DRONES@STEAM

Fostering digital Transformation in VET schools
and creating new job prospects in the labour market

Project Result No: 2

Activity 3: EDUCATIONAL PACK: TEACHING MATERIAL AND ASSESSMENT

UNIT 2, Chapter 2.4, Worksheet 2.4.2

Lead partner(s): University of Crete

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

CONTEXT

Grant agreement	2021-1-EL01-KA220-VET-000034686
Programme	Erasmus+
Key action	Cooperation for innovation and the exchange of good practices
Action	Strategic Partnerships
Project acronym	DRONES@STEAM
Project title	DRONES@STEAM: Fostering digital Transformation in VET schools and creating new job prospects in the labour market
Project starting date	28/02/2022
Project duration	28 months
Project end date	27/06/2024

WEBSITE:

https://dronesteam.eu/

CONSORTIUM: PARTNER LIST

- University of Crete (UoC) - Greece
- ECAM-EPMI (ECAM) - France
- Cyprus Computer Society (CCS) - Cyprus
- Politeknika Ikastegia Txorierri S. Coop (PIT) – Spain
- National Center for Scientific Research "Demokritos" (NCSRD) - Greece
- A & A Emphasys Interactive Solutions Ltd (EMP) – Cyprus
- Regional Directorate of Primary and Secondary Education of Attica (RDPSEA) – Greece

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Contents

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Lesson Plan 2.4.2

UNIT 2	
Chapter 2.4	**Drone flying and remote-control programming**
Equipment, Software, Consumables (if needed)	PC with access to the internet, a second micro:bit board, Air:bit Programmable Drone Kit
Duration	3 teaching hours
Short description	Program the required remote control functions in order to communicate and pilot the Air:bit DIY drone.
Learning Outcomes	Understand the basic principles of transmitting and receiving data
	Learn how to send and receive signals using the micro:bit
	Recognize and program the main functions (radio transmitter code): start/stop the propellers, pitch, roll, yaw, throttle.
Activities	
Activity 1	Worksheet 2.4.2.1 Transmitting and receiving data
Aim of the activity	Communication using the micro:bit radio and Bluetooth.
Duration	1 hour
Type of Activity	Worksheet
Teaching Objectives	Learn how two (or more) micro:bit boards can communicate with each other, including some useful applications (exchanging messages, proximity alert etc.)
Resources	Worksheet 2.4.2.1 / Exercise 1, Exercise 2
Activity 2	Worksheet 2.4.2.2 A remote control program
Aim of the activity	Develop a basic radio control program using a second micro:bit board as a controller
Duration	2 hours
Type of Activity	Worksheet
Teaching Objectives	Learn to code the main functions needed in a radio transmitter to start/stop the propellers, and control the pitch, roll, yaw and throttle.
Resources	Worksheet 2.4.2.2 / Exercise 3, Exercise 4
Further Reading	

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Activity worksheet 2.4.2 (student version)

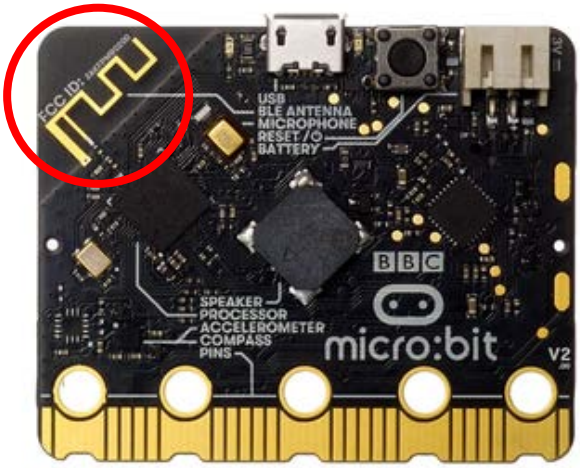
Chapter 2.4: Drone flying and remote-control programming

Level: Intermediate

Worksheet 2.4.2.1 Transmitting and receiving data

Radio communication between micro:bits

For sending and receiving messages or data, micro:bits use radio signals to communicate wirelessly with each other. Every micro:bit has an antenna that can detect or generate radio waves.



To use the radio feature more effectively and avoid interferences or mixing signals sent from other micro:bits, you can program your micro:bit to send/receive messages to/from a specific channel. This way, two or more micro:bits can belong to the same group. Groups are like channels on an FM radio or walkie-talkie and there are 256 different channels available (a number between 0 and 255). It is not important what number you pick as long as all micro:bits are using the same group number, and no one else nearby is using the same group number for other applications.

Exercise 1: Exchanging messages

In this exercise, we will send a simple message to every micro:bit belonging to the same group. The receiving micro:bit(s) will show the message on the LED display.

From the “Basic” Toolbox category use an “on start” block and drag into it a “radio set group” from the “Radio” Toolbox category. This defines the channel over which we can send or receive messages. Set the group number to 14.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Drag an “on button A pressed” block from the “Input” Toolbox category onto the workspace and put a “radio send string” into it with a specific message inside the quotes (e.g. “DRONES@STEAM”).

Drag an “on radio received receivedstring” block from the “Radio” Toolbox category onto the workspace and a “show string” block into it. Drag the “receivedString” variable from the “on radio received receivedstring” block and put it into the “show string” block.

Test your code first in the on-screen simulator by pressing the A button. You should see the message “Hello” scrolling on the second micro:bit. Both micro:bits are able to send and receive the same message.

Now you can download your code onto the micro:bit as well. The same program must be downloaded to each and every micro:bit device (two or more).

Exercise 2: Sending Temperature Data

In this exercise, we will send temperature readings from one micro:bit to a second micro:bit that will receive and display the data on its LED screen.

Code for the transmitter (sender):

Use an “on start” block and set the group number to 14. Create a variable called “myTemp” and use the internal temperature sensor to set its value (see Worksheet 1.3.1). Show the value of “myTemp” on the LED display and send the same value to a second micro:bit. Allow at least three seconds between temperature readings and transmissions.

Code for the receiver:

Use an “on start” block and set the group number to 14. Use an “on radio received” block to get the temperature data. If the received number (temperature) is between -3 and 43 degrees then display it on the LED screen. Otherwise, show an “X” icon on the screen.

Test your code first in the on-screen simulator and then you can download your code onto the two micro:bits (transmitter and receiver).

Exercise 3: Proximity alert

In this exercise, we will use the radio feature to detect how close another micro:bit is. The idea here is that signals get stronger the closer you are to the transmitter, so if the signal is strong it means the other micro:bit is close, if it is weak, the other micro:bit is further away.

We will develop two different programs (one for the transmitter and one for the receiver) and at least two micro:bits. We will store the strength of the signal in a variable and show it on the LED display.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Code for the transmitter:

Drag a “radio set group” block from the “Radio” Toolbox category into the “on start” block and set the group number to 14. Drag a “radio set transmit power” block (it can be found on the Radio->more category) into the “on start” block below the “radio set group” block and set the transmit power value to 1, which is low but works better over small distances. You can experiment with different power values.

Use the “forever” block to keep sending a message. Drag a “radio send string” into it and transmit any message e.g. the character “a” (it doesn’t matter what you actually send). It’s better to add a “pause” block to avoid sending too many messages too quickly. Setting the value to 250ms (i.e. 4 times a second) is enough.

Alternatively, instead of using a “forever” block, you can use an “every 250ms” block from the “Loops” Toolbox category. In this case, the “pause” block is no longer necessary.

Download your code onto the transmitter micro:bit.

Code for the receiver:

The main idea is the following: When the receiver picks up a message from the transmitter, it stores its strength in a variable called signal and shows it on its LED display.

Drag a “radio set group” block from the “Radio” Toolbox category into the “on start” block and set the group number to 14.

Drag an “on radio received receivedstring” block from the “Radio” Toolbox category onto the workspace, make a new variable called “signal” and drag the “set signal to” block into the “on radio received receivedstring” block. Drag a “received packet signal strength” block inside the value area of the “set signal to” block.

The signal strength can take values from -128 to -28 (-128 means a weak signal and -28 means a strong one.)

To show the signal strength on the LED display in a more human-readable format, we can map the values of the “signal” variable to a range between 0 and 9. Drag a “map” block from the “Math” Toolkit category and insert the values “from low”: -100, “high”: -40, “to low”: 0, “high”: 9. Drag a “show number” block from the “Basic” toolkit and put the “map” block inside it. Put the “show number” block below the “set signal to” block.

Download your code onto the receiver micro:bit and test it out.

The above code only works on the micro:bit, **not** in browser simulators.

You can notice that the refresh rate on the LED screen is low since scrolling numbers are displayed very slowly. For faster response times we can alternatively use a bar graph. To do that, use a “plot bar graph” block from the “Led” Toolkit category instead of a “show number”

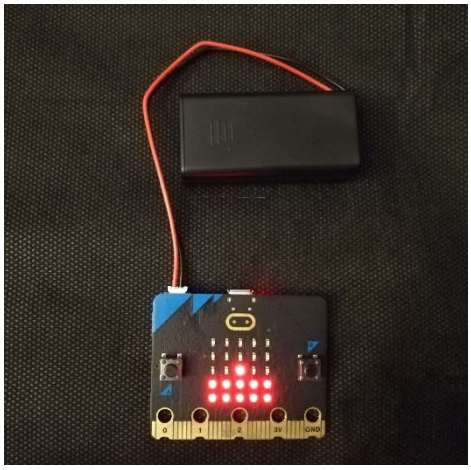
Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

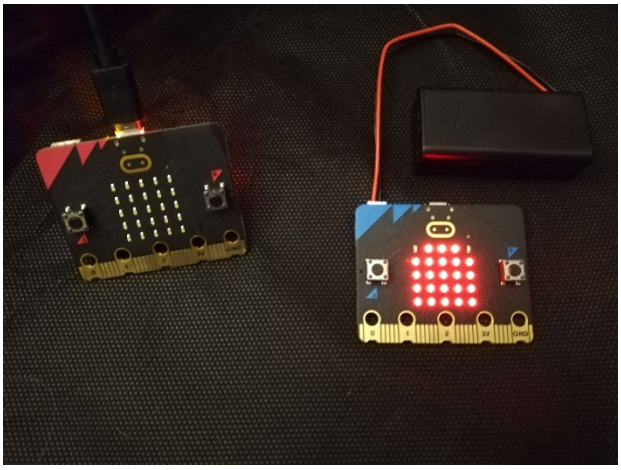
block. Drag the whole “map” block inside the “plot bar graph of” block and the value 9 into the “up to” section.

It displays a bar graph which gets bigger the stronger the signal and the closer you are. It uses the map block to map radio signal strength numbers from the range -100 (weakest) to -40 (stronger) to a range of 0 to 9 so we can use them to draw a bar graph.

Download the new code onto the receiver micro:bit and test it out.



Receiver when the transmitter is far away



Receiver when the transmitter is close

You can experiment by changing the transmitter power (it can be any number from 0 to 7).

Try to modify the code so the receiver micro:bit will sound a proximity alarm when the transmitter micro:bit gets too close.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Bluetooth communication with the micro:bit

The micro:bit can also be programmed with MS MakeCode using an Android device or an iPad/iPhone. To do that, you need to download the corresponding free app which sends code to your micro:bit using a Bluetooth connection.

You can download the app from ***Google Play*** here (requires Android 5 or later) or from the Apple ***App Store*** here (requires iOS 9 or later).

After downloading the app and starting coding, you will need to **pair** your device (phone or tablet) with the micro:bit. The app searches for a signal from your micro:bit and then they exchange a unique code to make sure the right micro:bit is paired with your app. The app contains instructions for this procedure.

Hints on Bluetooth pairing with the micro:bit

- Pair your micro:bit and mobile device **every time** you transfer a MakeCode program.
- During pairing, make sure you hold down the A and B buttons together long enough for all the LEDs on the micro:bit display to light up.
- In case you are using batteries to power your micro:bit, make sure they are fresh. Even if the micro:bit seems to be working, it may not have enough power for the Bluetooth radio function to work properly.

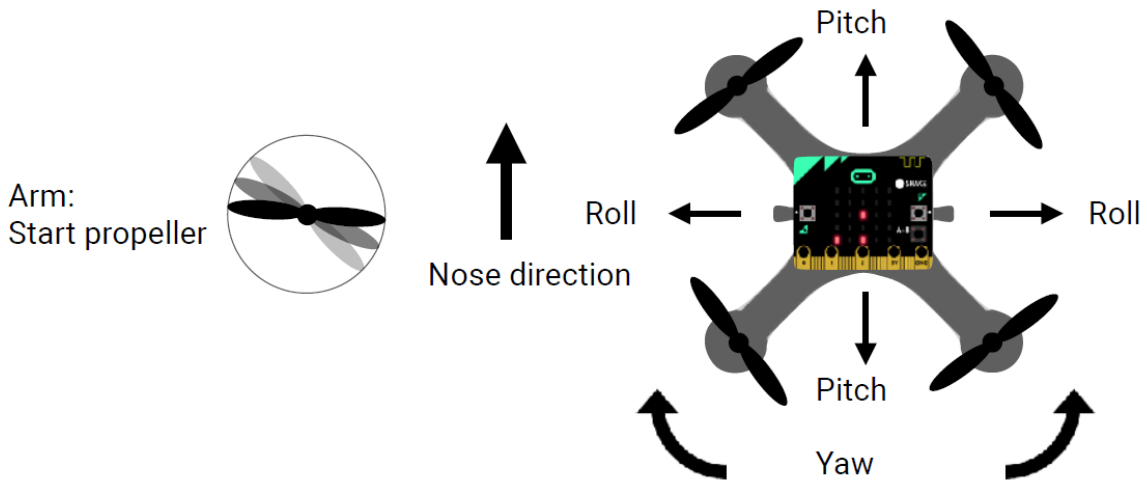
More information and exercises can be found in the next worksheet 2.4.3.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Worksheet 2.4.2.2 A remote control program

As discussed previously (see Chapter 2.2), we have five (5) very important parameters for drone flying: **P**itch, **A**rm, **R**oll, **T**hrottle, and **Y**aw (called **PARTY** in short).



For each one of them, we need to write code and control the corresponding parameter values from the remote control (i.e. the second micro:bit).

Initialize the radioGroup

The first step is to set the radioGroup to a specific number (e.g. 14) and display this number on the LED screen. This action should be done “on start”.

We are going to use four (4) new variables called “Arm”, “Throttle”, “Pith” and “Roll”.

Arm

To start and stop the propellers we need a special function. A good choice is to use both buttons A and B pressed together at the same time to either start or stop the propellers. We can use a Boolean variable called “Arm” for this purpose. Check the value of “Arm” to set it to 1 if it’s 0 or 0 if it’s 1. In either case, set the value of the “Throttle” variable to 0.

For safety precautions, we could use a few different ways to implement an “emergency stop” for shutting down the drone e.g. in case of an imminent crash. To do that, we need to set both the “Arm” and “Throttle” variables to 0.

Do that using at least 3 different ways: a) shaking the micro:bit controller, b) putting the micro:bit face down (with the screen towards the ground), and c) if the micro:bit falls out of your hands (free fall).

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Throttle

To increase or decrease the “Throttle”, we are going to use the A and B buttons of the micro:bit respectively. Pressing button A decreases the “Throttle” by 5 (i.e. -5), while pressing button B increases the “Throttle” by 5. Use the “on button pressed” and “change variable by” blocks.

Pitch, Roll, display, and send the values over the radio

Inside the “forever” loop we need to add the following:

a) Set the “Pitch” variable to the return value of the “rotation (°)” function
b) Set the “Roll” variable to the return value of the “rotation (°)” function
c) Clear the LED screen
d) Initialize the plot to x=0 and y=0 if the “Arm” variable is equal to 1
e) Plot the “Throttle”, “Roll” and “Pitch” variables. To display correctly these variables on the LED screen, use the “plot x y” function and the “map” block as follows:

f) Send the values of “Arm”, “Pitch”, “Roll” and “Throttle” variables over the radio, using the letters “A”, “P”, “R” and “T” respectively and the “radio send value” block.

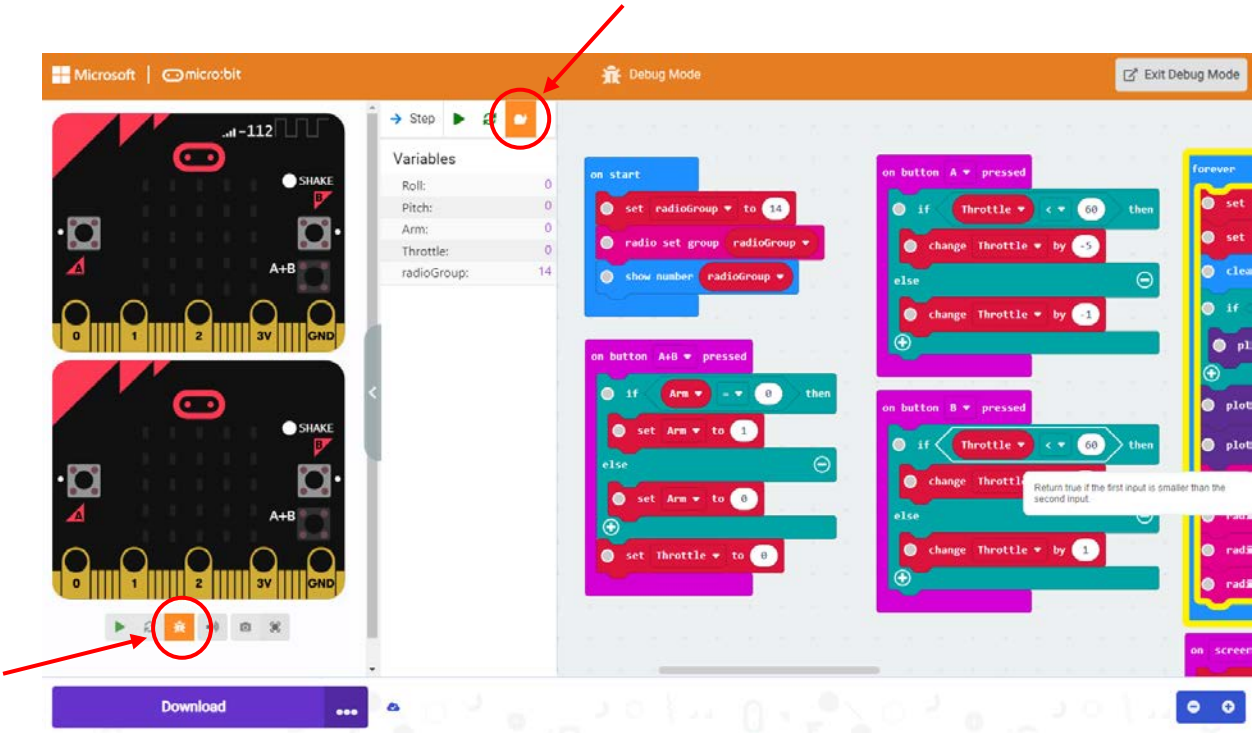
Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686

Exercise 4: Checking and debugging your code

As a first step to debug your code, before downloading and testing your code to the micro:bit, use the “Debug mode” on the MakeCode environment to check the values of all the variables.

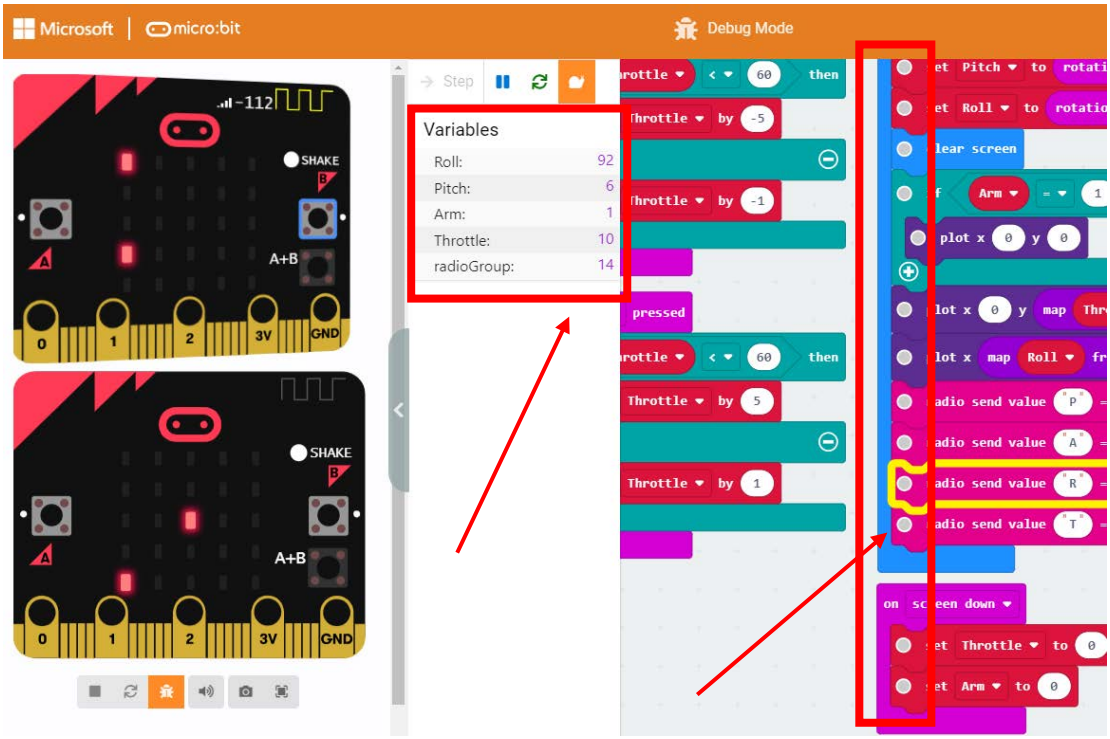
Click on the little bug icon under the simulated micro:bit on the left to enter the Debug Mode. Then click on the “Slow-Mo” icon (the little snail button).



This feature allows us to see step-by-step (in slow motion) the sequence of execution for each command and block. After a few seconds, you can see the values of all your variables in the middle column. Try to push a few buttons (e.g. A, B, A+B) and move the simulated micro:bit around to watch how the values change. Be sure to try every combination of buttons and any movement to verify that everything is working properly.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686



You can start, pause, and restart the execution, execute one-by-one the commands ("→Step" button), or even set a breakpoint to any command you choose by selecting the radio button on the left of each command (toggling the breakpoint on and off).

What is a breakpoint?
In software development, breakpoints are temporary markers that you place in your program to stop the execution at a given point. It's an intentional stop or pause in a program, set by the programmer, for debugging purposes.

Exercise 5 (homework assignment)

Write a program on a receiving micro:bit to test and/or debug your code. Check and display the values of "A", "P", "R" and "T" that are transmitted by the remote controller micro:bit.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein. Project number: 2021-1-EL01-KA220-VET-000034686