

Fitness tracker: creation of a pedometer project

Worksheet for students

Team.....

Time for brainstorming

What do you know about pedometers and fitness tracker devices in general? *Work with your team to search for information online and write your findings below.*

.....

.....

.....

.....

.....

Do you use (or would you consider using) a device or application to monitor your daily physical activity?

- What data do you (or would you) check most often?
- Do you (or would you) make decisions based on this data?

Discuss with your team and write your answers below.

.....

.....

.....

.....

.....

Do you know what sensors such devices or applications use to monitor various data? How is this data transferred and/or stored?

Discuss with your team and write your answers below.

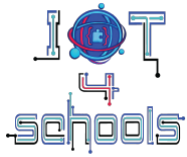
.....

.....

.....

.....

.....



[Time to design your own pedometer device](#)

Think about what data you would like your pedometer to measure and how you would attach your device to your body (bearing in mind that you will also need an external power source. *Discuss with your team and write or draw your thoughts and ideas below.*

.....

.....

.....

.....

.....

[Time to create a pedometer that can count your steps](#)

Let's get started.

Use the micro:bit's built-in accelerometer to create a device that can count your steps.

1. Open the Microsoft Makecode block-based environment (<https://makecode.microbit.org/>) and create a new project.
2. Create a variable. Name this variable "steps" (or any other name you like).
3. Assemble the following semi-structured script to make your device count one step each time the micro:bit is shaken
4. Display the counted steps on the micro:bit's LED screen



Download the script to the micro:bit and test if and how your pedometer works.

[Time to make the pedometer sending data to an application \(Level 1\)](#)

Let's use the micro:bit's built-in Bluetooth antenna to send the counted steps to an application.

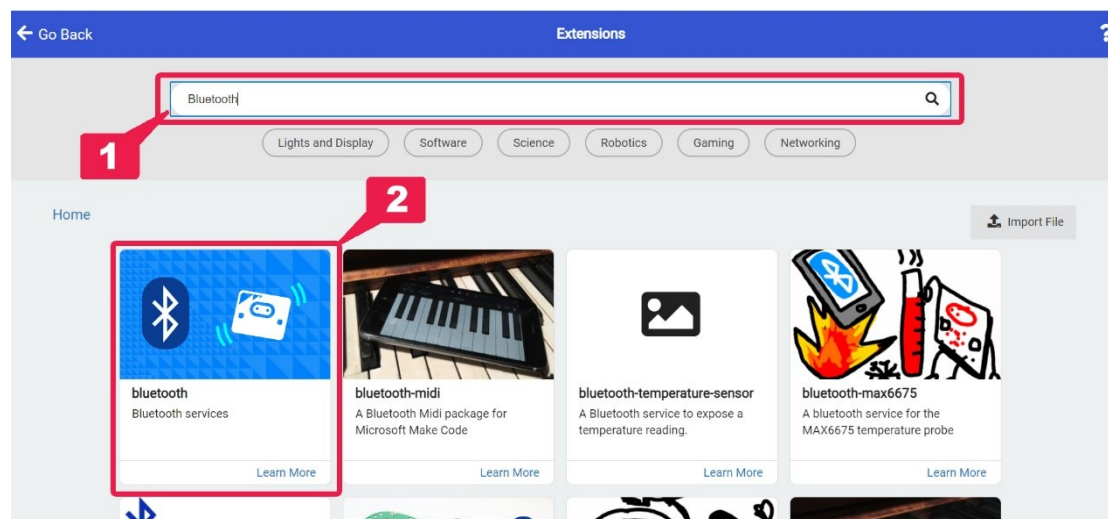
Note: Using the Bluetooth antenna, the micro:bit can send messages to other micro:bit's or other Bluetooth devices.

To do this, you need to:

- a. modify the script you have previously created
- b. develop an application to receive this data via Bluetooth

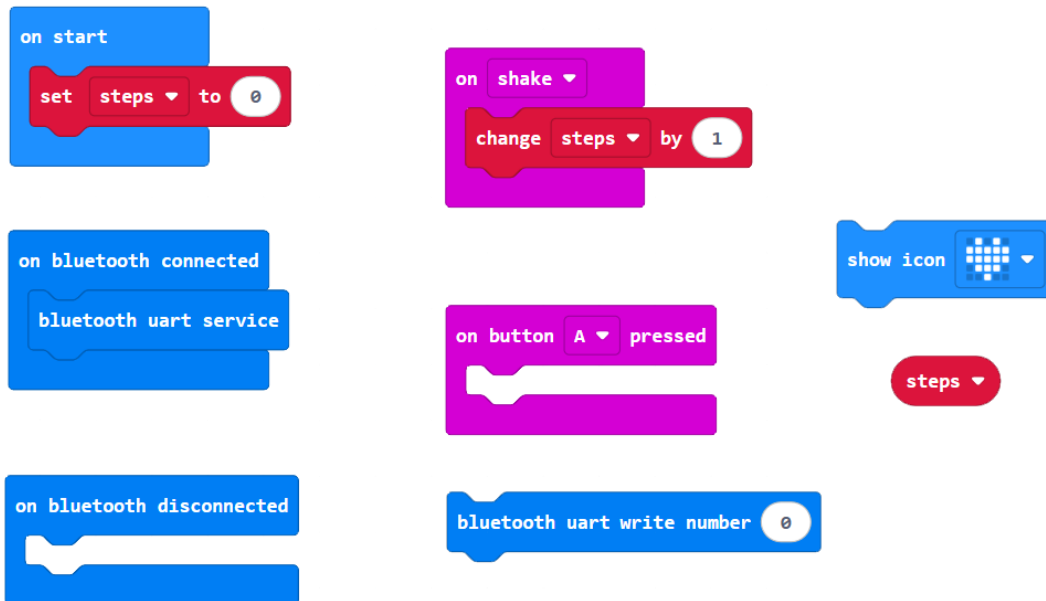
a. Modifying the previous script

From the Extensions menu, add the Bluetooth command groups to your project by typing "Bluetooth" (1) in the search bar and selecting Bluetooth (2) from the results returned.



Tip: A warning will pop-up, informing you that the radio extension is incompatible to Bluetooth and should be removed. Click on the "Remove extension(s) and add bluetooth" button to confirm your choice.

Then try to modify the previous script by assembling the commands that appear in the following semi-structured code. The aim is to create a script that will send the counted steps via Bluetooth to another device, when the micro:bits built-in Button A is pressed. Another goal is to be visually notified whether the Bluetooth connection was successful or not. This can be done by using a "show icon" command for both cases (when the Bluetooth is connected and when it is disconnected).

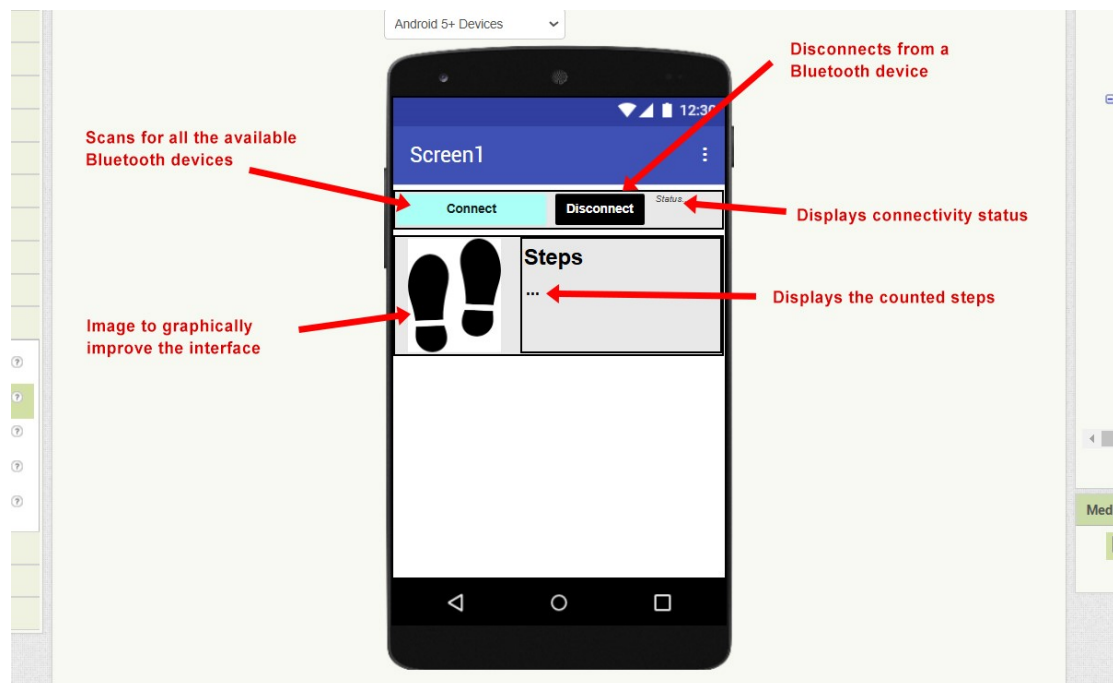


After assembling the script, download it to the micro:bit

b. Develop an application to receive data via Bluetooth

To develop the application you will use the MIT App Inventor software (<https://appinventor.mit.edu/>)

The following image shows a preview of the interface of the application you are about to develop.



As you can see, this application has two buttons (Connect and Disconnect) that enable the application to connect or disconnect from another Bluetooth device. It also has some fields that inform the user about the counted steps and the connectivity status.

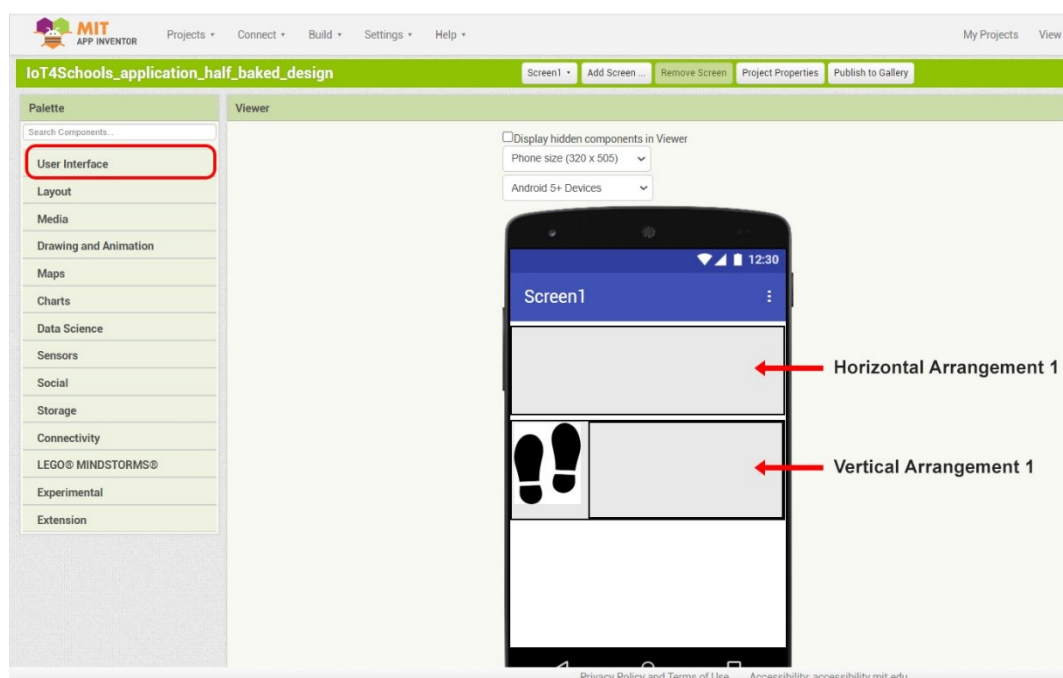
[Time to design the application \(optional\)](#)

Import the file: Import the “IoT4Schools_application_half_baked_design.aia” file into the MIT App Inventor. This file contains a half-baked version to the application you need to design. This means that some key parts are already provided, while others are missing. Your task is to add the missing elements to finalize the app

Adding components: From the User Interface tab, in the Palette menu, drag and drop the following components into the Horizontal Arrangement 1 layout:

- A ListPicker component
- A button component and
- A label component

Then drag and drop to label components into the Vertical Arrangement 1 layout.



Select each of these components and use the properties menu to modify their text and font (if you wish). It is also a good practice to rename each of these components to something meaningful (e.g. ListPicker1 to Connect_btn). This will help you later when programming.

The following table contains an indicative text and name for each component

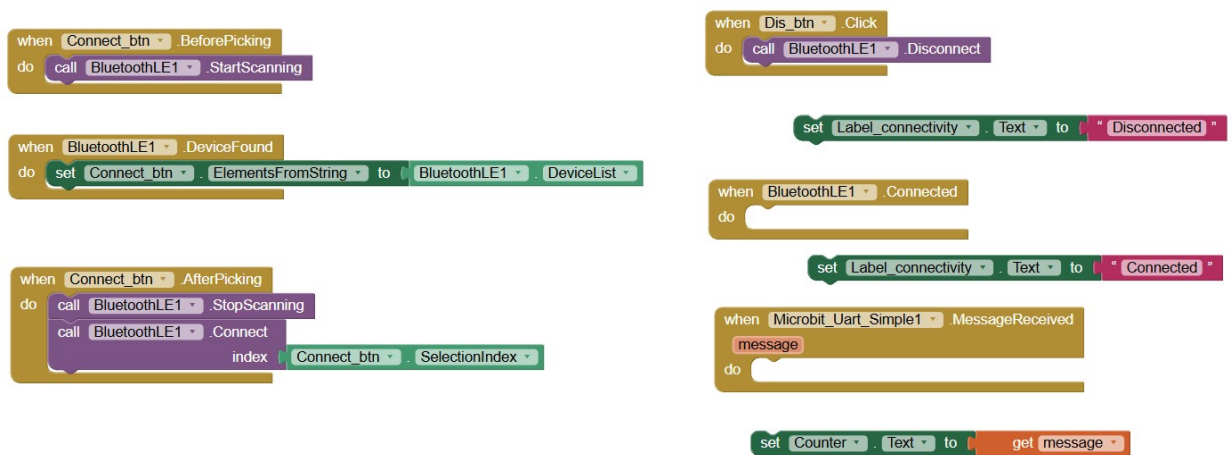
Component	Text (properties menu)	Name (All components menu)
ListPicker1	Connect	Connect_btn
Button1	Disconnect	Dis_btn
Label1	Status	Label_connectivity
Label2	Steps	Label_steps
Label3	...	Counter

Time to program the application

Import the “IoT4Schools_application_design.aia” file or continue working on the previous file.

Then go to the Blocks menu, to program the application.

The following script is semi-structured. Try to assemble it so that the “status” label (i.e. label_connectivity) changes to “Connected” when the Bluetooth is connected, and to “Disconnected” when the disconnect button is pressed. You will also need to program the application to display the number of steps counted when it receives such a message from the micro:bit.



```

when Connect_btn BeforePicking
do
  call BluetoothLE1 StartScanning

when BluetoothLE1 DeviceFound
do
  set Connect_btn ElementsFromString to BluetoothLE1 DeviceList

when Connect_btn AfterPicking
do
  call BluetoothLE1 StopScanning
  call BluetoothLE1 Connect index Connect_btn SelectionIndex

when Dis_btn Click
do
  call BluetoothLE1 Disconnect

set Label_connectivity Text to "Disconnected"

when BluetoothLE1 Connected
do
  set Label_connectivity Text to "Connected"

when Microbit_Uart_Simple1 MessageReceived
do
  message
  set Counter Text to get message
  
```

After compiling the script, build and install the application on a smart device, with the help of your teacher.

Then connect the application to the micro:bit to test your pedometer device.

[Time to make the pedometer sending various data to the application \(Level 2\)](#)

Let's program the micro:bit to send various data relevant to physical activity.

Distance covered calculation

Measure your step length (*or stride length*).

Write the distance here:

.....

Based on your step length, how many steps are required to cover the distance of 1 meter?

Write your answer here:

.....

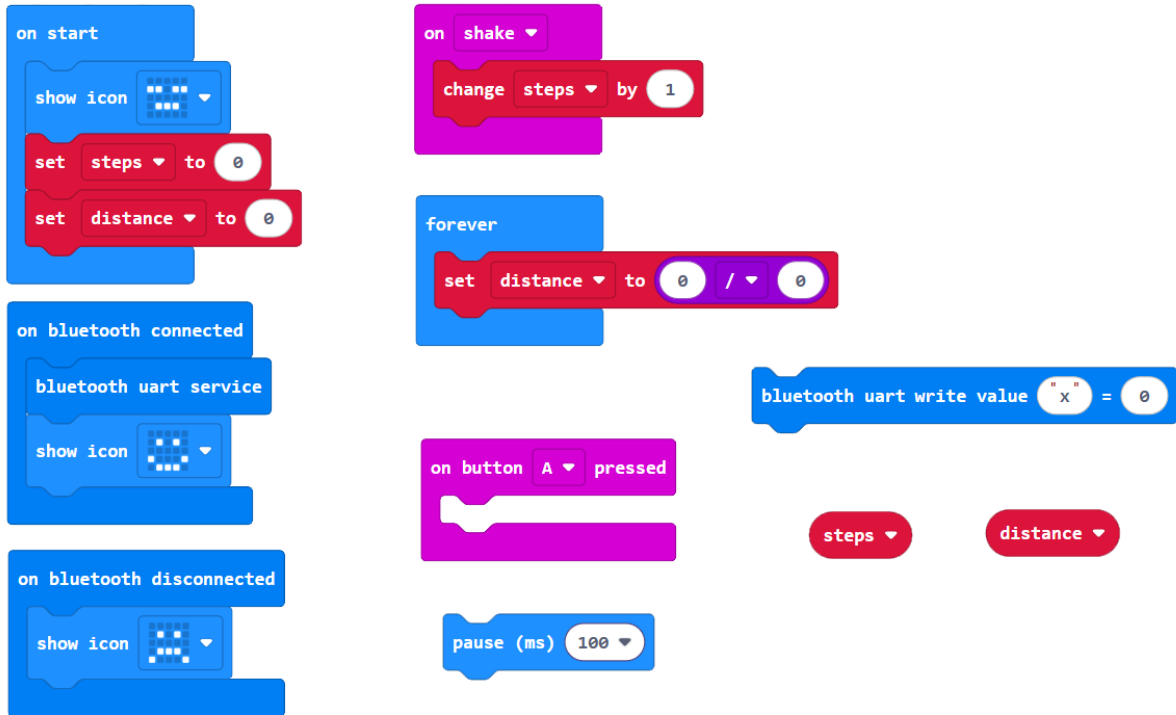
Go to Makecode and continue working on the previous script.

Create a new variable and name it "distance" (or any other name you like).

Then try to assemble the following semi-structured script so as:

- The distance is calculated on the basis of the quotient of the number of steps counted divided by the number of steps required to cover a distance of 1 meter.
- When Button A is pressed, the pedometer sends to the application:
 - the number of counted steps,
 - and after a while (e.g 1000ms), the distance covered.

Note: use the respective commands as many times as necessary



Download the script to the micro:bit and try to send the new data to the pedometer.

Do you need to make any changes to the application? Write your answer below

.....

.....

Wrapping up: reflecting on functionality and possible improvements

Can you think of other parameters related to physical activity that could be measured and monitored by your pedometer? Write your answer below and feel free to experiment further.

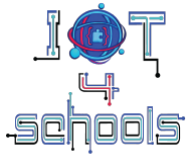
.....

.....

In general, are there any limitations or areas where your pedometer could be improved? Consider any inaccuracies or challenges you faced while using the pedometer and suggest possible solutions.

.....

.....



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