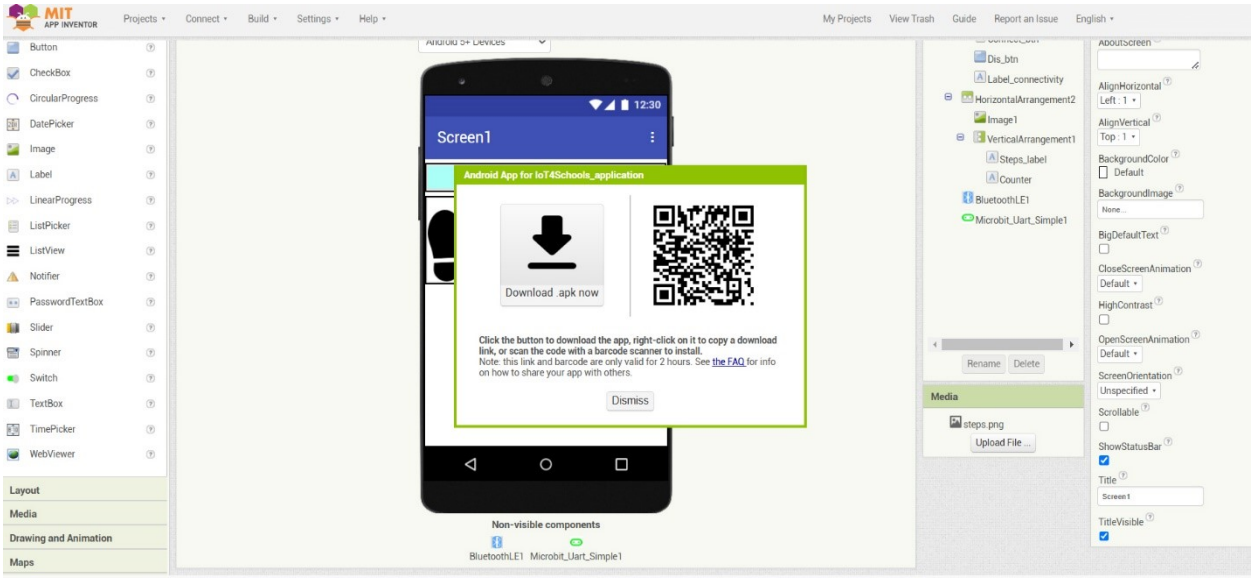


Figure 12: Download the .apk file or scan to install the application

Note 1: MIT AI2 Companion is an application/service that is available for free and can be found in the "Play Store" service of your smart device. This application is a medium that facilitates the successful installation of the generated .apk file on your smart device.

Note 2: During the installation of the .apk file, you may receive a number of messagess regarding the safety/security of this file might appear. Ignore all of them and ask your device to continue with the installation process.

Pairing the application with the pedometer

As mentioned, to connect the application to the pedometer, you need to press the "Connect" button and select the micro:bit's Bluetooth address from the list. However, it is likely that the micro:bit will not be included in the list of available Bluetooth devices. To solve this problem, go back to the main menu of the application and **activate** the "airplane" mode on your smart device for a few seconds. Then **deactivate** it and press the Scan button again. Micro:bit's Bluetooth address will now be available.

Note: Make sure "Location" is also activated to your smart device

2.1.3 Crafting

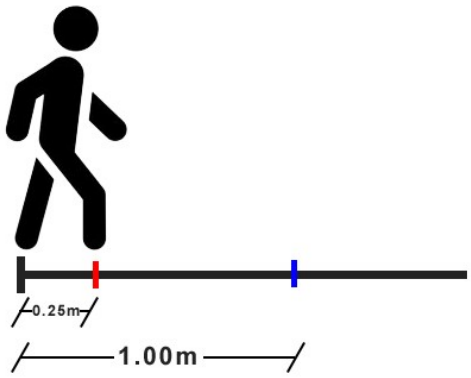
Crafting in this project is limited. Encourage your students to find ways to attach the micro:bit with the external power source to their bodies. Suggest them to attach the pedometer device to different parts of their body (arms, legs, etc.) and observe if the results are affected.

2.2 Level 2: Programming the pedometer to record multiple data

2.2.1 Making the pedometer measure the distance covered

In this level, students will learn how to make a more advanced pedometer by programming the device to measure the distance covered while walking. To do this, they will measure the distance based on the number of steps taken.

First, encourage them to measure the distance while taking only one step. Let's say a person covers 25 cm while taking only one step (as shown in the diagram below). Based on this measurement, this person would need to take approximately 4 steps to cover a distance of 1 meter.



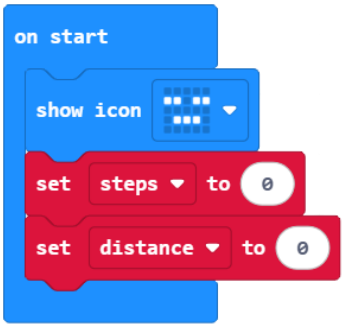
With this as a solid base, the students will make the following changes to the micro:bit script:

2.2.2 Programming

Makecode programming part

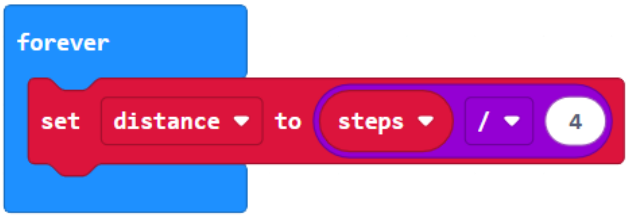
Continue working on the previous script, but first save your current work as a new file.

First, create a new variable for distance. Then add a "**set distance to 0**" command to the "**on start**" script to initialize the distance.



Then, add a "**forever**" loop in which you will include a script that calculates the distance based on the counted steps, as mentioned in the previous example (i.e., the distance covered diagram).

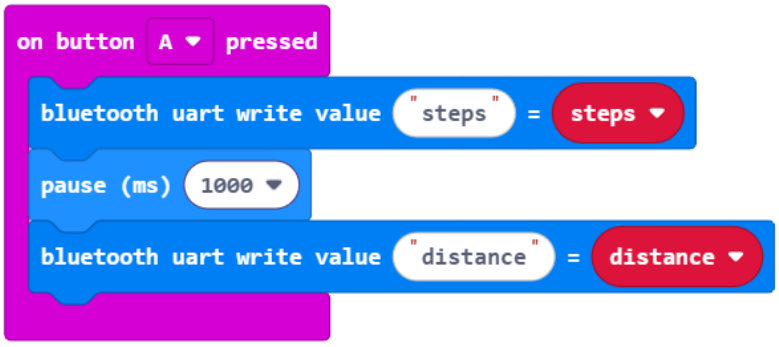
Add a "**set distance to…**" command and in the value field add a division (**…/…**) command from the **Math** command group. Then, in the dividend field, add the variable **steps** and in the divisor field, the approximate number of steps needed to cover a distance of 1 meter (e.g. the number 4 if the distance covered with only one step is approximately 0.25m).



The final step is to transfer these data to the application when Button A is pressed.

Currently, micro:bit cannot send multiple data simultaneously to an application developed with MIT App Inventor. What is feasible, is to send multiple data, by sending one data after the other.

Within the "**on button A pressed**" command, replace the "**Bluetooth uart write number**" command with two "**Bluetooth uart write value 'x' = 0**" commands. In the 'x' fields type 'steps' and 'distance', while in the 0 fields place the "**steps**" and "**distance**" variables, respectively. Between these commands add a "**pause**" command and set the duration to 1000 ms (i.e. 1 second).



Download the modified script to the micro:bit board and test the results.

MIT App Inventor programming part

You don't need to make any changes to the script in App Inventor. For consistency, you can simply change the text of the "Steps_label" component (on the Designer menu) to "Steps/Distance" or something meaningful like "measurements" or "fitness data".

3 Tips and recommendations

3.1 Further expansion of the project

Depending on the level of your students and the time available, you can extend the project to measure more fitness and health-related data by adding other variables, such as the calories burned while walking (it is estimated that a person burns about 0.04 to 0.05 calories per step), or by adding external sensors such as a heart rate sensor (for example the Gravity Heart Rate Monitor sensor which is compatible with the micro:bit).

3.2 Pedometer personalization

You can encourage your students to create their own personalized pedometers (based on the distance they cover with just one step, the different way they attached the device on their body etc.) and then share their devices with other peers to see if there are any differences in the data received.

3.3 Checking if the micro:bit can be connected via Bluetooth

Before trying to connect the micro:bit to the created application, make sure that your smart device actually recognizes the micro:bit. To do this, open the Bluetooth menu on your smart device and check that the micro:bit board appears in the list of available connections.

3.4 Discussing the advantages and disadvantages

After completing the project, encourage your students to share their thoughts and ideas about the advantages and disadvantages of using devices such as a pedometer to make decisions about our daily habits in relation to physical activity and health issues. You could also ask them what they think about sharing such data on the cloud and whether they think there are any risks in terms of personal data and security.

4 References

[1] Pedometer: https://en.wikipedia.org/wiki/Pedometer

[2] Microsoft Makecode software: https://makecode.microbit.org/

[3] MIT App Inventor: https://appinventor.mit.edu/

[4] MIT AI2 Companion App: https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3&hl=en

[5] Gravity Heart Rate Monitor Sensor: https://grobotronics.com/gravity-heart-rate-monitor-sensor-ppg.html?sl=en&srsltid=AfmBOoo77f0hnby0h2SwewfcRUGdHkYM5Xfm_NB7vAAWGHf5b2kaWUKB