Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

The DIY robotic car project

Introducing the 5 Big Ideas in Artificial Intelligence using Internet of Things in STEM education

T2.4 IoT Projects Design & Resources Development

06.10.2023 | EDUMOTIVA
PROJECT NUMBER: 2022-1-FR01-KA220-SCH-000085611

AI4STEM IoT Projects Design & Resources Development
Project: The DIY robotic car

Copyright

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Table of Contents

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

1.Introduction to the Project

The present project focuses on the creation of a DIY robotic car that can be controlled by voice commands, while being able to collect a number of data (such as the temperature, the light level, the distance, the acceleration etc.) that can lead to certain decisions to optimize its performance. This project-based learning intervention will help both teachers and students to be introduced to the fields of AI and IoT, through the lens of the 5 Big Ideas (namely Perception, Representation and Reasoning, Learning, Natural Interaction, and Societal Impact), as well as in the light of Robotics by implementing a series of hands-on and computer-based practices and activities. With regard to AI, they will be introduced to Speech Recognition and how this service can be used to navigate a robotic artifact. To this end, they will be introduced to the processes of designing and programming an AI-based application using the MIT App Inventor software, as well as with training a model to classify the incoming information using the Personal Audio Classifier environment. Concerning Robotics, they will learn how to construct a robotic car by using the BBC micro:bit microcontroller and several compatible electronic components, as well as how to program this robotic artefact using the block-based software Makecode. The project will be divided into 5 Activities. Each of these activities will revolve around one of the 5 Big Ideas. During these activities the students will be invited to focus on different parts of the implementation process and engage with different aspects of AI and IoT. The activities will include guidelines for teachers and several suggested tasks for students to ensure a smooth introduction and implementation of all the aforementioned concepts and their inherent aspects, ultimately leading to the acquisition and development of several 21st century skills such as creativity, critical thinking, problem solving and collaboration.

1.1 The scope of the Project

Through this project, the students will be introduced to the field of AI and the 5 Big Ideas, as well as the field of IoT, in the light of robotics. In particular, through the 5 activities and the associated tasks, which revolve around the construction and the programming of the robotic car, as well as the design and programming of the AI application, the students will better understand the 5 Big Ideas and will delve into some of the core mechanisms of AI and IoT. More specifically, in the 1st Activity the students will be introduced to IoT by learning how to program their robotic car to collect and monitor a number of environmental data through an IoT analytics platform service (i.e., ThingSpeak). Through the tasks contained therein, they will be introduced to the idea of Perception. In the 2nd Activity, students will be introduced to the idea of Representation and Reasoning by learning how the robotic car can “think”, and how data can be represented in multiple ways. To this end, they will learn how to create decision trees and flowcharts, and use this information to create an application that will allow the robotic car to be navigated using voice commands. Therefore, they will learn how to use the Speech Recognition AI service and evaluate the results of its implementation. In the 3rd Activity, the students will be introduced to the idea of Learning and, by using a Machine Learning (ML) tool (the Personal Audio Classifier), they will become familiar with methods that the robotic car can learn from data and through ML. In the 4th Activity, they will be introduced to the idea of Natural Interaction, by integrating the trained model (produced in the 3rd Activity) into the created application and evaluating the results in the light of a physical implementation, thus becoming aware of the limitations of AI systems in terms of natural interaction. Finally, in the 5th Activity they will be introduced to the idea of Societal Impact, by reflecting on their entire experience and becoming aware of the advantages, disadvantaged and risks behind the use of AI and IoT in our daily lives. The ultimate goal of this project is to increase students’ confidence in all of the

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

aforementioned concepts and aspects and to create meaningful learning experiences that will help them develop a number of 21st century skills such as creativity, critical thinking, problem solving and collaboration.

1.2 The target groups

The project is aimed at students between the ages of 12 and 16 years old. Students should have some experience with block-based programming environments.

1.3 The purpose of this document

The aim of this document is to provide teachers with some concrete ideas and learning activities on how the concepts of AI and IoT can be meaningfully introduced and taught to students, through the lens of Robotics and a series of hands-on tasks.

2. Glossary of the Unit

Word	Definition
ThingSpeak	An IoT service platform to monitor data collected by the ESP8266 WiFi module
MIT App Inventor	Software for creating applications
Speech Recognizer	An AI service in MIT App Inventor that recognizes speech and turns it into a text
React App or Personal Audio Classifier	A machine learning (ML) audio classification and model training tool compatible with MIT App Inventor

3. Introducing the "DIY Robotic car that can be controlled and navigated using voice commands"

3.1 Description

This project will introduce learners to the 5 Big Ideas of AI and IoT in STEM education through the creation of a DIY robotic car (*Figure 1*), programmed to be navigated using voice commands. Specifically, learners will be encouraged to design and assemble the robotic car by using various electronic components and simple materials, and to program it using block-based programming environments. They will learn to collect data, in real-time, and to reflect on the results obtained and how they can be used to make specific

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

decisions about the performance of the robotic car. In addition, they will be encouraged to design and program an application that uses the Speech Recognition AI service to enable verbal navigation of the robotic car. Moreover, they will learn how to create a trained model to optimize the performance of the application.

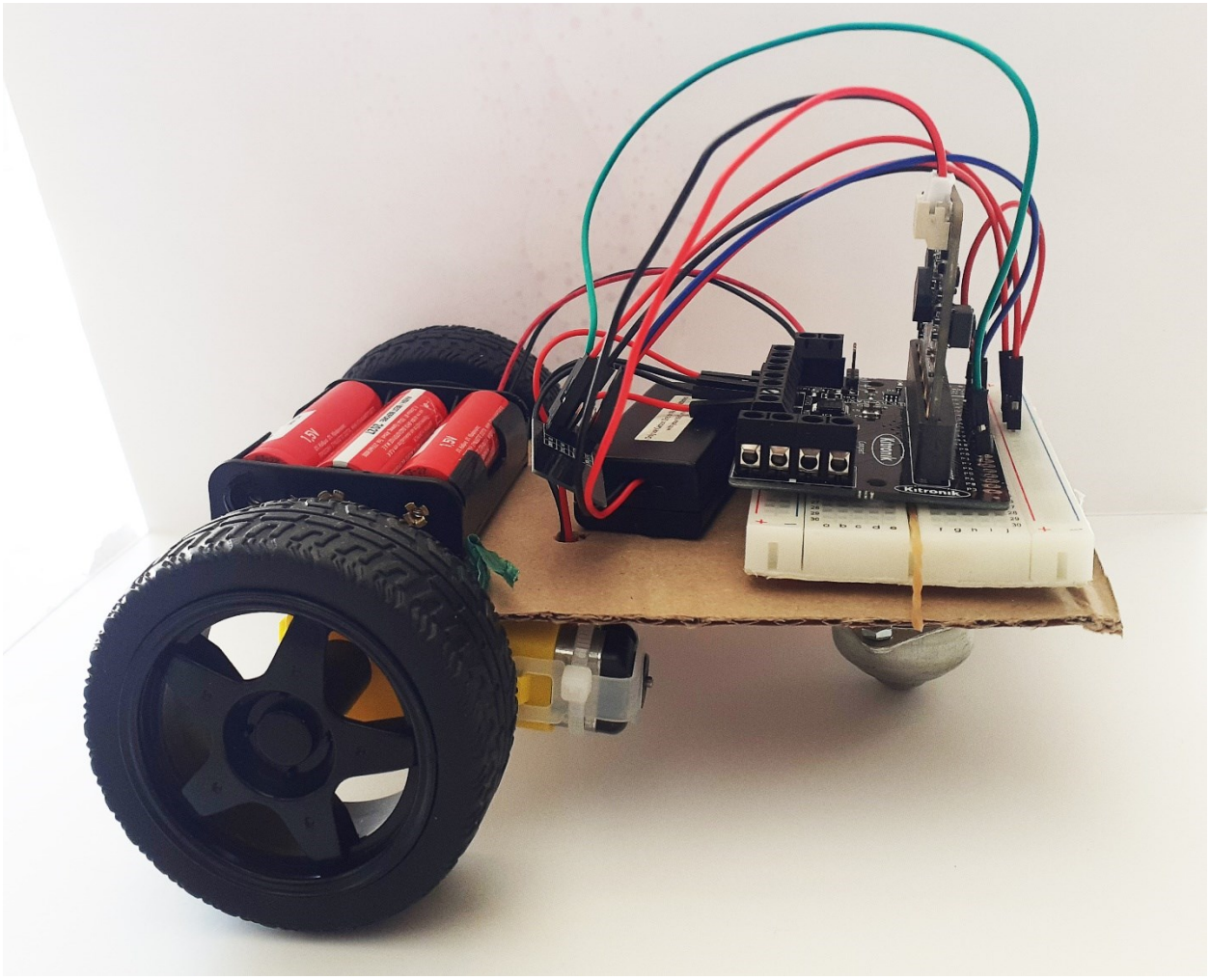


Figure 1: The DIY robotic car

To proceed with the activities contained in this document, you must first follow the instructions included in the file "*T2.4_Creating_the_robotic_car.pdf*", while you can encourage your students to carry out some warm-up programming activities, similar to those contained in file "*T2.4_WarmUp_programming_activities_for_the_robotic_car.pdf*". It is highly recommended that you divide your students into teams and encourage them to discuss the different aspects of each activity. To facilitate this process, in this document you will find links to material that can help you to introduce each step of the project smoothly, as well as some suggested questions that can help you to initiate a dialogue with your students about different parts and aspects of the project.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.2 Learning objectives & outcomes

On successful completion of this unit, learners should be able to:

- Identify the presence and use of AI robotics in everyday life
- Discuss and understand the role of AI in robotics
- Explain and discuss different aspects of integrating AI into a robotic project through speech recognition and voice commands
- Understand how AI services such as Speech Recognition work
- Understand how decision trees or flowcharts can show possible logical paths and also lead to decisions about operators that can be used at a later stage
- Discuss the role of Machine Learning (ML) in AI robotics and how ML can be used to train a robotic artefact to perceive its surroundings
- Explain the basic concepts of audio classification
- Identify and discuss the advantages and risks of implementing voice commands in driving
- Identify and discuss the advantages and risks of audio classification
- Explain basic programming constructs/concepts related to the implementation of speech-to-text methods
- Understand the key concepts underpinning IoT and the implications of monitoring data
- Reflect on the impact of data-driven decisions in everyday life
- Understand that services such as Speech Recognition are prone to error

Students will also learn to:

- Construct a robotic artefact and create circuits as part of a robotic construction
- Monitor data using an IoT analytics platform service
- Create decision trees and flowcharts to represent a type of information
- Use programming commands coupled with AI methods to address specific behavior to a robotic artefact
- Program a robot to be instructed using voice commands
- Express their ideas through programming
- Use the Personal Audio classifier ML tool to classify different sounds
- Evaluate the results produced by an ML tool
- Make improvements to a trained model based on evaluation
- Investigate factors in the training data that may lead to bias

3.3 Estimated duration of the Unit

This is a rather extensive project, needing several hours to properly address all the included aspects. The following duration is indicative and may vary depending on the age and level of your students.

Activity 1: 4 – 6 hours

Activity 2: 8 – 15 hours

Activity 3: 2 – 4 hours

Activity 4: 2 – 4 hours

Activity 5: 1 – 2 hours

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.4 Activity 1 – Introducing the Big Idea of Perception through IoT:

3.4.1 Description

This activity is a warm-up learning intervention to introduce students to the idea of Perception in the light of IoT and Robotics. They will explore how their robotic car can sense and perceive its environment by collecting and storing data, that can later be used to make some decisions about the performance of the car. In particular, they will explore how sensors built into the BBC micro:bit board, or sensors that can be connected to the board (e.g. an Ultrasonic sensor) can collect data, and be monitored using an IoT analytics platform service. The service used for the needs of this activity is ThingSpeak, a platform that allows aggregation, visualization and analysis of live data streams in the cloud.

Tip: Understanding such an environment (i.e. ThingSpeak) may be a bit challenging for your students. Therefore, depending on their level, you may decide to introduce this activity after the end of activities 2 to 4, in order to better realize how IoT can help to optimize the robotic car. If your students are under the age of 12, you can skip this activity.

3.4.2 Hardware

In addition to the BBC micro:bit, the hardware required for this activity is the ESP8266 Wi-Fi module and optionally some other sensors such as an Ultrasonic sensor (preferably the HC-SR04).

3.4.3 Setup

Before you start wiring and coding, you need to create a channel to monitor the data you receive. To do this, go to the ThingSpeak website (https://thingspeak.com/) and sign in to your account, by entering your Email (**1**.), or create an account by clicking on the “Create One” (**2.**) option (*Figure 2*).

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

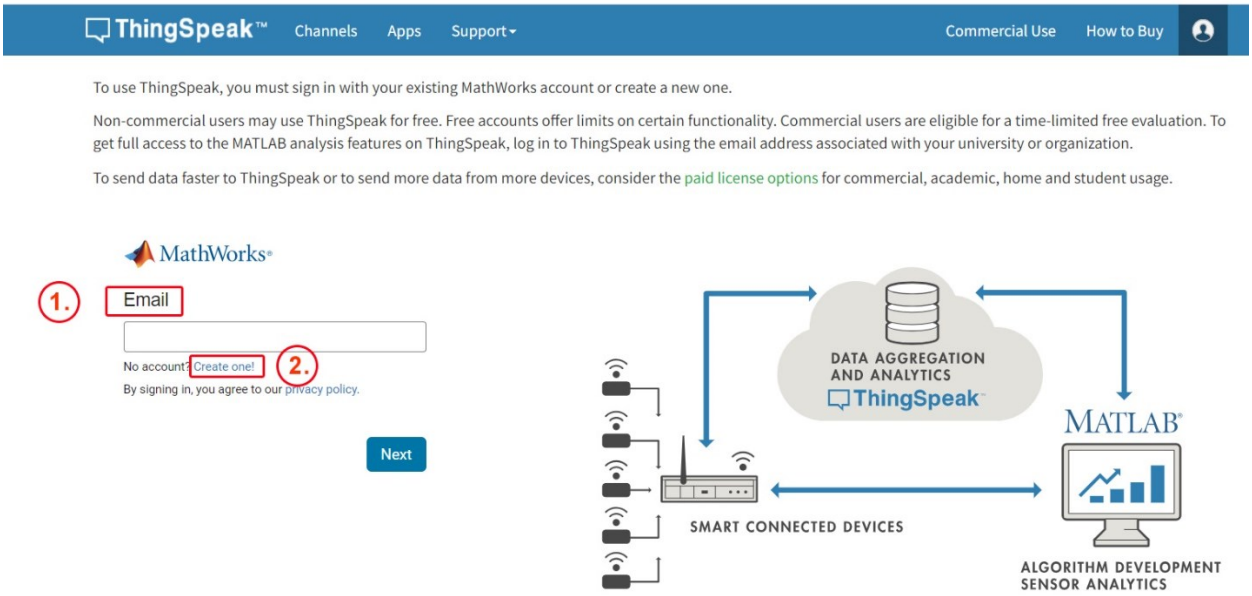


Figure 2: Menu for signing in or creating a new account to ThingSpeak

Then, go to the Channels menu and click the "**New Channel**" button under "My Channels" to create a new channel to record and display data from your sensors (*Figure 3*).

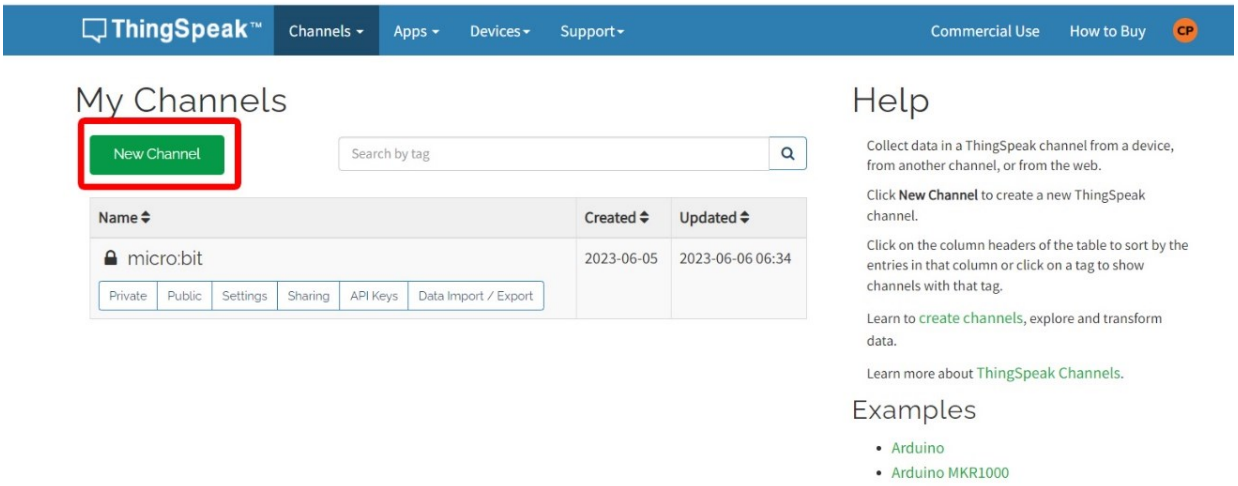


Figure 3: The New Channel button, located at the Channels menu

The next step is to create your New Channel by inserting the name (**1.**) of your channel (e.g., microbit) and the parameters that you want to monitor in the Fields provided (**2.**) (*Figure 4*). For instance, in Figure 3 the user has inserted Temperature to Field 1 and Light to Field 2. You can activate more fields by clicking on the checkbox (**3.**), next to a Field. Optionally, you can write a description to help you remember which parameters your channel is monitoring.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

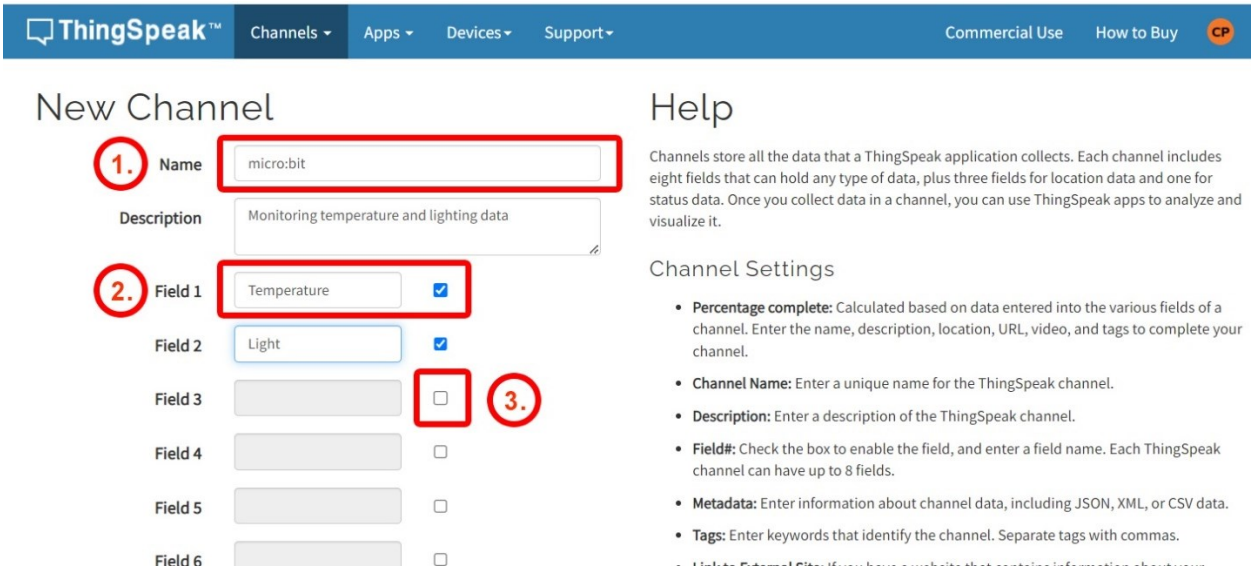


Figure 4: Creating a New Channel

Once you have declared all the parameters you want to moniotor, scroll down the New Channel page and click on Save Channel button (*Figure 5*), to save all the inserted information.

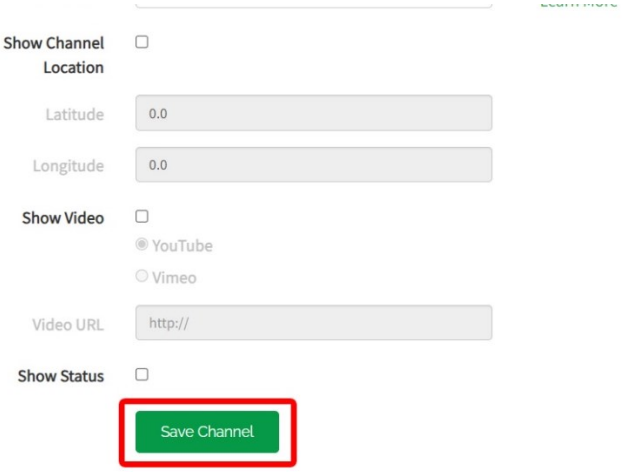


Figure 5: Scrolling down the New Channel page to click on the Save Channel button

After pressing Save Channel, you will be automatically redirected to your New Channel. Click on the API Keys menu (**1.**), and write down the Key (**2.**) that appears in the Write API Key field (*Figure 6*).

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

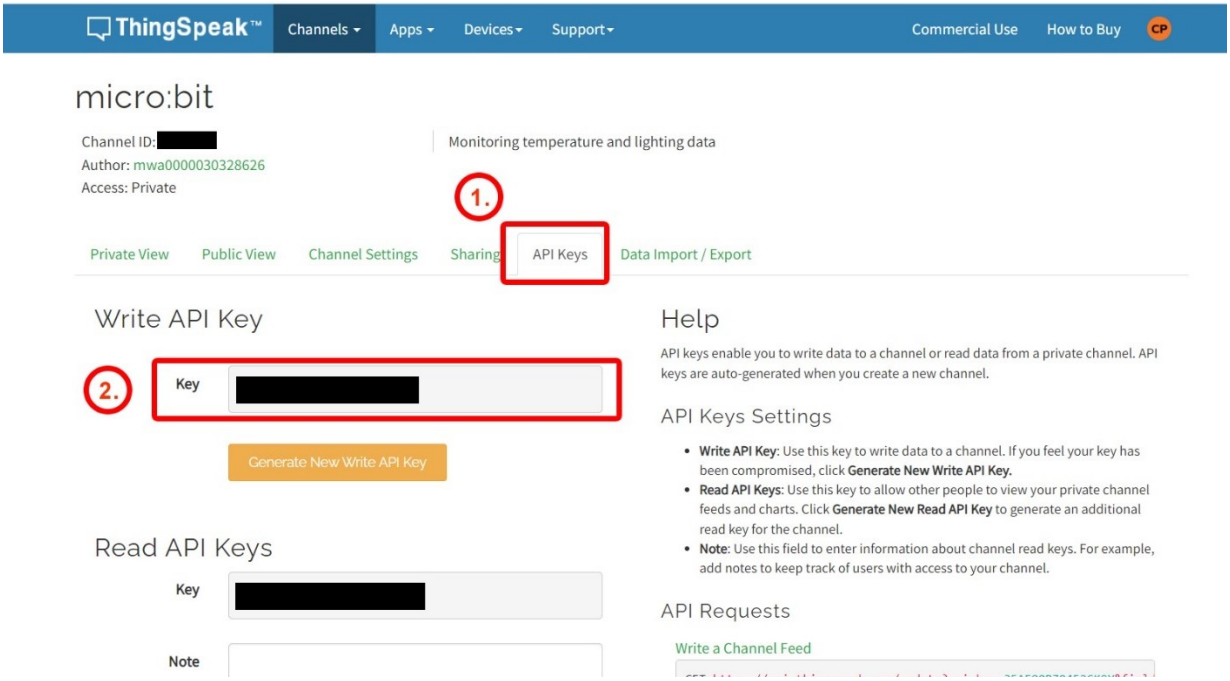


Figure 6: Finding the API Key

This Key will then be inserted into one of the command blocks that you will use in the Makecode programming environment.

Now that you have registered your channel to the ThingSpeak site, you can proceed with the wiring and programming phase of the ESP8266 Wi-Fi module.

3.4.3.1 Wiring

Figure 7 presents how to connect ESP8266 Wi-Fi module to the BBC micro:bit microcontroller. The ESP8266 module has 8 different pins. You need to connect 5 of them to the micro:bit board. In particular, you need to connect the 3V3 and EN pins (**1**) to the power (3V), and the GND pin (**4**) to the GND pin of the micro:bit. Finally, connect the TX pin (**2**) and the RX pin (**3**) to pins P1 and P0 respectively. Once the circuit is complete, connect the microbit to your computer by using a USB cable.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

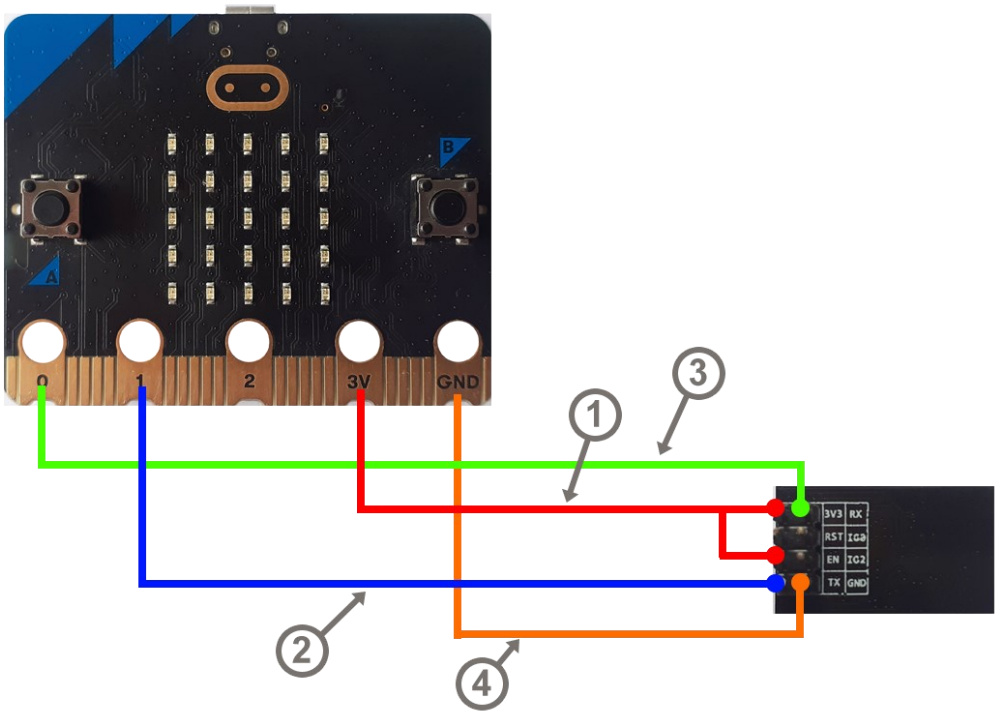


Figure 7: Connecting ESP8266 WiFi module to the BBC micro:bit microcontroller

3.4.3.2 Code

The next step is to open the Microsoft Makecode block-based programming environment (https://makecode.microbit.org/#) and create a New Project. On the Makecode home page, click on the New Project tab (**1.**), and on the pop-up menu enter a name for your project (e.g. ThingSpeaK_WarmUp). Then, click on the Create (**2.**) button (*Figure 8*).

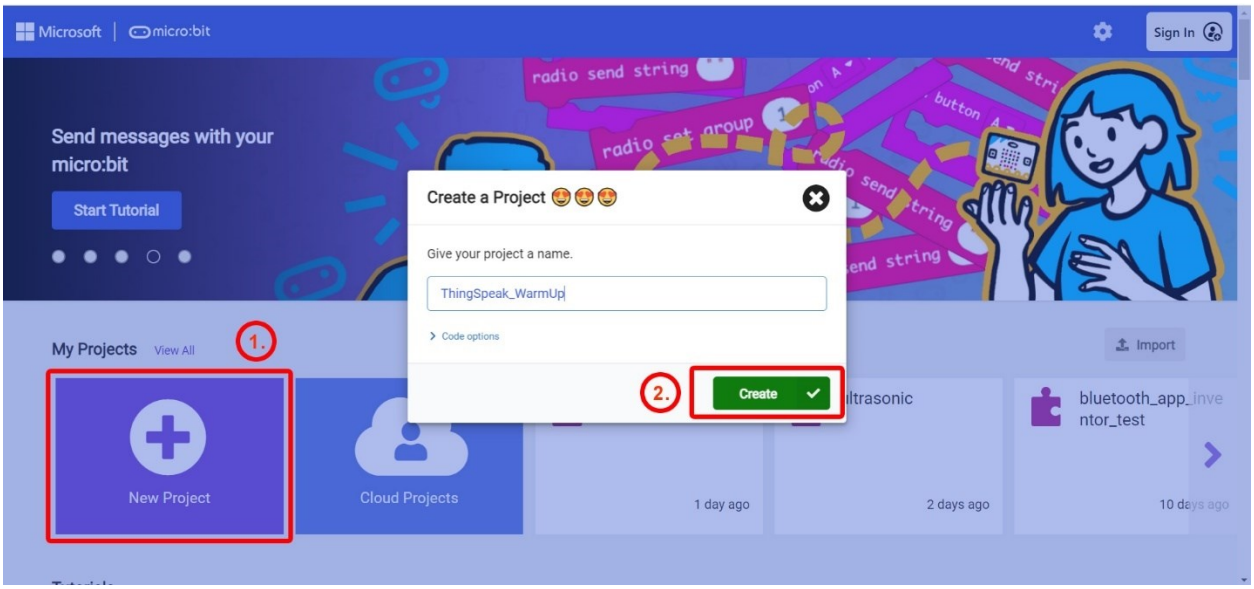


Figure 8: Creating a new project in the MakeCode programming environment

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

You will be automatically redirected to the area where you can assemble your code. To include the blocks for programming the ESP8266 Wi-Fi module, click on the +Extensions menu (*Figure 9*).

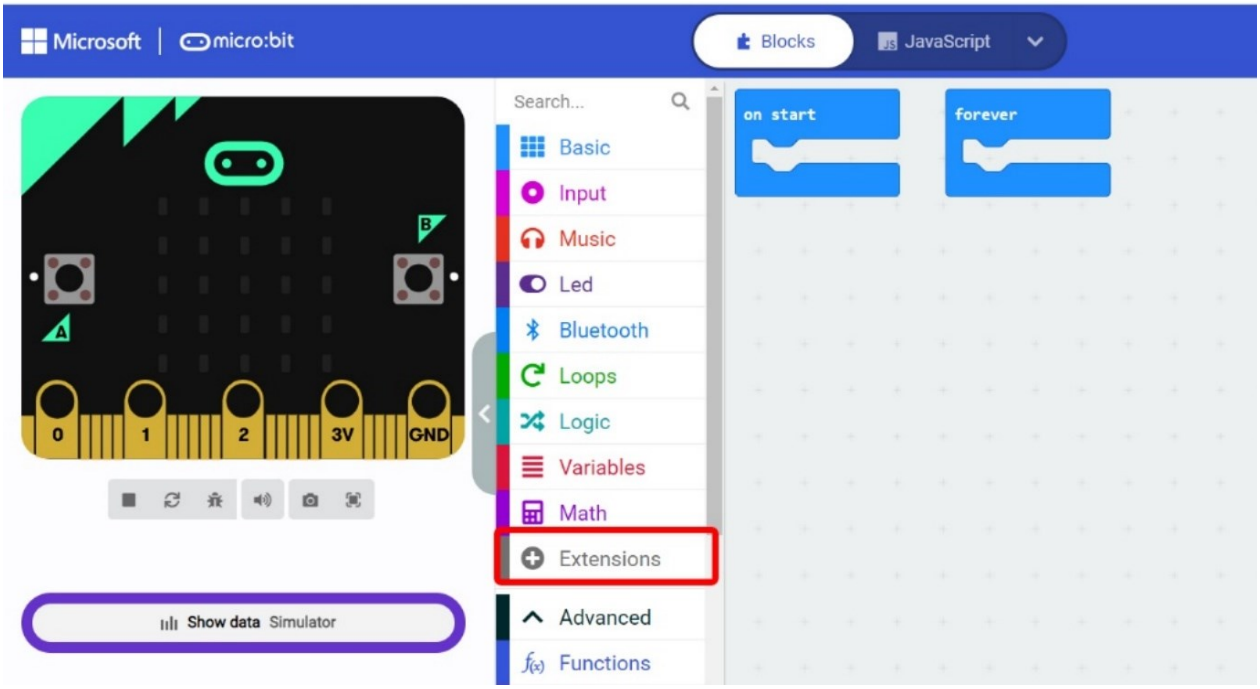


Figure 9: The +Extensions menu

Type "ThingSpeak" in the "Search or enter project URL" field (**1.**) and select the corresponding extension (**2.**) (*Figure 10*).

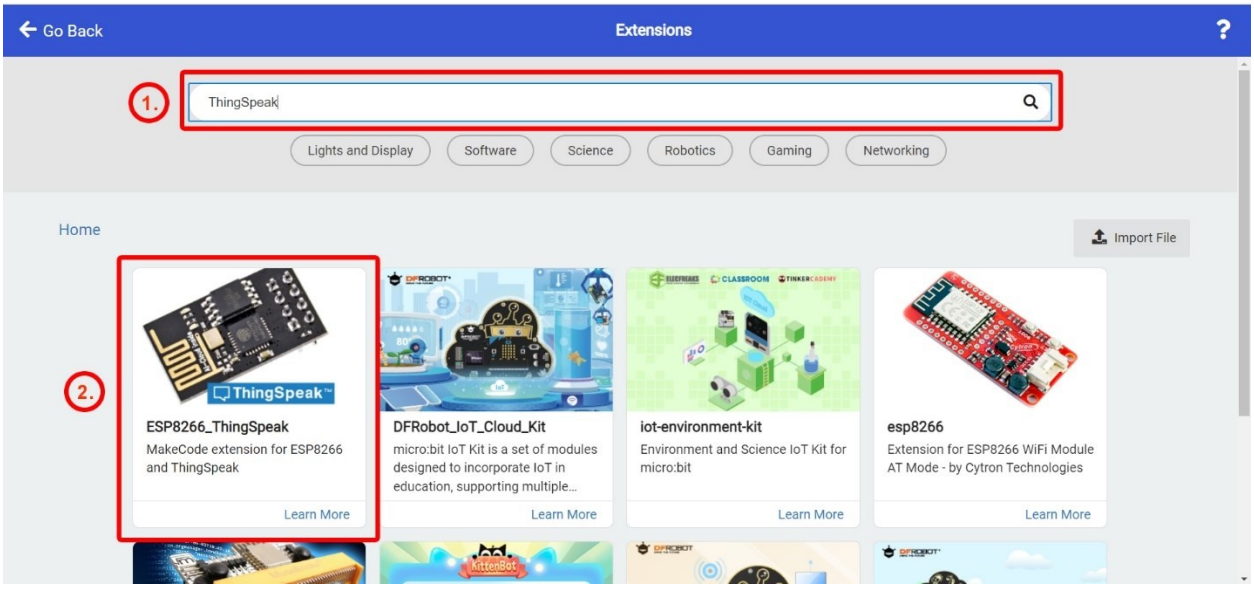


Figure 10: Finding and selecting the ESP8266_ThingSpeak extension

A new menu of blocks (i.e., ESP8266 ThingSpeak) will appear in the command menu (*Figure 11*).

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

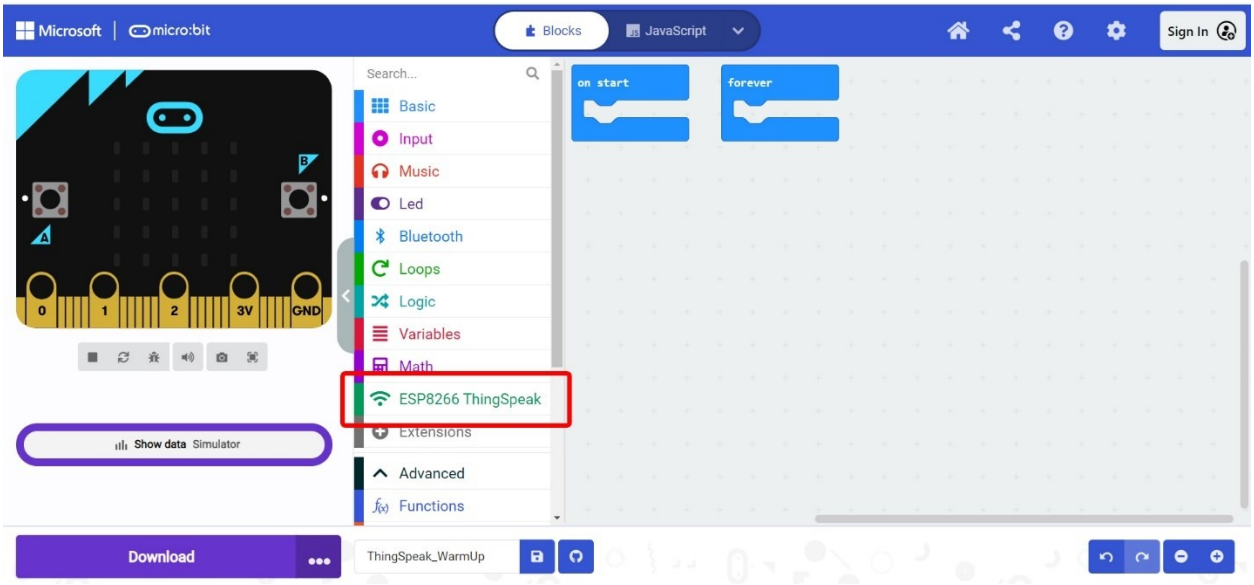


Figure 11: The new menu with block of commands for programming the ESP8266 WiFi module

First, you need to initialize the Wi-Fi module by declaring the pins it is connected to, and inserting information from the Wi-Fi router you are going to connect to. To do this, click on the ESP8266 ThingSpeak menu, and select the **initialize ESP8266** block command from the floating menu (*Figure 12*).

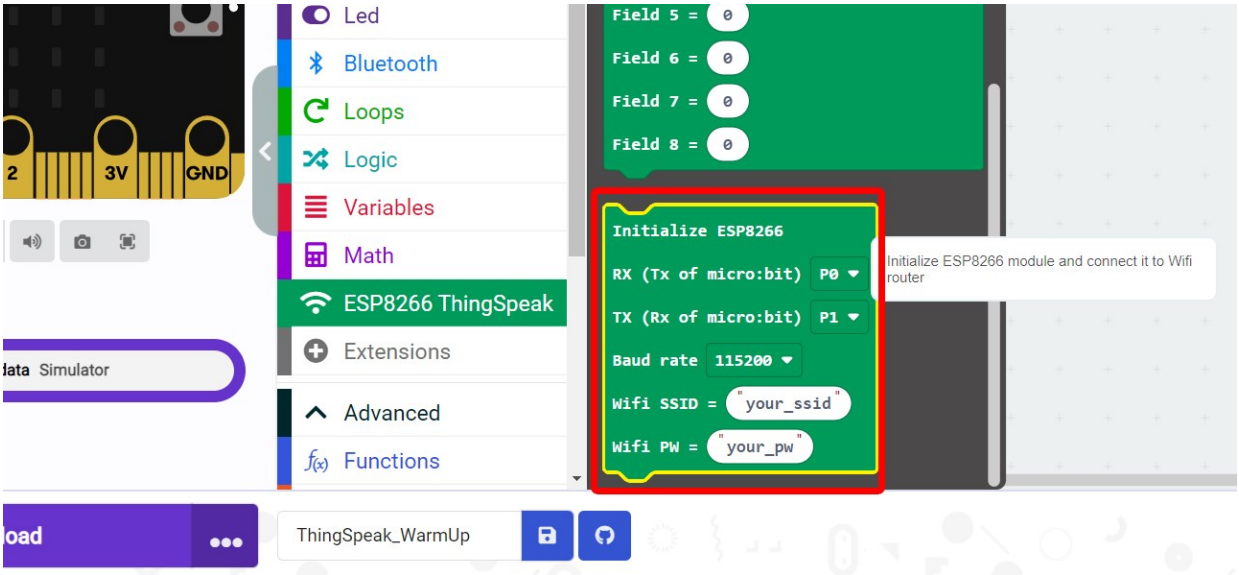


Figure 12: Finding and selecting the Initialize ESP8266 block command

Drag and snap this command to the "**on start**" block. In the RX and TX fields (**1.**) specify the ports to which you have connected the corresponding pins of the module (namely P0 and P1) (*Figure 13*). Then, in the wifi SSID and wifiPW fields (**2.**) enter the name of your Wi-Fi router and its password. This will connect the Wi-Fi module to your Wi-Fi router.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

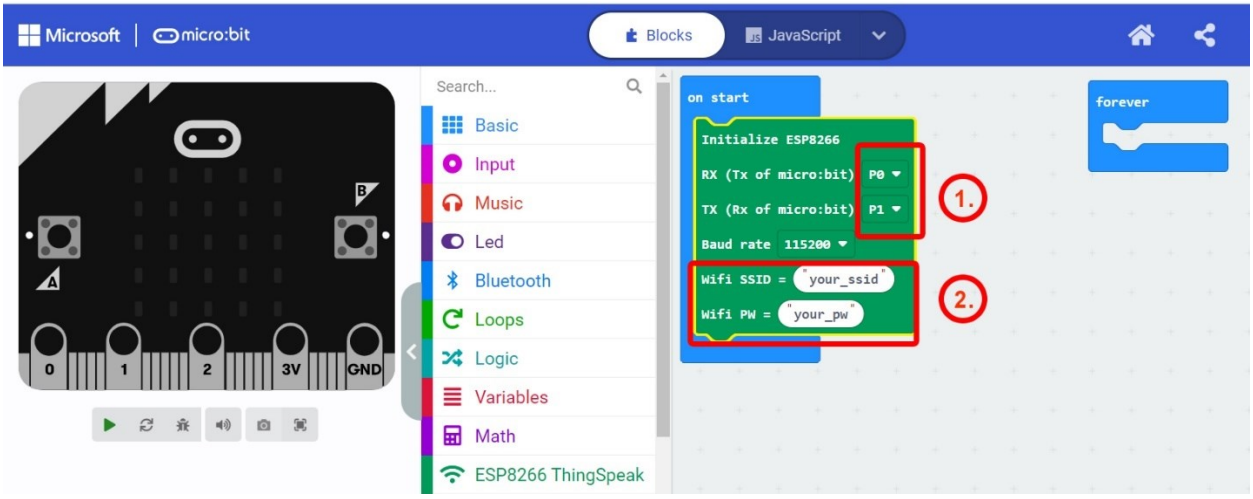


Figure 13: Declaring the pins that ESP8266 is connected and inserting information about wifi connection

The next step is to enable the Wi-Fi module to upload the data it receives from microbit, to the ThingSpeak platform. To do this, drag and snap the "**upload data to ThingSpeak**" block command from the ESP8266 TingSpeak menu, to the "**forever**" block. In the Write API key field (**1.**) (*Figure 14*), type the Write API key that you received after creating your channel on the ThingSpeak platform (*Figure 6*). Next, to declare the data that will be monitored by the ThingSpeak platform (i.e., temperature and light), go to the "**Input**" menu, and drag and drop the "**temperature**" (**3.**) and the "**light level**" (**2.**) input commands to Field 1 and Field 2, respectively (*Figure 14*).

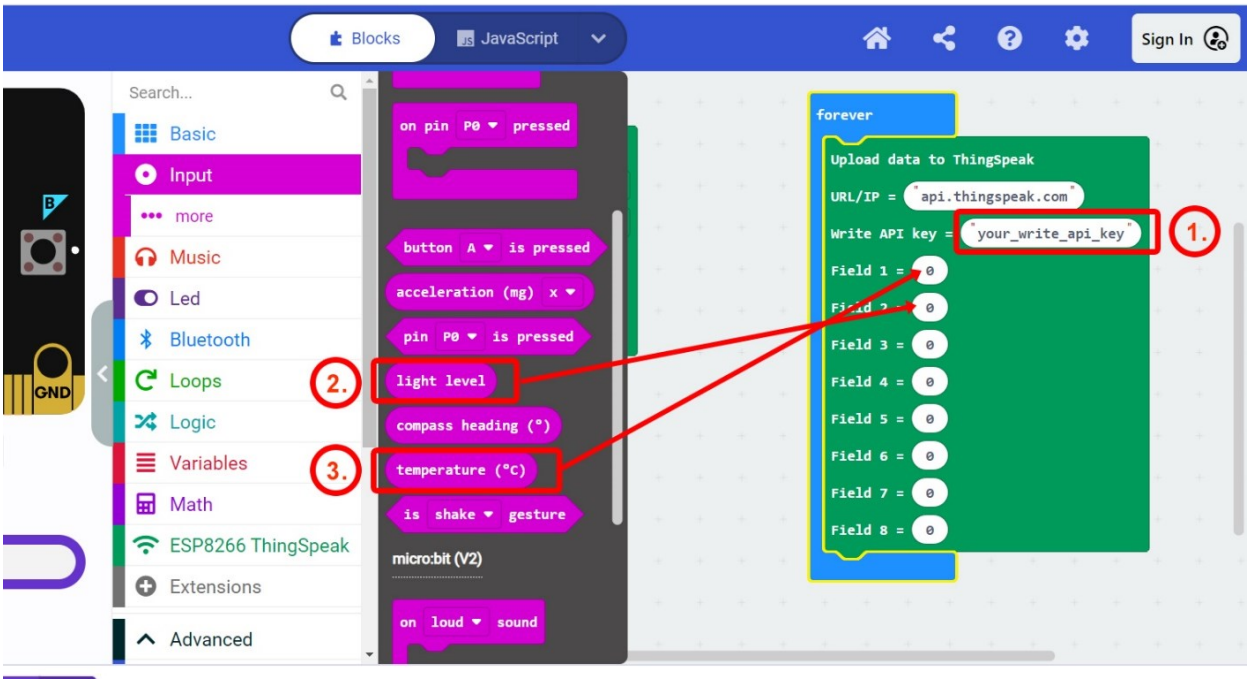


Figure 14: Inserting the API key, and adding the parameters that will be monitored

After that you can download the script to you micro:bit by clicking on the download button.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

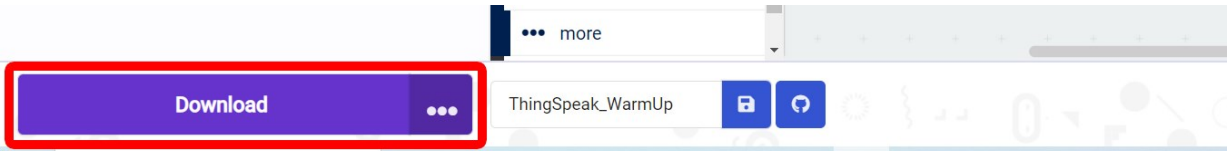


Figure 15: The Download button

Note: Keep in mind that you may need to add a pause command of 15 seconds within the forever loop command, to allow for a smooth transfer of data to the ThingSpeak platform.

Optionally, you can add the script shown in Figure 16 to your code to display some of the received data on your microbit's LED screen. These commands (i.e., "**serial on data received…**" "**serial read string**") are located in the **Serial** command menu.

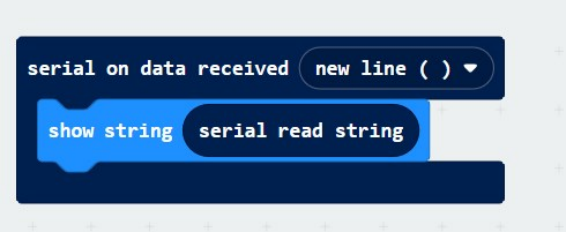


Figure 16: Displaying the received data on microbit's LED screen

After downloading the code, go to the ThingSpeak platform and use the Private View tab to see how the data received by microbit changes over time (*Figure 17*).

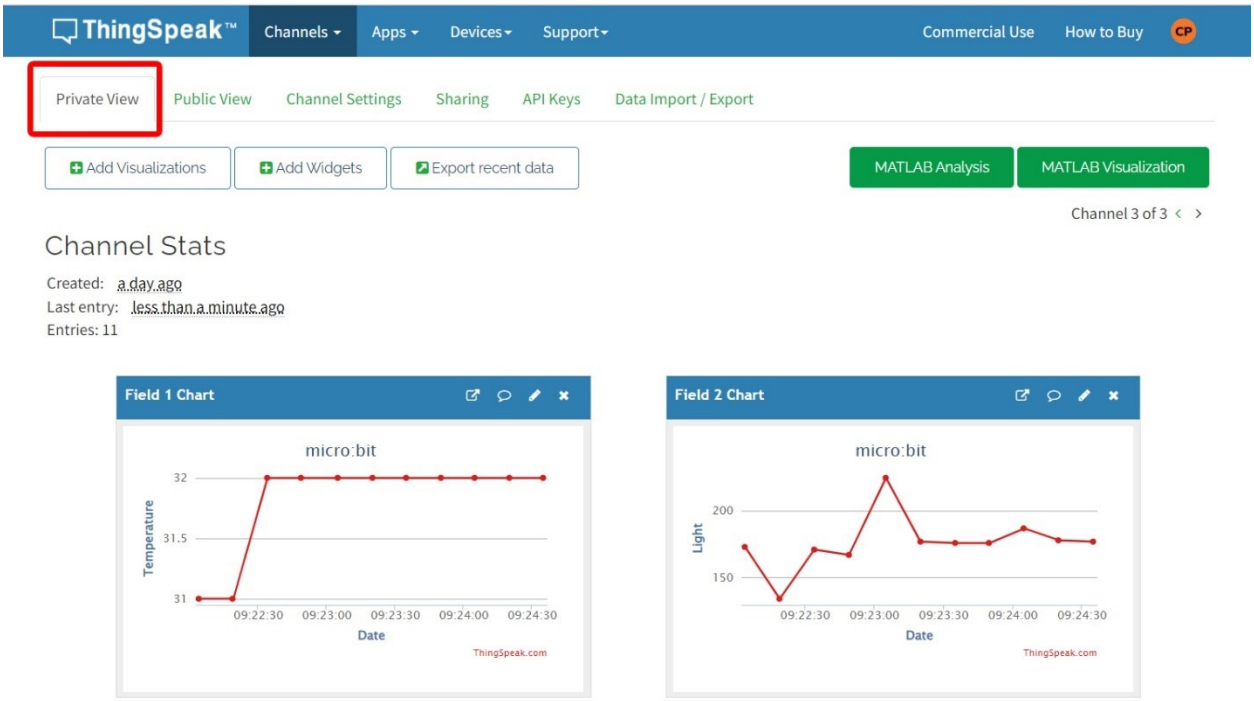


Figure 17: Checking how the received data changing over time

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.4.4 Experiment 1

Repeat the aforementioned process with your students. Before this, create some accounts on the ThingSpeak platform and share them with your students. Divide them into teams of 2 or 3 and give each team a micro:bit board and an ESP8266 Wi-Fi module. Encourage your students to look at the Wi-Fi module to explore the built-in pins, and then ask each team to connect the module to the micro:bit board. To facilitate this process, you can give them the circuit map contained in the file "*Circuit_card_Arctivity1.pdf*".

Go through all the teams to make sure they have all successfully created the circuit. Then proceed with the programming process. Give each team one of the accounts you created on ThingSpeak platform, and ask them to log in. Encourage them to look at the different tabs (i.e., Private View, API Keys etc.). Then give each team a copy of the "*Half_baked_Activity1.pdf*" file and ask them to create the script to program the Wi-Fi module to monitor the temperature and the light level recorded by the micro:bit. Ask them to download the script to their micro:bit and advise them to observe the data being monitored on the ThingSpeak platform. Encourage them to use a flashlight or/and move the circuit closer to colder or hotter surfaces to observe how the monitored data changes dynamically.

Then discuss with them some of the following issues:

- What information is being extracted?
- Does this information affect the performance of the robotic car?
- Can this information lead to some data-driven decisions to optimize the performance of the robotic car?
- Think of other parameters that could be recorded. What kind of sensors do you need to connect to the robotic car to record these parameters? Can all these parameters lead to significant data-driven decisions for the performance of the robotic car?
- Think about AI services that could benefit from such data. How can these services use these parameters to extract the information behind a dataset, thus optimize the performance of the robotic car?

Through this activity and the following discussion, students will understand:

- How sensors can be used not only to sense an environment, but also to make some data-driven decisions.
- That the perception of a robotic artefact is based on collected data
- The importance of monitoring and evaluating data to make optimal decisions for the performance of an autonomous vehicle
- The monitored data can be used by AI services to optimize their performance.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.5 Activity 2: Introducing the idea of Representation and Reasoning

3.5.1 Description

This activity aims to introduce students to the 2nd Big Idea, namely the idea of Representation and Reasoning, and to familiarize them with the way in which a robotic artefact can "learn" from data. During this activity, the students should keep in mind that AI and intelligent agents can "think": **a)** by constructing representations of the world in the form of data structures, and **b)** by using algorithms to help them to make sense of these data structures (reasoning). Therefore, students will learn how data structures can be represented, and how algorithms can be used to extract specific information from the representations. To this end, they will be encouraged to create some decision trees. Through this process, they will understand the criteria for selecting an algorithm that will lead to the best possible solution.

In particular, students will learn how to program an application that allows the user to navigate the robotic car using voice commands. To do this, they will use the Speech Recognition AI service and be introduced to the App Inventor environment. Before developing the application, they will create a decision tree to represent the decision that their robotic car should make when a voice command is received. They will then decide which logical path to follow when programming their application.

3.5.2 Creating a decision tree

To start creating a decision tree, you need to think about how the AI service will be implemented and the possible actions that can be activated. The main goal is to create an application with a button that records voice commands when pressed. Based on this, you can start your tree by assuming that the command received contains the word "*forward*". If so, the robotic car should be instructed to move forward. If it does not contain "forward", then there are at least two different logical paths: **a)** The command is not recognized at all, so the robotic car will not move (or it will continue to execute the previous command), **b)** the received command contains another directional keyword, so the robotic car will be instructed accordingly. Figure 18 shows an indicative example of such a decision tree. The tree is not complete. More branches can be added based on the different commands received.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

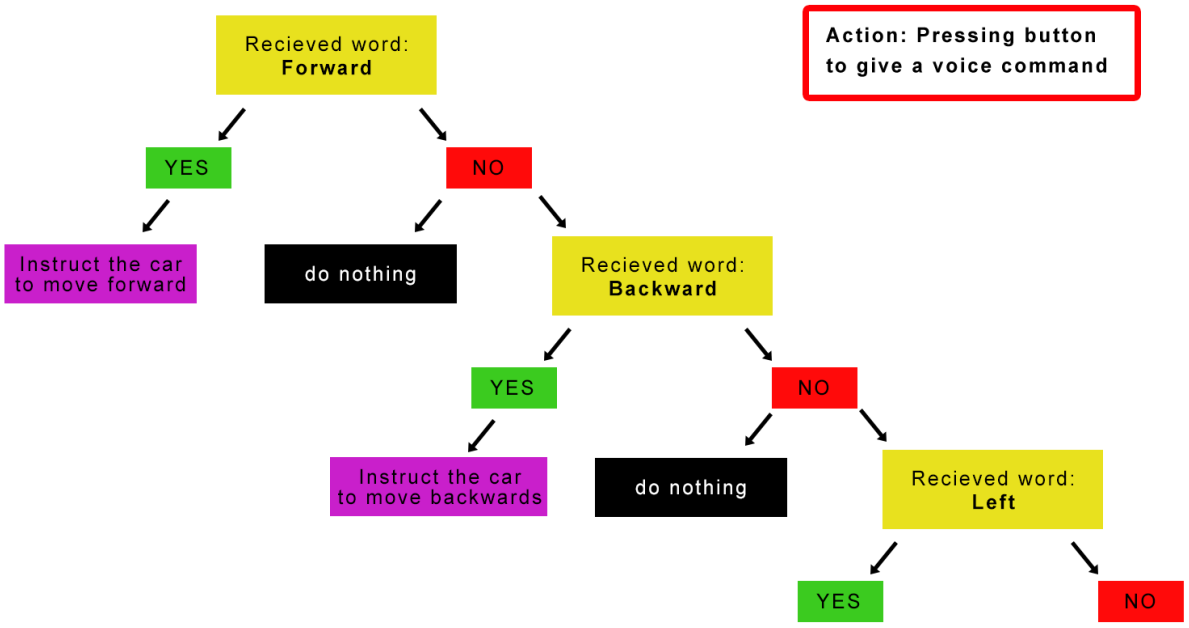


Figure 18: An indicative decision tree depicting the possible actions that can be activated when Press to Speak button is pressed

Based on the decision tree, the students can decide which logical paths should be used when programming. The decision tree shown in Figure 18 suggests that a series of "if…then" commands should be used to successfully program the application.

3.5.3 Designing and programming the application

After analyzing the possible actions that can be activated, the next step is to create the application that will use the Speech recognizer AI service to allow the user to control the robotic car using voice commands. Before starting to create the application, it is highly recommended that you carry out with your students the warm-up activity contained in the file "*2.4_App_Inventor_Warm_Up.pdf*", and continue working on the same "*.aia*" project. Alternatively, you can open the "*Remote_control_Microbit.aia*" project file and work on it. In that case, you are highly advised to save the project under a new name.

Note: If you are using the aforementioned .aia file, instead of doing the App Inventor warm-up activity, you must also remember to advise your students to download the script described in the "*T2.4_Programming_the_robotic_car.pdf*" file to the robotic car.

Design process

First, you need to add a new button that will activate the Speech Recognition AI service. Drag and drop a Button item under the navigational buttons (**1**), rename it to btn_speak (**2**), while changing its text to "Press to speak" (**3**) (*Figure 19*).

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

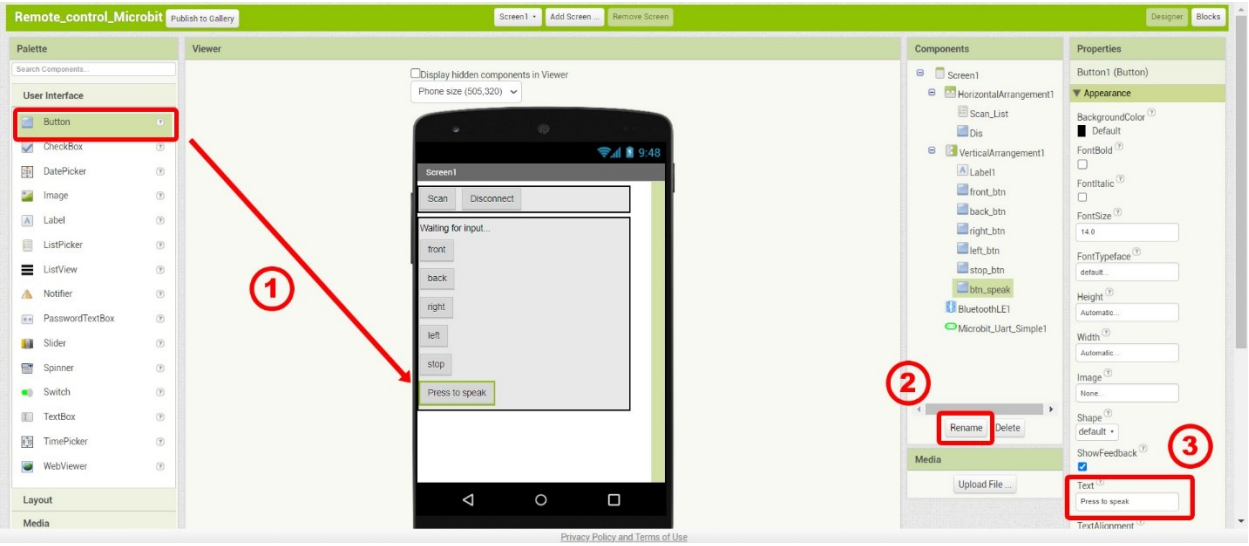


Figure 19: Adding a new button (1) and renaming to btn_speak (2), while changing the text to Press to speak (3)

When the "**Press to speak"** button is pressed, the application will start to detect your voice. Then, by default, it will use the speech-to-text AI service to convert the received voice commands into text.

To use this service, you need to add the "SpeechRecogniser" component. Find it in the "Media" tab, and drag and drop it on the screen (*Figure 20*).

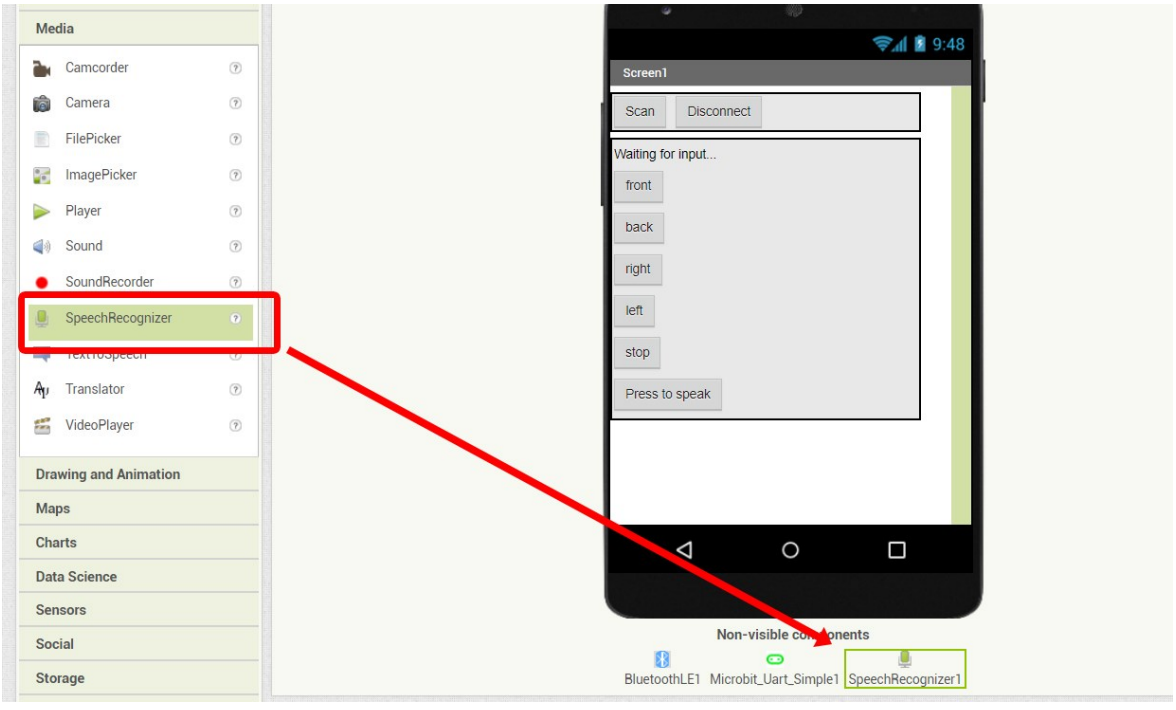


Figure 20: Dragging and dropping the SpeechRecognizer component to the design area

Co-funded by the European Union

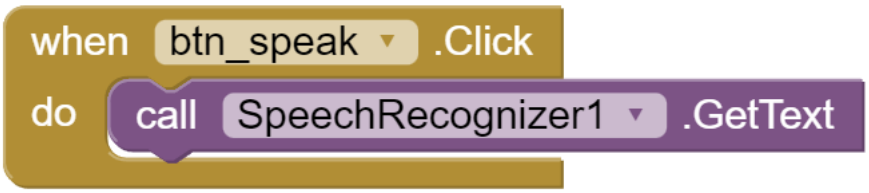
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Note: Similar to the BluetoothLE and Microbit_Uart_Simple components, the SpeechRecognizer is a non-visible component and therefore does not appear on the screen, but in the area below the screen preview.

Programming process

This programming solution will instruct your application to recognize a voice command, each time the "Press to Speak" button is pressed.

Go to the Block menu and select the "btn_speak" component. From the floating menu select the **"when btn_speak.Click do"** event handler and drag it in the area for assembling the script. Then, click on the Speech Recognizer component and from the floating menu select the **"call SpeechRecognizer.GetText"** command, and place it inside the handler event.



This block programs the button btn_speak to **"get the text"** that SpeechRecognizer detects, each time the button "Press to speak" is pressed. Concurrently, the speech-to-text AI service is activated and the user's voice is converted into text, enabling the robotic car to perform the instructed movement (i.e., move forward if the received voice command is "front" etc.)

Note: the microphone on the user's smart device is used for voice capture and recording.

Reacting to voice commands

Through the above programming method – and by implementing the speech-to-text AI service - we have found a way to store a voice command as a text. This text is used as input to command the robotic car to perform different movements. To do this, the incoming voice commands are filtered and searched for specific keywords that apply to our case, namely front, back etc. In this way, the application can correspond to a number of different voice commands. The following table briefly explains this concept:

If the sentence received by the Speech Recognizer contains the word:	***Then, send the message:***	***And, the automobile will:***

Co-funded by the European Union

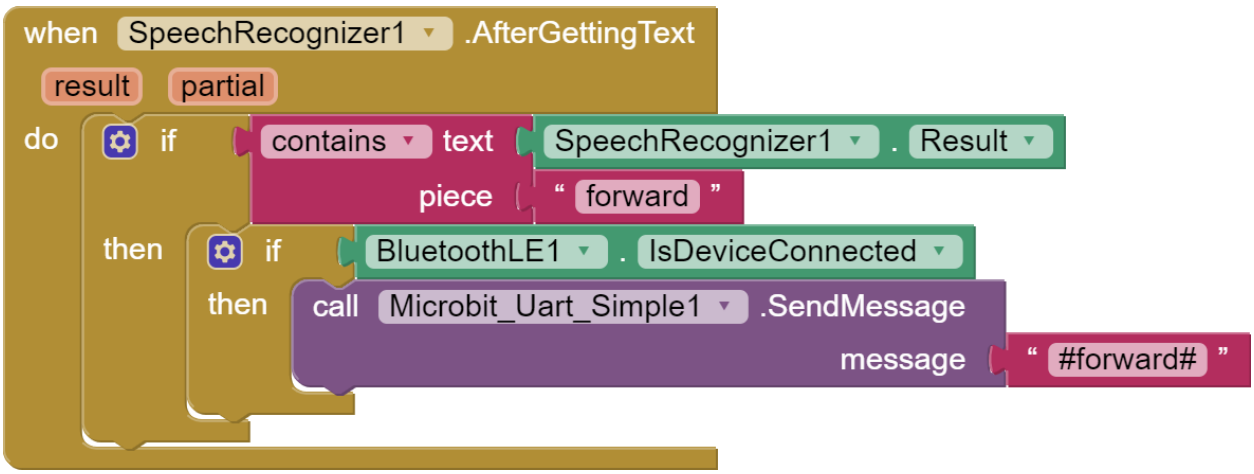
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

“forward”	#forward#	move forward
“backwards”	#backwards#	move backwards
“left”	#left#	turn left
“right”	#right#	turn right
“stop”	#stop#	stop

Sentences similar to the following will have the same effect on the car’s movement:

Forward: “Move **forward**”, “Go **forward**, please”, “I want you to move **forward**”, etc...

The following script instructs the application to send the message #front# to the robotic car, when a recorded voice command contains the word “forward”. When the message #front# is received by the robotic car then (based on the code already uploaded to the robotic car from the Makecode environment), the robotic car moves forward.



In the following, the functionality of each block of commands is explained in more detail:

When_SpeechRecogniser.AfterGettingText: This block receives the text result after the recorded voice command has been converted to text. So, when the user’s voice command is converted to text by the SpeechRecogniser, this block will **do**/execute the following blocks of commands:

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

If the **SpeechRecogniser.Result,** namely the text that is returned as the result of the SpeechRecogniser process **contains_text (), piece "forward"** (i.e., the received text contains the word "forward")

Then if BluetoothLE1 is **connected** to the **desired** device, the **Microbit_Uart_Simple1.SendMessage** is **call**ed, and sends the message **#forward#** (**text "forward"**) to the robotic car. As a result, the robotic car is moving forward.

To set it in line with the produced decision tree:

"When the SpeechRecogniser has converted the user's voice command to **text,** check **if** the **SpeechRecogniser's result** (namely the text) **contains** the word **"forward". If** it does**, then send** the message **"front"** through the **Microbit_Uart_Simple".**

The rest of the voice commands can be recognized in the same way. Therefore, the above script should be expanded by adding four more **"if...then"** conditions, and the corresponding **text pieces** and **text** messages, that will instruct our robotic car to execute a specific movement (i.e., **backwards** and message **"#backwards#"** for moving backwards, **right** and message **"#right#"** for turning on the right etc.).

The entire script is shown in the following image (*Figure 21*):

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

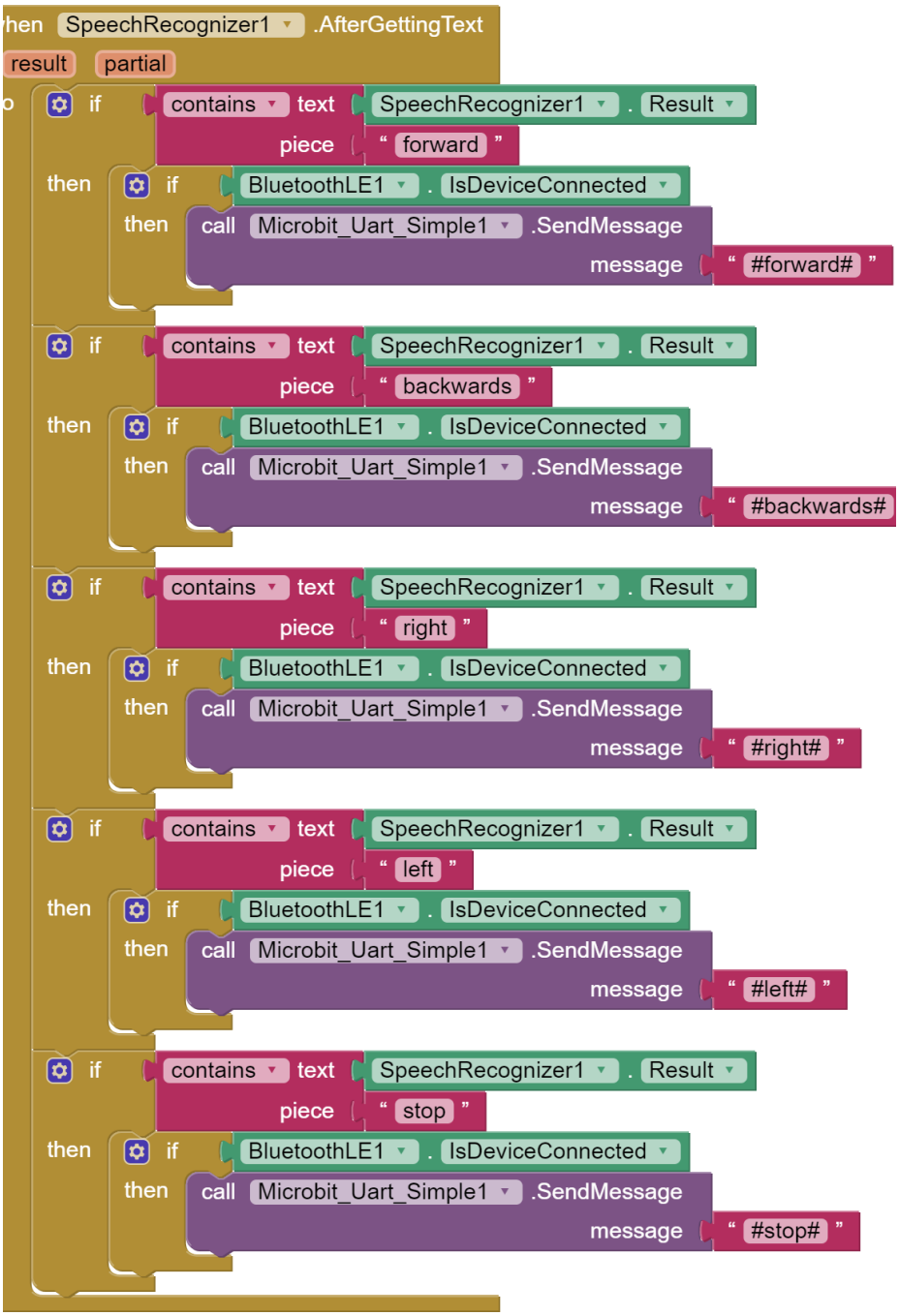


Figure 21: The entire script for SpeechRecognizer component after getting text

Limitations – towards an optimal programming solution

The above programming solution has one major drawback; it requires from the user to press the "Press to speak" button every time they want to give a voice command. This makes the whole process less intuitive in terms of driving habits and may also reduce the initial enthusiasm of your students, as they may find this solution very similar to the remote-control process (i.e., always pressing a button to give a single voice command).

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

In order to overcome this disadvantage, an optimal programming solution is also proposed, in which the user only has to press the "Press to speak" button once at the beginning of the process and then the application will try to hear the user's voice every 2 seconds.

Note: This is a slightly more advanced solution, so depending on the level of your students, you may skip this part, and end this learning activity here.

Adding a Clock sensor

In this solution, we will add a Clock sensor that will set a countdown timer to the application. When the "Press to speak" button is pressed, the timer is fired/ignited, and the countdown starts for a specified time interval. During this time interval, the application listens for an incoming voice command. If the application recognizes a valid voice command, then it will instruct the robotic car to perform the corresponding movement. Otherwise, it will continue to search until it hears a valid command. When the countdown ends, the timer is automatically re-set and a new countdown -for the same time interval- begins.

Therefore, the user only needs to press the "Press to speak" button once to initialize the whole process, bearing in mind that the application can receive only one valid voice command between time intervals.

Here is how this solution can be implemented.

First, you need to add the Clock sensor to the application. In the designer menu, go to the "Sensors" tab and drag and drop the "Clock" sensor to the screen (*Figure 22*).

Note: The Clock is also a non-visible item, so it does not appear in the screen's preview

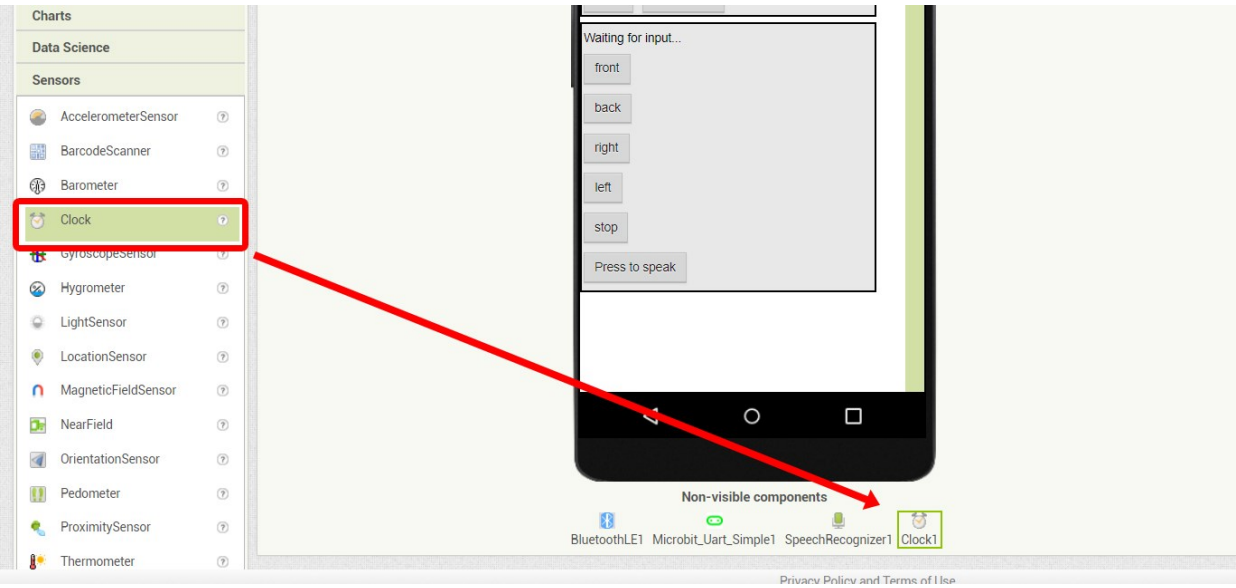


Figure 22: Adding the Clock item to the application

Select the Clock component, on the components list, and on the Properties menu, set the following parameters: a) tick the box under the TimerAlwaysFires to enable the Clock's Timer to fire every time the

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

countdown ends, b) in the TimerInterval field set the duration of the timer, by manually inserting a numeric value in milliseconds (i.e., 2000 in the example) (*Figure 23*).

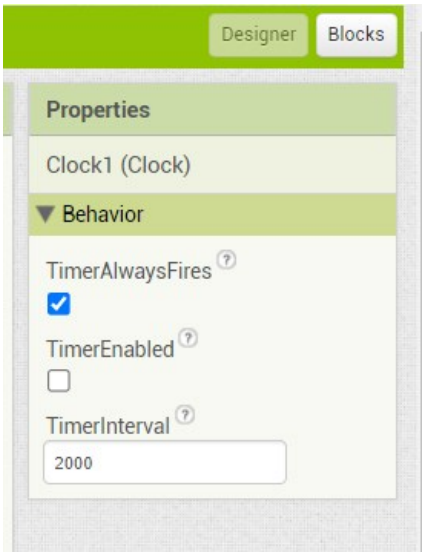


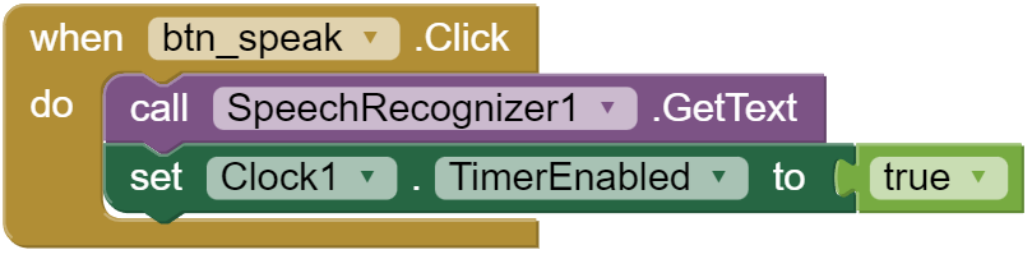
Figure 23: Setting the properties of the clock component

Note 1: the duration of an interval is measured in milliseconds. 1 second is equal to 1000 milliseconds. Therefore, 2000 ms are equal to 2 seconds.

Note 2: the box under the TimerEnabled is not checked, because we do not want the Timer to be enabled when our application is initialized. We want it to be enabled when the "Press to speak" button is pressed.

The next step is to program this new component.

On the Blocks menu, add the "**set Clock1.TimerEnabled to**" command inside the **"When btn_speak.Click"** event handler, and under the **"call SpeechRecognizer.GetText"** command. Then snap a **"true"** condition on the right side of the command. Through this command the Clock's timer is enabled after the initial activation of the SpeechRecognizer.



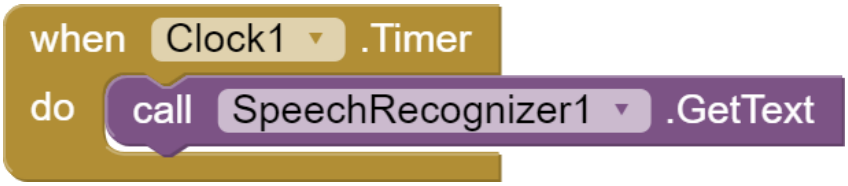
Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Now that we have fired/ignited the timer for the first time (after pressing the "Speak" button), the next step is to re-activate the SpeechRecognizer each time the countdown ends. The timer's duration is automatically renewed through this process.

To do this, select the Clock component and from the floating menu select the **"When Clock1.Timer"** event handler. Then, place the **"call SpeechRecognizer1.GetText"** command inside the handler.

Through this script, we instruct our application to **"call the SpeechRecogniser to get the text"** of what it hears, every time the Clock's Timer fires.



Explanation of the entire code: When the user clicks the "speak" button for the first time, the SpeechRecogniser is called and then the Clock's Timer is enabled. When the Timer is enabled, it will fire every 2 seconds. Every time it fires, the SpeechRecogniser is called, waiting for receiving an audible/verbal input (in our case a voice command).

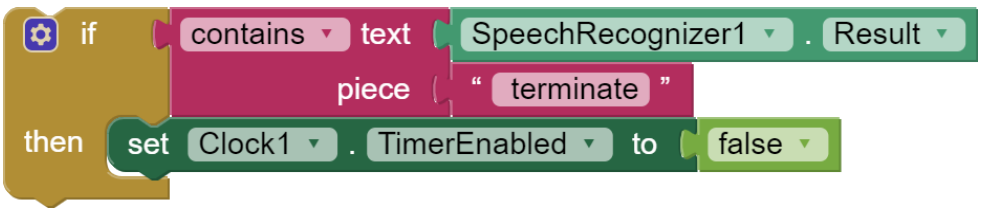
So, you have created an application which enables the user to click the "Press to speak" button only once, at the beginning, and then automatically calls the SpeechRecogniser every 2 seconds to check for new voice commands.

Adding an extra command to terminate the application

By adding the Clock component, you have created a more intuitive application. However, the application will run forever. This is because the Clock's Timer fires forever, and every 2 seconds, creating an infinite loop, in which the SpeechRecogniser is constantly looking for valid voice commands.

To solve this problem, you need to add some extra blocks of code in the application. Specifically, you need to add to the script in Figure 20, the following block of commands.

In particular, add one more **"if...then"** condition at the end of the script. Inside the **if** part, place a block of commands that will enable **SpeechRecogniser** to identify the **text piece "terminate"**. Inside the **then** part, place a block of command that will disable the Clock sensor, namely the **"set Clock1.TimeEnabled to" "false"** blocks.



Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Through these additional blocks of code, the Clock's Timer will be disabled when the user says the word terminate. Thus, the loop will be terminated and the SpeechRecognizer will stop looking for valid voice commands.

Tip: Depending on your students' level, you can encourage them to find their own solution to the inifinite loop problem. In this way they can better realize how the Clock sensor works, and feel more confident about the whole process.

3.5.4 Experiment 2

This 2nd activity is quite extensive, so you need to foresee a couple of hours to deal with all the concepts and their inherent aspects.

To introduce the second idea, apart from encouraging your students to design decision trees or flowcharts to represent possible actions or behaviors that the robotic car can adopt, you can also initiate a dialogue by asking some of the following **questions**:

- What kind of representations do you need to create to better illustrate the encoded information?
- What logical paths or operators do you need to adopt to help an intelligent agent to take the best possible action?
- What is the best way to use the results of the AI's perception? (e.g. based on the robotic car's reaction, think of cases where such technology would be of added value)

Through this activity and the discussion that follows, students will be able to understand:

- How decision trees and flowcharts can reveal possible logical paths, leading also to decisions about which operators to use
- Understand how an intelligent agent can adopt the best possible behavior or performance
- Comprehend how intelligent agents can perceive their environment and act on incoming information

For the design and creation of the application you can use the file "Students_Worksheet_for_Activity_2.pdf" and discreetly guide your students through the process, giving them tips where necessary. You can also encourage them to experiment with different voice commands (and therefore **text pieces**) and observe the results. To this end, you can advise them to make a table – such as the one below – in which they will record which voice commands are successful and which are not in moving the robotic car in a given direction.

Voice command	**Success**	**Failure**
Move forward	√	
Go to front		√

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

….		

For the optimized solution (with the addition of the Clock item), you can encourage your students to set different durations for Timer Interval and record their observation on the functionality of the application. You can also suggest them to create a table (such as the one below) to organize their observations. Moreover, you can encourage each team to use a different time interval and compare the results in parallel demonstrations.

TimerInterval	**Observation**
500ms	The application is working very fast
2000ms	The application is properly working
….	….

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.6 Activity 3: Introducing the idea of Learning by training a model for recognizing voice commands

3.6.1 Description

In this activity the students will be introduced to the 3rd Big Idea, namely Learning, by training a model to recognize specific voice commands. Through this activity, they will understand the role of Machine Learning (ML) and Machine learning algorithms in assisting computers to learn. In particular, they will learn how to use the Personal Audio Classifier ML tool to train a model that classifies voice commands according to specific criteria.

3.6.2 Using Personal Audio Classifier to train a model

In the previous activity you learn how to use the Speech Recognition service to record voice commands and turn them into text, so as to navigate the robotic car. In this activity you will learn how to train a model to perceive a number of voice commands, and classify them according to specific criteria. To do that you will use the Personal Audio Classifier (PAC) training environment (https://c1.appinventor.mit.edu/).

Figure 24 presents the PAC training environment. As you can see the model is empty. There are no categories, no labels, and no classified sounds. You will therefore need to create a number of categories. Each of these categories will contain several recorded audio samples (e.g., different commands that can be identified such as front, back etc.), which will be represented by a common label (e.g., front, back etc.).

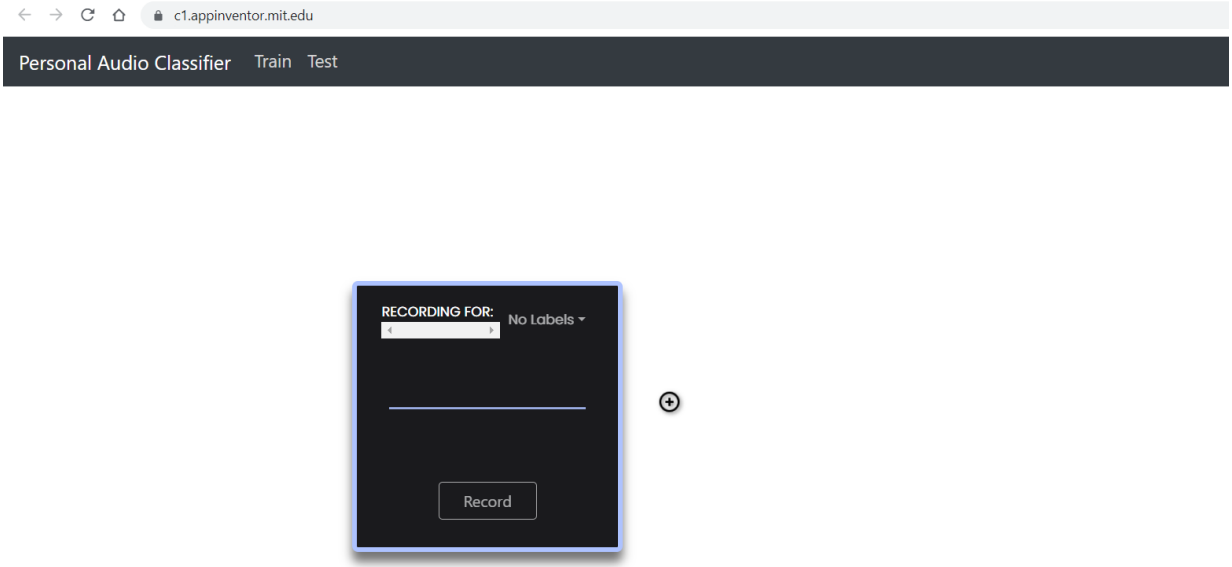


Figure 24: Personal Audio Classifier (PAC) training environment

Note: Make sure that your device (PC, laptop etc.) is equipped with a microphone/speaker. Otherwise, you will not be able to record sounds.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

The first step in creating a category of common sounds is to create a new label. Click on the **+** button, and in the floating menu "Create New Label" that appears, type in the name of the category you want to create (i.e., "front" in the example shown in *Figure 25a*). Press the "Enter" key and a new window will appear with the title of the label you have created (*Figure 25b*). This is the first category of sounds which you will fill with audio samples. To do this, press the Record button (*Figure 25b*) (**1**) to start recording the audio received by your microphone (for more information, see the "**Tips on recording**" section). Make sure that you are recording for the correct category, by checking the name of the label that appears next to the "Recording for" section (*Figure 25b*) (**2**).

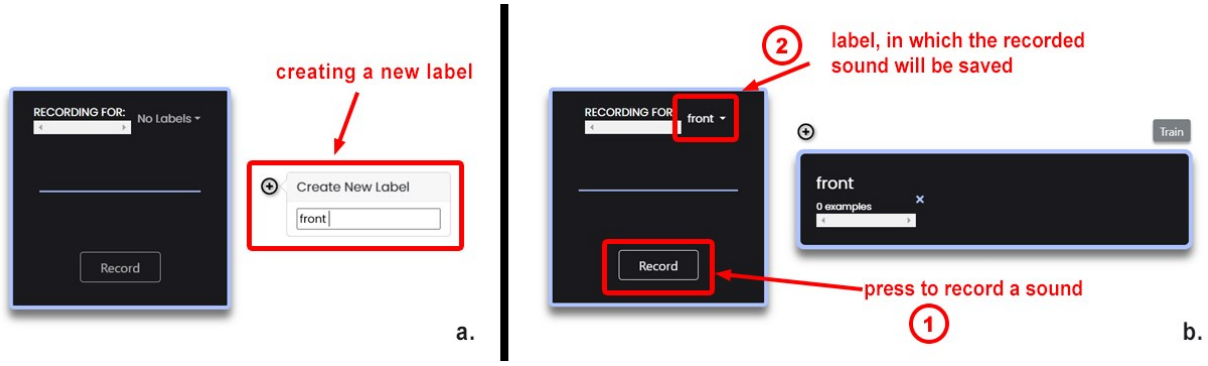


Figure 25: a. Creating a new label; b. Starting recording audio samples to fill the "front" category

Repeat the same process to create all the different categories of voice commands that you want your application to recognize and robotic car to execute (e.g., front, back, right etc.).

Note: It is important to remember the exact name of each label/category, in order for the application to successfully retrieve these categories. Encourage your students to create a table in which they will note the name of each label and a short description of the sounds stored therein.

Label/Category	**Result of voice command**
front	Move the robotic car forward
clap	Move the robotic car backwards
left	Make the robotic car turn on the left
…	…

Tips on recording (*Figure 26*):

When you press the Record button, the recording process is activated for about 1 second. During this time your microphone will record every sound that can be heard. Because of this time limit, it is recommended that you record short voice commands (e.g., "front" instead of "forward").

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

You can check the recording status by looking at the blue line **(1)** above the Record button. If the straight line turns into a waveform, your microphone is working properly. Each recorded sound is saved as a new audio file within the selected category ("front" in the example). If your microphone has successfully recorded a sound, it will be converted into an audio file, and a colored spectrogram icon will appear **(2)**. If your microphone didn't pick up any sound, a white box will appear, indicating that the saved audio file is empty **(3)**. If you want to remove an audio file that has already been saved, hover your cursor over the corresponding audio file. An X mark will appear in the top right corner. Click on it, and the audio file will be deleted.

To create a rather reliable trained model, you need to include at least 5 to 6 audio file examples/samples in each category. The number of audio files already recorded is indicated in the field below the main title of each category **(4)**. If you want to delete an entire category, press the X button **(5)** next to the "examples" field.

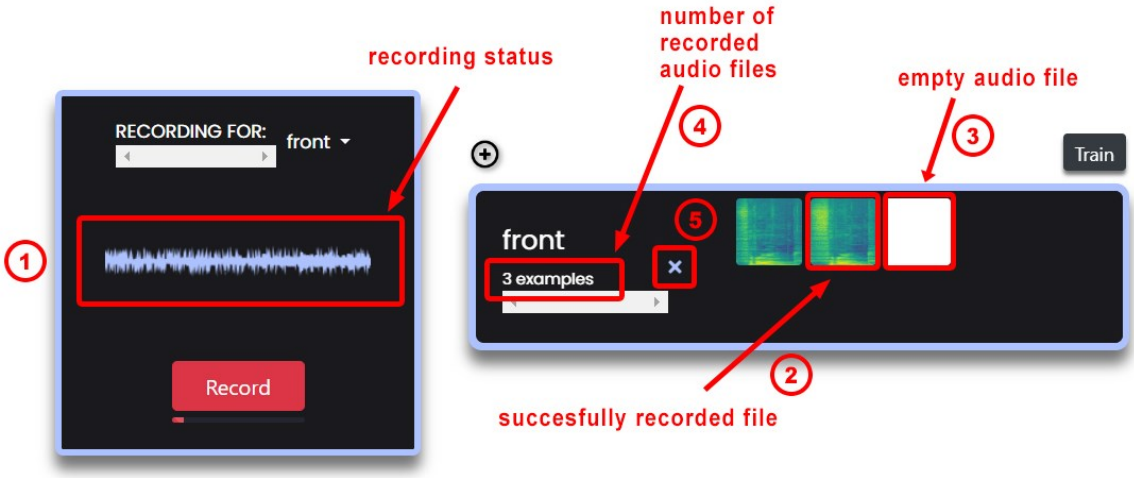


Figure 26: Information regarding the recording process and the recorded audio files

Training, testing and exporting the model

Once all the labels/categories have been created, with at least 5 to 6 sounds recorded in each category, you can proceed to train the model (*Figure 27*). To do this, click on the Train button **(1)**. A new window will pop-up in allowing you to adjust some parameters. Keep the default parameters, and click on the Train Model button **(2)** to start the training process.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

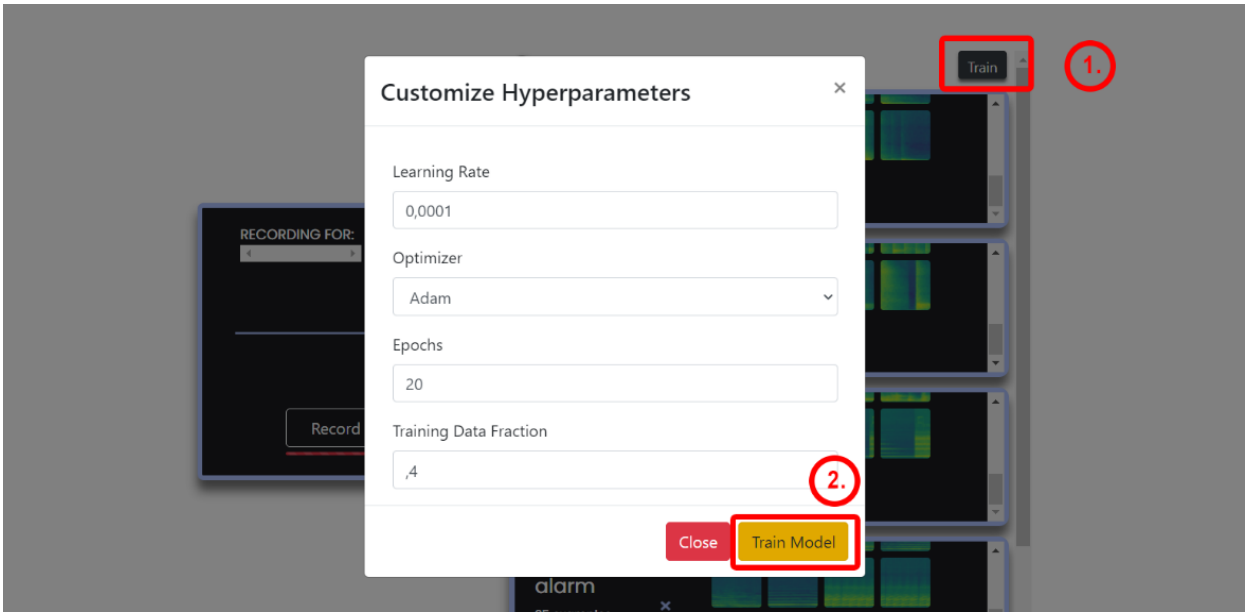


Figure 27: Training the model

Wait for a few minutes for the training to complete. Before exporting the trained model, you are highly advised to test the created trained model by clicking on the Record button **(2)**, recording a sound (e.g., press record while saying "back") and see if the recorded sound is recognized and classified into the corresponding category. You can do this by checking which of the categories in the Classification window **(4)** turn green. The spectrogram of the recorded sound **(3)** also appears, confirming that the incoming sound has been successfully recorded.

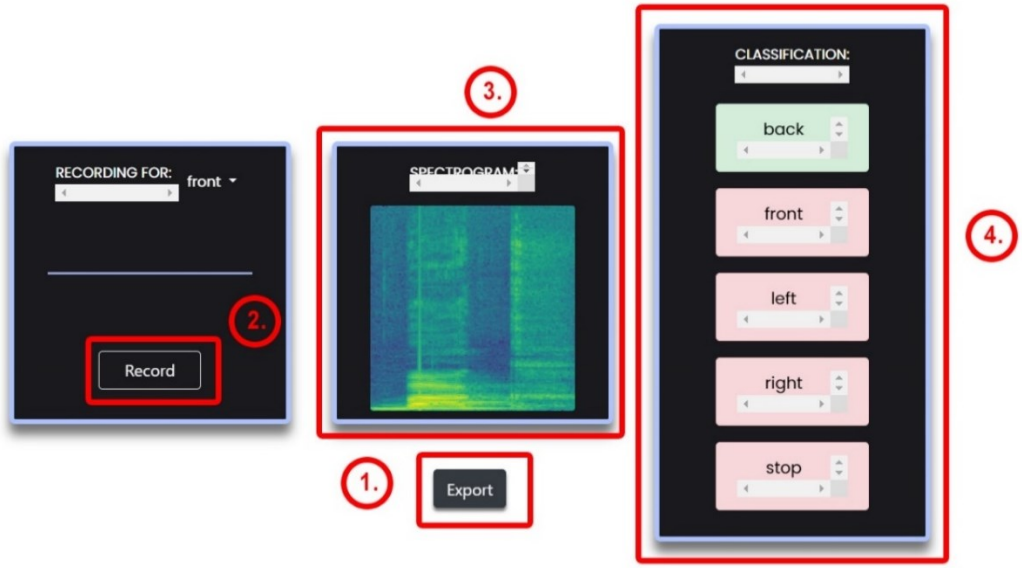


Figure 28: Testing and exporting the trained model

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

You can also scroll down the page to see the confidence level of the prediction. For example, in Figure 29 you are informed that the confidence level of the "back" prediction is 38%.

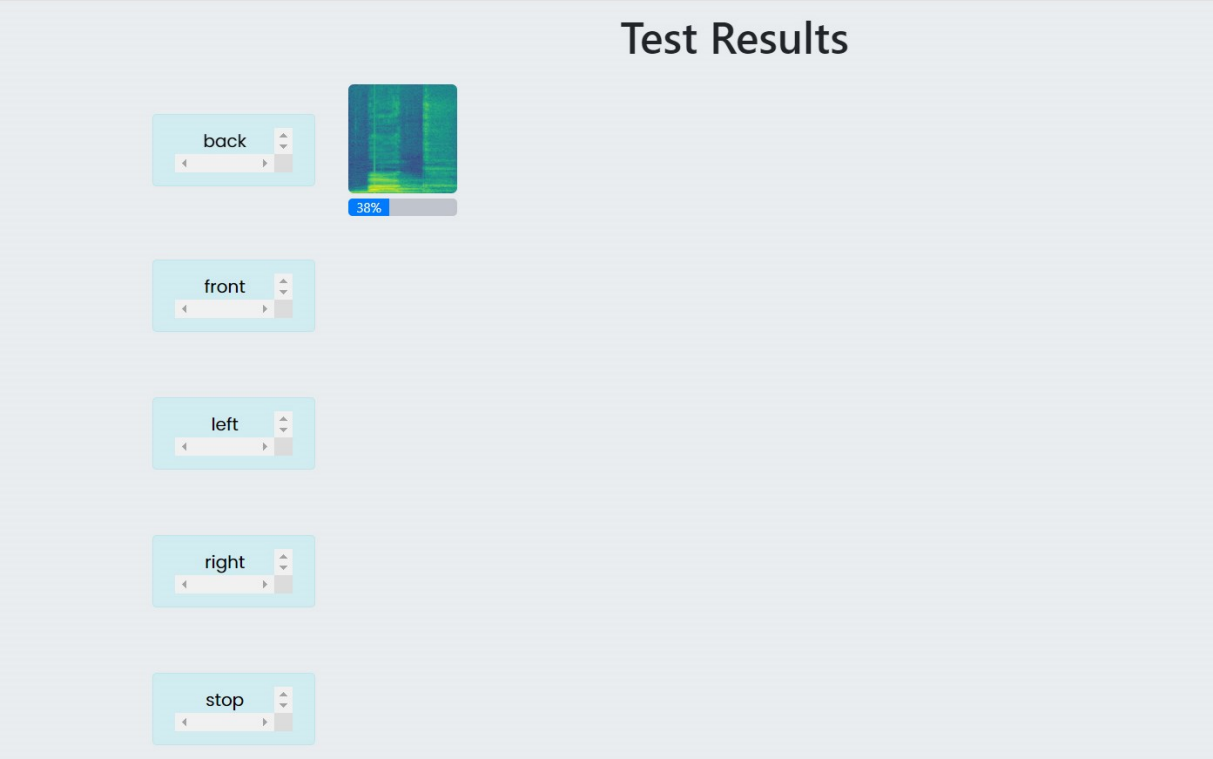


Figure 29: The confidence level of the prediction for the incoming recording

Encourage your students to run several tests with the trained model. If too many errors are found, advise them to click on the "train" tab (*Figure 30*) to make changes to the recorded data.

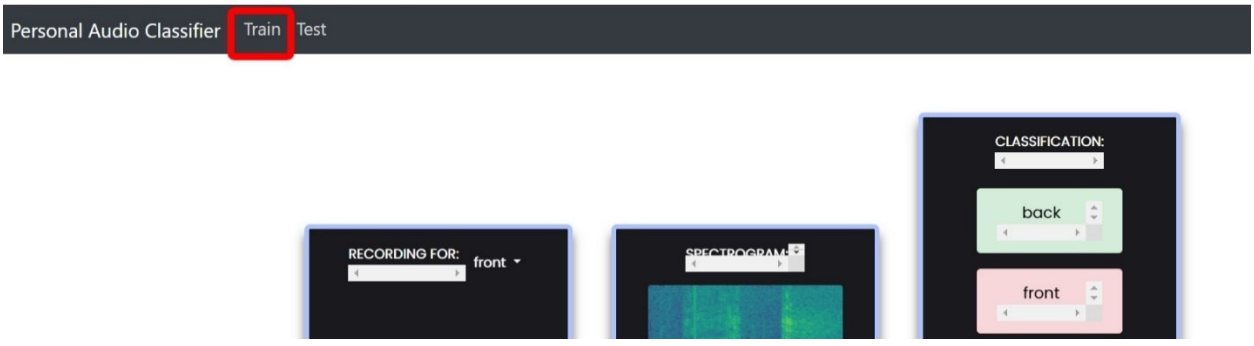


Figure 30: Click on the Train tab to return to the recording mode

If the trained model is properly working, click on the Export **(1)** button (*Figure 28*) to save the trained model locally to your computer, as an *.mdl* file.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Note: the file is automatically named to *model.mdl* when it is downloaded, but you can manually rename this file (after the download process) to something meaningful (e.g., *voice_commands_model.mdl* etc.).

Important note: unfortunately, the current version of Personal Audio Classifier environment does not allow any modification on the trained model, after leaving the page. Therefore, you should foresee enough time to complete this activity during a teaching/making hour.

In the 4th activity, you will learn how to integrate this model into the application that you have already designed in App Inventor, and use the exported model to classify the incoming voice commands to instruct the robotic car to move accordingly.

3.6.3 Experiment 3

The previous section gave some tips on how to introduce this activity to your student. The first thing to do is to explain to them what the purpose of this activity is (i.e., to learn how to use an ML tool to train a model that can be used by the robotic car at a later stage). You can encourage them to think of different scenarios where audio classification could help the robotic car to learn from its environment, but as a starting point, encourage them to think about how they can train a model to classify different voice commands.

To this end, and before opening the Personal Audio Classifier tool, encourage them to write down which voice commands they want to record, and what the expected result of each of these voice commands is (e.g., what they expect from the robotic car to do, if the voice command is “right”). To facilitate this process, you can advise them to create a table like the one below and write down their ideas.

Voice command	Result

Then, ask them to open the Personal Audio Classifier tool and briefly explain how to use it. Encourage them to create a number of labels and record some audio samples in each category. Some tips that you can give them are:

- to create as many labels/categories as the movements that the robotic car will perform
- experiment with different accents or try to emphasize different parts of a recorded word
- write down the exact name they give to each label/category. This is important for the next activity (i.e., 4th Activity)

Once they have created all the categories, ask them to train and test the model. When testing, advise them to also check the confidence level for each prediction. You can encourage them to note on a table

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

like the following whether the recorded voice command was correctly perceived and classified and what the confidence level of each prediction is.

Sound	Successfully classified	Confidence level (%)
	YES / NO	
	YES / NO	
	YES / NO	
	YES / NO	

When they are happy with the result, ask them to export the model, thus saving it locally on their computer.

During this activity you can discuss with them the different aspects of this process. To initiate a dialogue with them you can ask some of the following **questions**:

- What can an ML tool do?
- What parameters do you need to take into account when training your model?
- Is it important to evaluate a trained model before using it in an AI application?
- How can a biased trained model, classifying voice commands, effect driverless cars?
- How can we avoid biased trained models?

Through this activity the students will learn:

- How to use the Personal Audio Classifier ML tool, or similar to this ML tools
- How to train a model based on a planned classification
- How to test and evaluate a trained model
- How biased data can affect the accuracy of a trained model

3.7 Activity 4: Introducing the idea of Natural Interaction by integrating a trained model into an AI application

3.7.1 Description

In this 4th Activity, the students will learn how to integrate the trained model, produced in the context of the 3rd Activity, into the application created in the 2nd Activity, in order to observe how the performance of the robotic car can be affected when a trained model is integrated into the application. In this way,

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

they will become aware of how AI systems can be prone to errors due to the limitations of AI in interacting in a natural way.

3.7.2 Integrating the trained model to the AI application

For the needs of this activity, you can either continue working on the application created in Activity 2, or you can use the "*Robotic_car_SpeechRecognizer.aia*" file. Alternatively, you could consider of creating a new *.aia* project, but it is highly recommended to continue working on the previous file, as it would be easier for students to compare the differences in the performance of the robotic car, when using the Speech Recognizer and when using the Personal Audio Classifier.

Adding some extra components

First you need to add some extra components to application screen (on the designer menu). Specifically, you need to add **a)** a WebViewer component, **b)** the Personal Audio Classifier extension and **c)** an additional label where the results of the classification will be displayed.

a) The WebViewer component enables applications to host a URL and redirect the user to a specific web page. For the purposes of this activity, this component is added to allow the Personal Audio Classifier to load the database embedded in the trained model.

To find this component, go to the User Interface sub-menu, and drag and drop it onto the screen, under the Horizontal Arrangement layout (Figure 31).

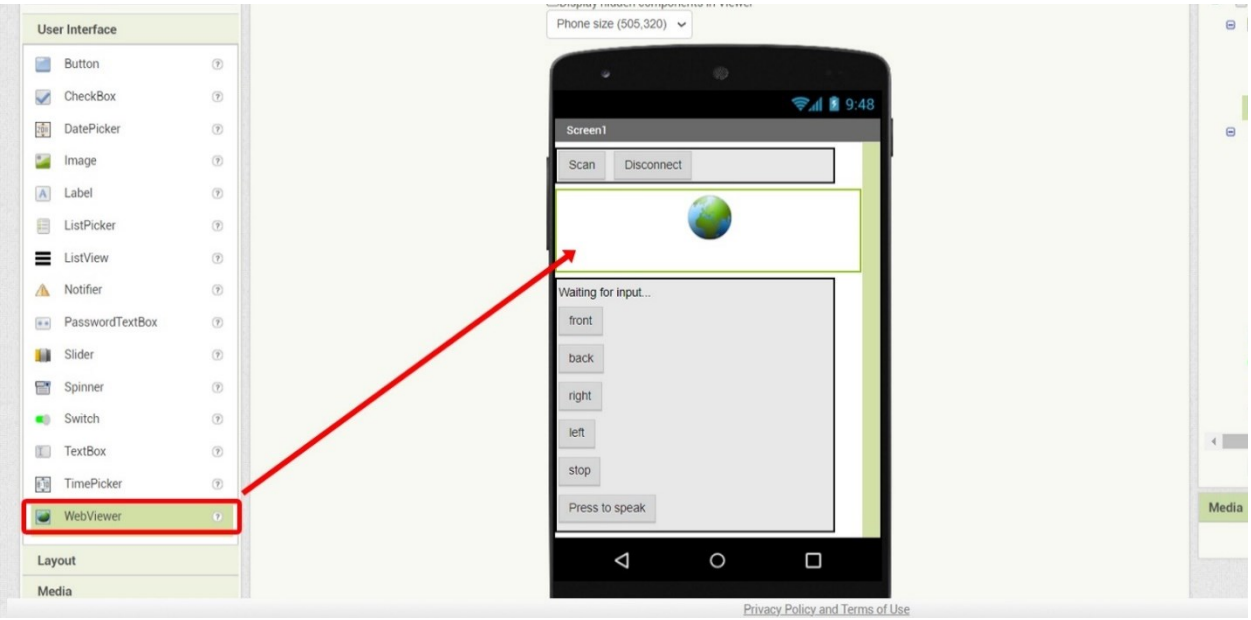
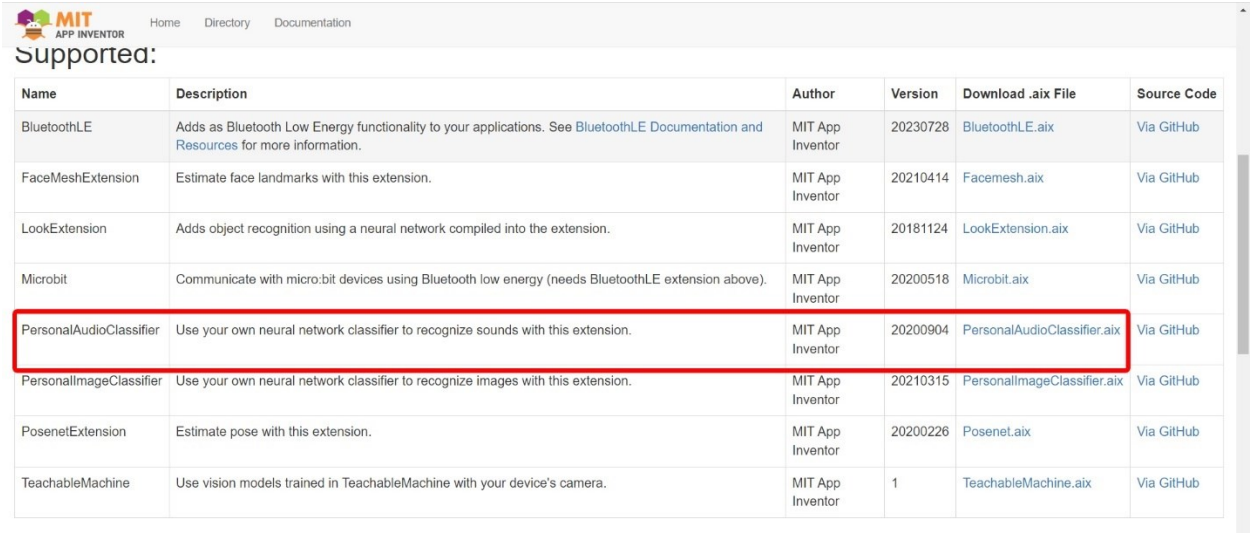


Figure 31: Adding the WebViewer component

b) The next step is to add the Personal Audio Classifier component, which allows the application to use the trained model previously created by the PAC training environment. To do this, you must first download this extension locally to your computer. Go to this link: https://mit-cml.github.io/extensions/ and save the corresponding .aix file to your local disc (*Figure 32*). Then add this extension to the App Inventor environment in the same way as you added the extensions for Bluetooth and micro:bit.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.



MIT APP INVENTOR Home Directory Documentation

Supported:

Name	Description	Author	Version	Download .aix File	Source Code
BluetoothLE	Adds as Bluetooth Low Energy functionality to your applications. See BluetoothLE Documentation and Resources for more information.	MIT App Inventor	20230728	BluetoothLE.aix	Via GitHub
FaceMeshExtension	Estimate face landmarks with this extension.	MIT App Inventor	20210414	Facemesh.aix	Via GitHub
LookExtension	Adds object recognition using a neural network compiled into the extension.	MIT App Inventor	20181124	LookExtension.aix	Via GitHub
Microbit	Communicate with micro:bit devices using Bluetooth low energy (needs BluetoothLE extension above).	MIT App Inventor	20200518	Microbit.aix	Via GitHub
PersonalAudioClassifier	Use your own neural network classifier to recognize sounds with this extension.	MIT App Inventor	20200904	PersonalAudioClassifier.aix	Via GitHub
PersonalImageClassifier	Use your own neural network classifier to recognize images with this extension.	MIT App Inventor	20210315	PersonalImageClassifier.aix	Via GitHub
PosenetExtension	Estimate pose with this extension.	MIT App Inventor	20200226	Posenet.aix	Via GitHub
TeachableMachine	Use vision models trained in TeachableMachine with your device's camera.	MIT App Inventor	1	TeachableMachine.aix	Via GitHub

Figure 32: Download the Personal Audio Classifier extension by clicking on PersonalAudioClassifier.aix

After importing the PersonalAudioClassifier extension, drag and drop in onto the screen of the designed application. PersonalAudioClassifier is also a non-visible component, therefore it will appear on the non-visible components section.

Then select the PersonalAudioClassifier component from the Components menu to modify its properties (*Figure 33*). Specifically, click on the "None…" field under both the Model **(1)** and the WebViewer **(2)** properties to correspondingly upload the *.mdl* trained model file (which you previously created on the PAC training environment) and to select the WebViewer1 component from the floating list **(3)**.

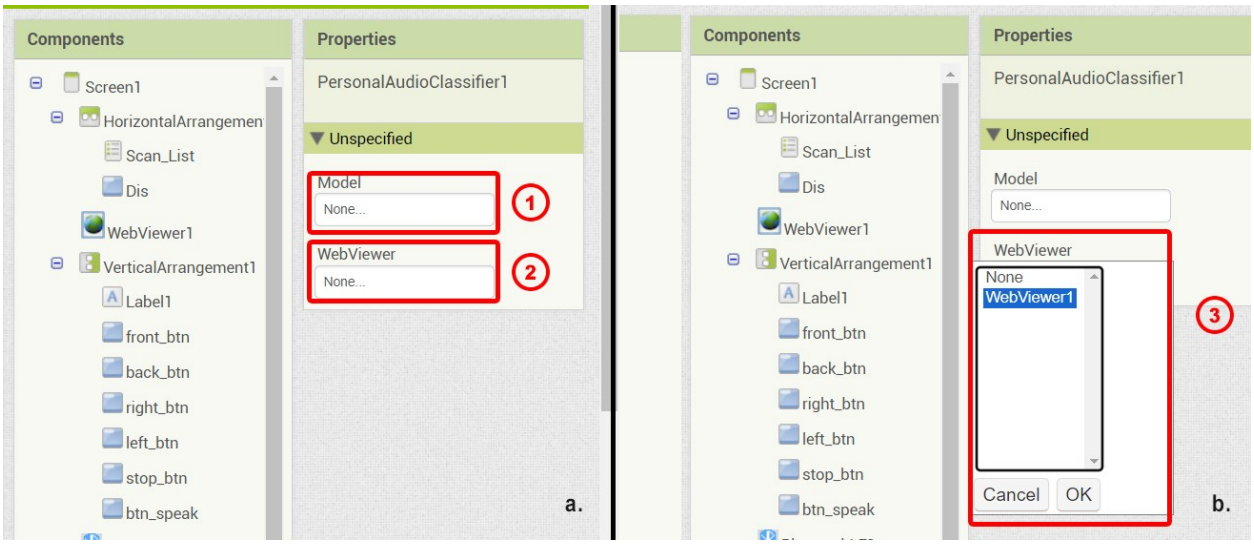


Figure 33: a. Modifying the properties of PersonalAudioClassifier; b. Upload the file of the trained model and select the WebViewer1 component

c) The last step is to add one more label, which will display the results of the classification, each time a new sound is recorded by the application.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Drag and drop a label component on the screen, under the WebViewer component and from the Properties menu, change the text of the label to “Awaiting classification” or something similar. You can also rename the label component – from the Components menu – to something meaningful such as Classification_label (*Figure 34*).

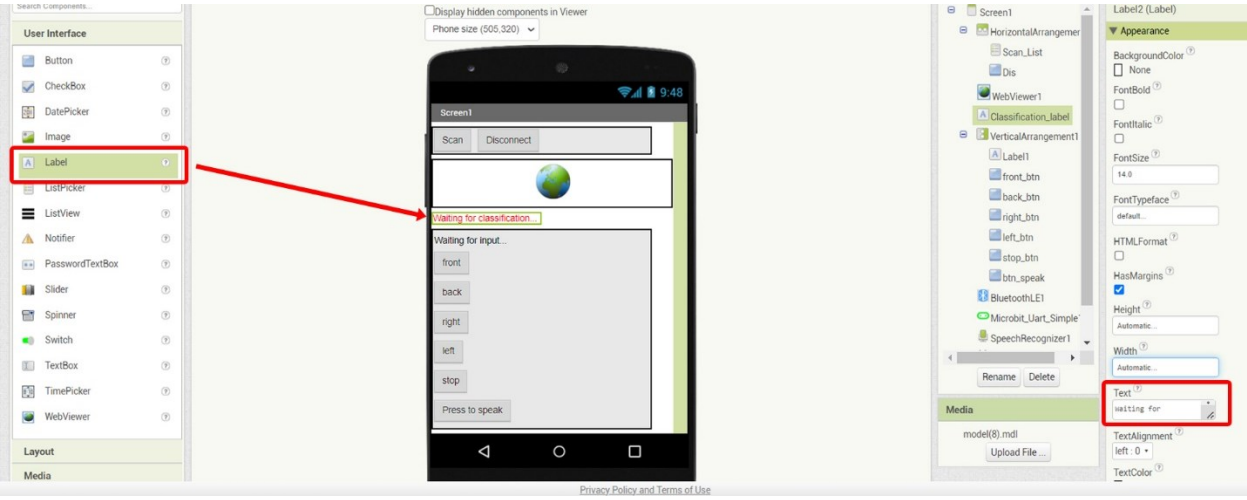


Figure 34: Adding a second label for displaying the classification results

After this step, you can program the new components.

Programming the PersonalAudioClassifier component

The next step is to program the PersonalAudioClassifier component and specifically the functionality of it after the audio classification of an incoming sound has been completed.

For this step of programming, the event block command **“when PersonalAudioClassifier1 .GotClassification…result…do” (2)** located in the floating menu of the PersonalAudioClassifier component **(1)**, must be implemented (*Figure 35*). Drag and drop this event block command onto the area where the code is assembled.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

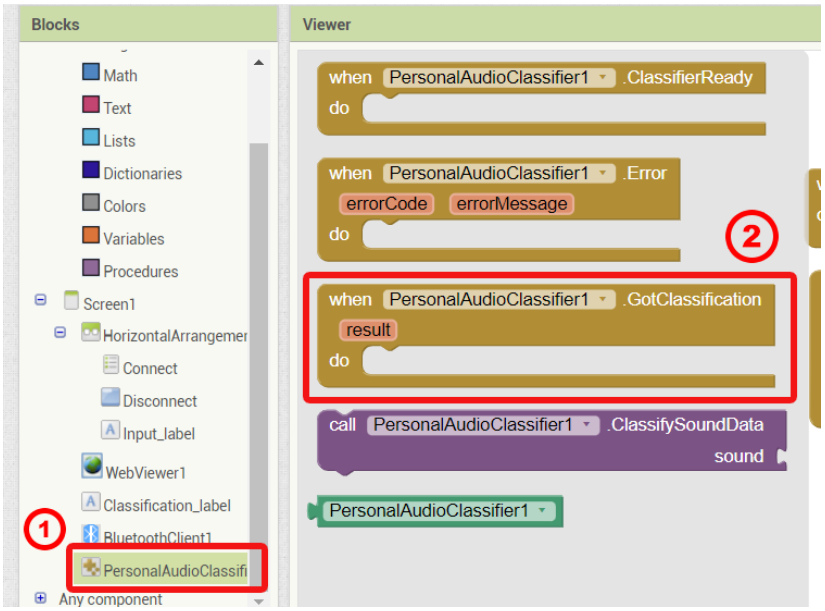


Figure 35: Finding the when PersonalAudioClassifier1 .GotClassification…result ..do..” event block command

The Personal Audio Classifier component adds a Record button to the application. This button is not visible in the design area. It will only be visible on the smart device (after the design and programming of the application has been completed, and once the application has been built and installed on a smart device) (*Figure 36*). The Event block command above determines what the application should do with the results of the recording, after the incoming voice command has been classified (i.e., instruct the robotic car to move accordingly).

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

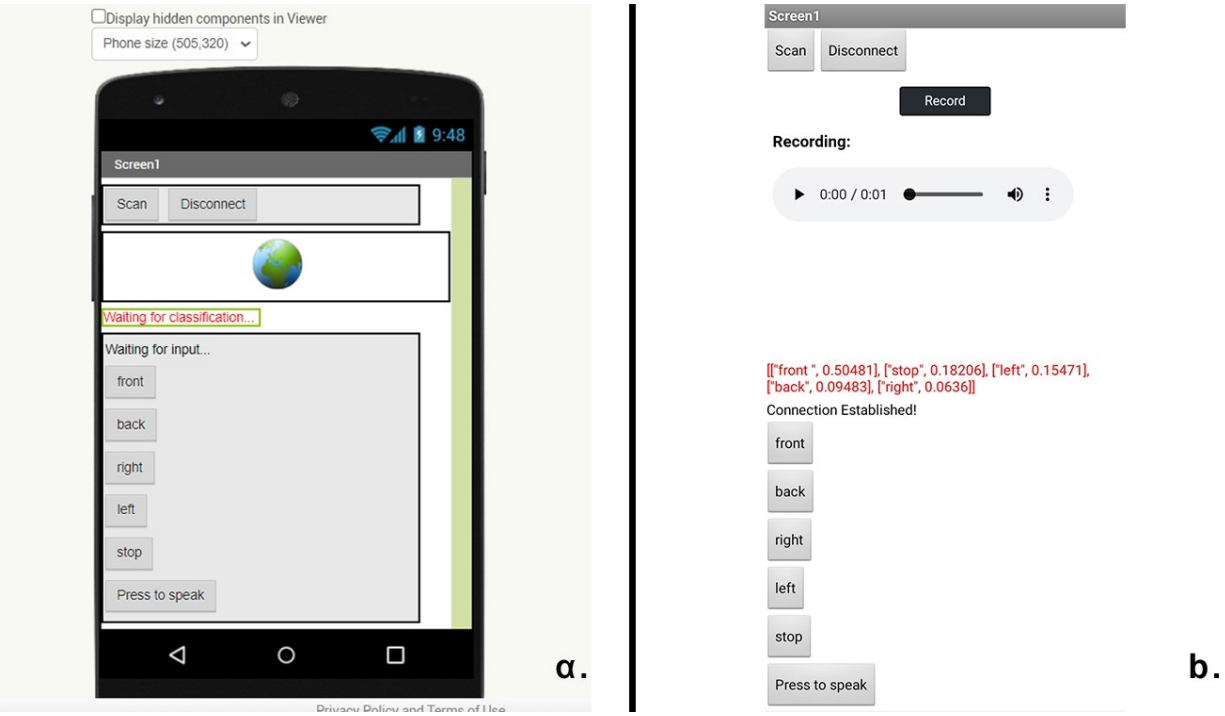


Figure 36: a. the application as appeared in the designer view; b. preview of the application when installed to a smart device

First, you need to add a command that allows the Classification_label component to display the results of the audio classification. To do this, select the Classification_label component (1), and from the floating menu drag and place the **"set Classification_label". Text to"** block command (2) inside the event handler command (*Figure 37a*). Then, move the cursor over the result **(3)** field, and drag and snap the **"get result" (4)** block command to the **"set Classification_label" . Text to"** command (*Figure 37b*).

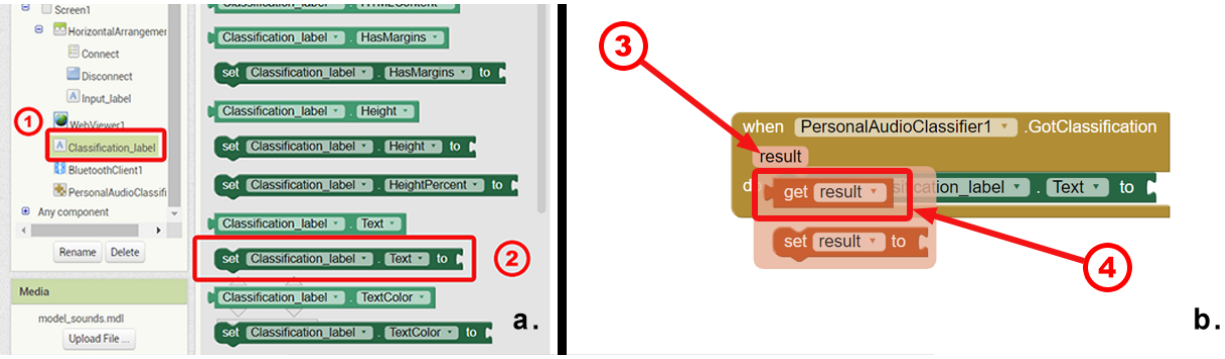


Figure 37: a. Finding the "set… Text to" block command; b. finding the "set result" block command

Through this process, the "waiting for classification" text, in Classification_label (*see Figure 34*), will change to the results of the audio classification, based on the classification model that we have uploaded, and followed by the confidence level (e.g., if the front voice command is recorded, then the text of the label could potentially change to something like the one shown in Figure 36b, i.e., "*[front, 0.50], [stop, 0.18], [left, 0.15], etc.*").

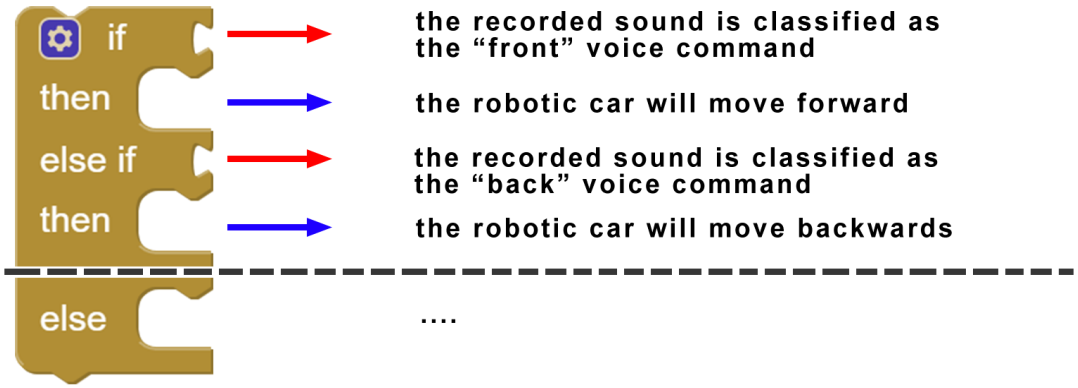
Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Then, under the "**set Classification_label" . Text to"** command, place an **"if then…else if then …else" (2)** condition, located at the Control menu **(1)** (*Figure 38*), to determine what the application will do based on the results obtained from the classification process.

Figure 38: Finding the "if then…else if then…else" condition

The following diagram presents the parameters that we need to insert inside the "**if then… else if then …else**" condition.



In particular, we need a set of commands that will recognize and check which classified category/label is being called. These commands will be placed inside the "if" or "else if" statement. We also need some more commands that will instruct the robotic car to perform the appropriate movement, according to the results of the classification. These commands will be placed inside the "then" statement.

To check which classified category/label is called, an equal (=) block **(2)** from the logic menu **(1)** (*Figure 39a*)**,** and a **"select list item list… index…"** block of command **(4)**, located at the Lists menu **(3)** (*Figure 39b*) should be used.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

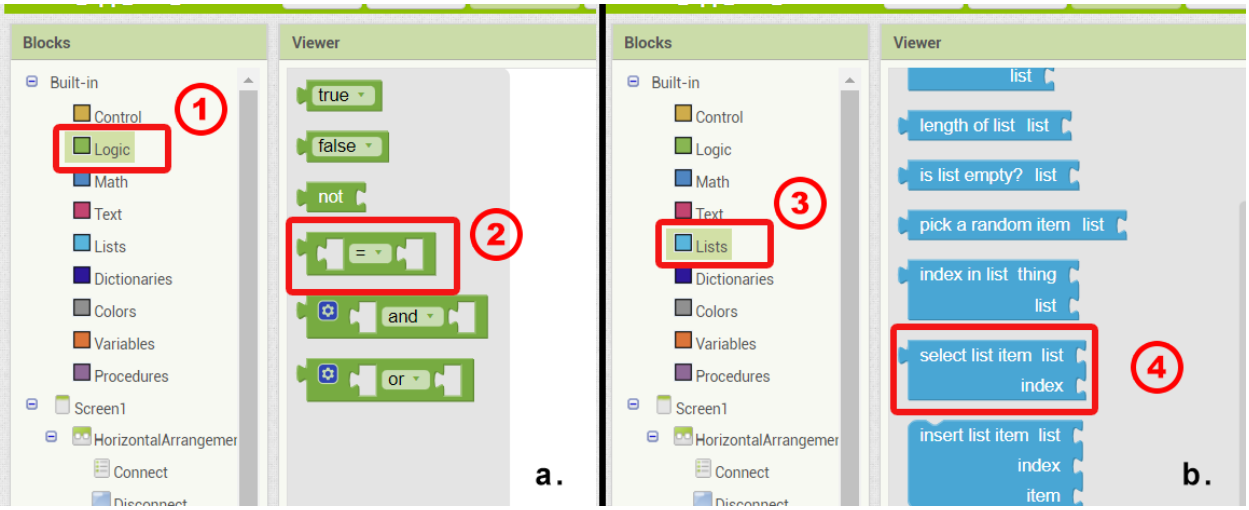


Figure 39: a. Finding the equal (=) block; b. Finding the "select list item list index" block

First, place the equal (=) block inside the if statement, as indicated in the following block of commands.

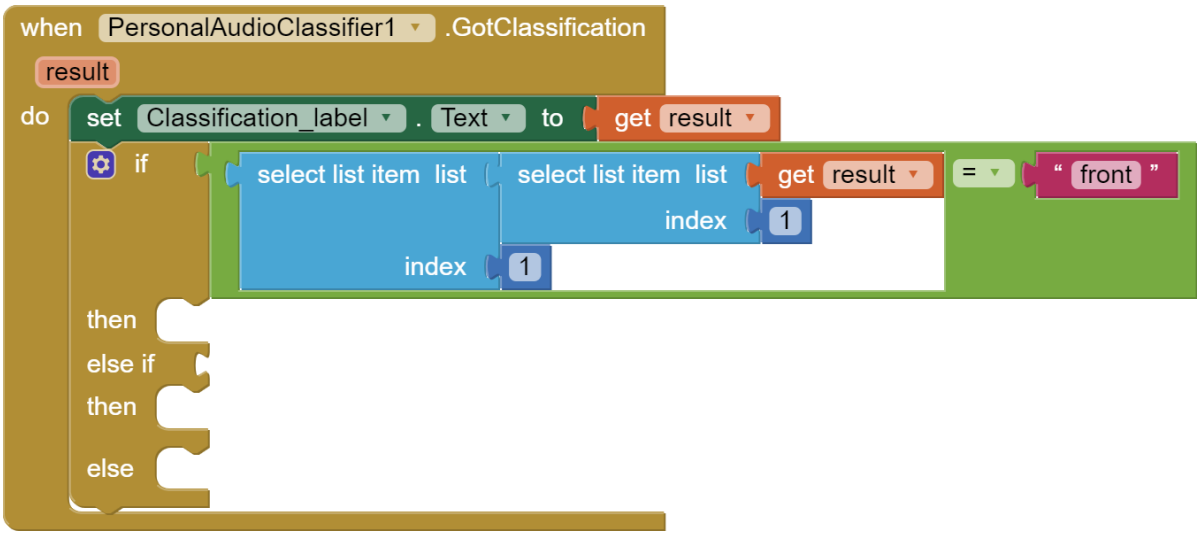
The returned result is actually a **list** with the classifications from the uploaded model. The list contains five sub-lists. Each sub-list contains one of the labels/categories (i.e., "front", "back" etc.) that the classifier thinks is a match, followed by the confidence level. In the aforementioned indicative example: *[front, 0.50], [stop, 0.18], [left, 0.15], etc.,* the classifier is 50% confident that the received command is "front", 18% confident that the received command is "stop" etc. We need to pull out the first item in the first sub-list ("back" in the aforementioned case) and test if this is the category in which the incoming sound belongs to.

To do this, **two** "**"select list item list… index…"** block of commands will be used.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

To begin with, we will get the first item (index) from the **get result** (our main list). Then this item will become our new main list from which we will extract again the first item (index), which in our example is "front" (typed inside a **text input block**). The first list item is always at index 1, and in our case is typed inside a **basic number block**.



Note: The basic number block is located at the **Math** menu, while the text input block at the **Text** menu.

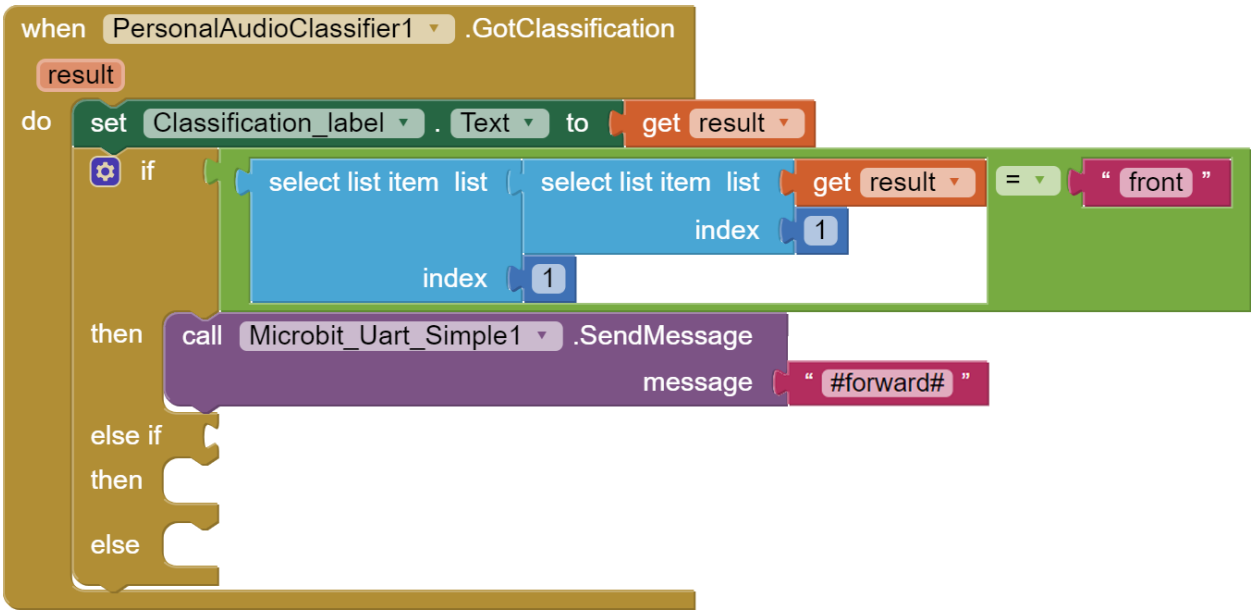
The next step is to instruct the robotic car to move forward. To do this, we need to instruct the application to transmit (trhough Bluetooth) the corresponding message (i.e., #forwardt#) to the robotic car (i.e., as has been declared in the Makecode script).

Therefore, inside the "then" statement place a "**call Microbit_Uart_Simple1 .SendMessage message**" block. Then attach/snap a **text input block ""**, and type the word **"#forward#"**.

This is how the code will look like after completing the first two statements of the "**if then..else if then…else**" condition.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.



Repeat the same process for the other commands to program your application instructing the robotic car to perform other movements, according to the classified incoming sound.

Note: if you want to add more "else if" conditions click on the blue gear (1) next to "if" statement, and from the floating menu (2), drag and drop as many new "else if" conditions as you need (*Figure 40*).

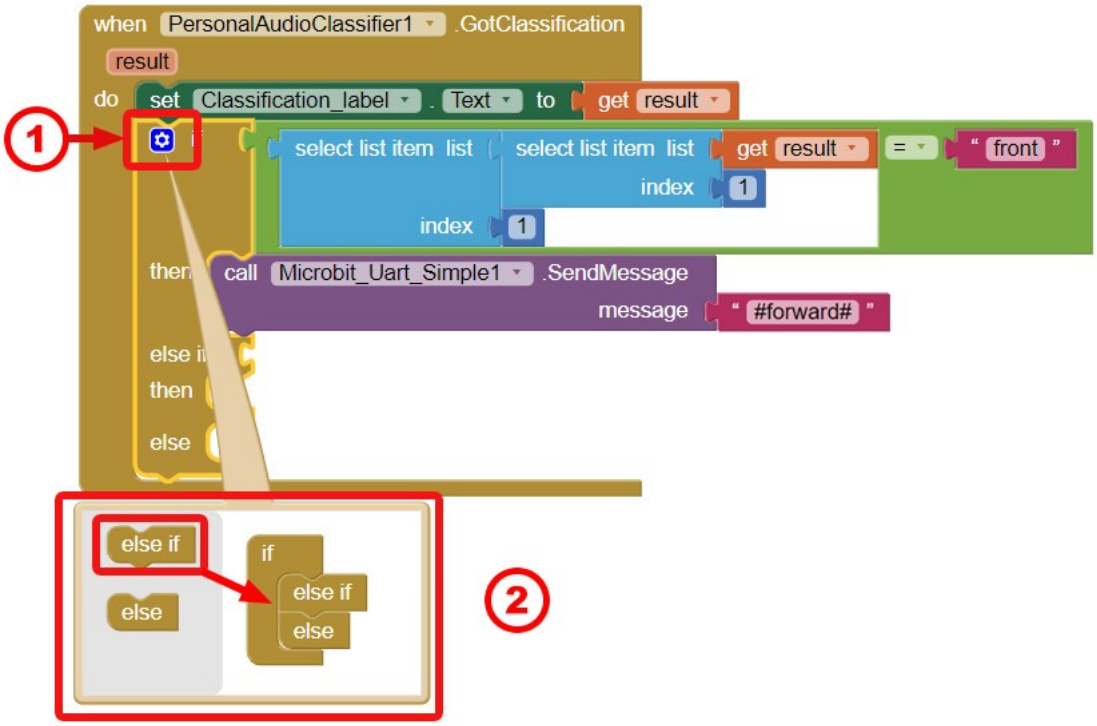


Figure 40: Adding more else if conditions

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Note: Inside each text input block, it is crucial to **a)** use the same label that you used during the production of the classification model, **b)** insert the same message that you have declare in the Makecode script.

The following image presents the entire script for the example of the classification that have been created in the present document.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

When you finish all the aforementioned steps, the application is ready to be uploaded and installed to your smart device. Go to the Build menu and select “Android App (.apk)” from the drop-down menu to begin the process of the .apk file production. This might take a few minutes. Then, install the application to your smart device through the MIT AI2 Companion application.

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Important note: In some cases (especially if the trained model has too many labels/categories), when the trained model may not work properly when integrated into the application, leading to false results. If your students encounter such a problem, encourage them to experiment with a **smaller** trained model (with only 2 or 3 labels) and think of alternative scenarios where this AI service would be more useful (e.g. starting and stopping the car). In the case of a simpler trained model, the PersonalAudioClassifier script could look like the one shown in Figure 41.

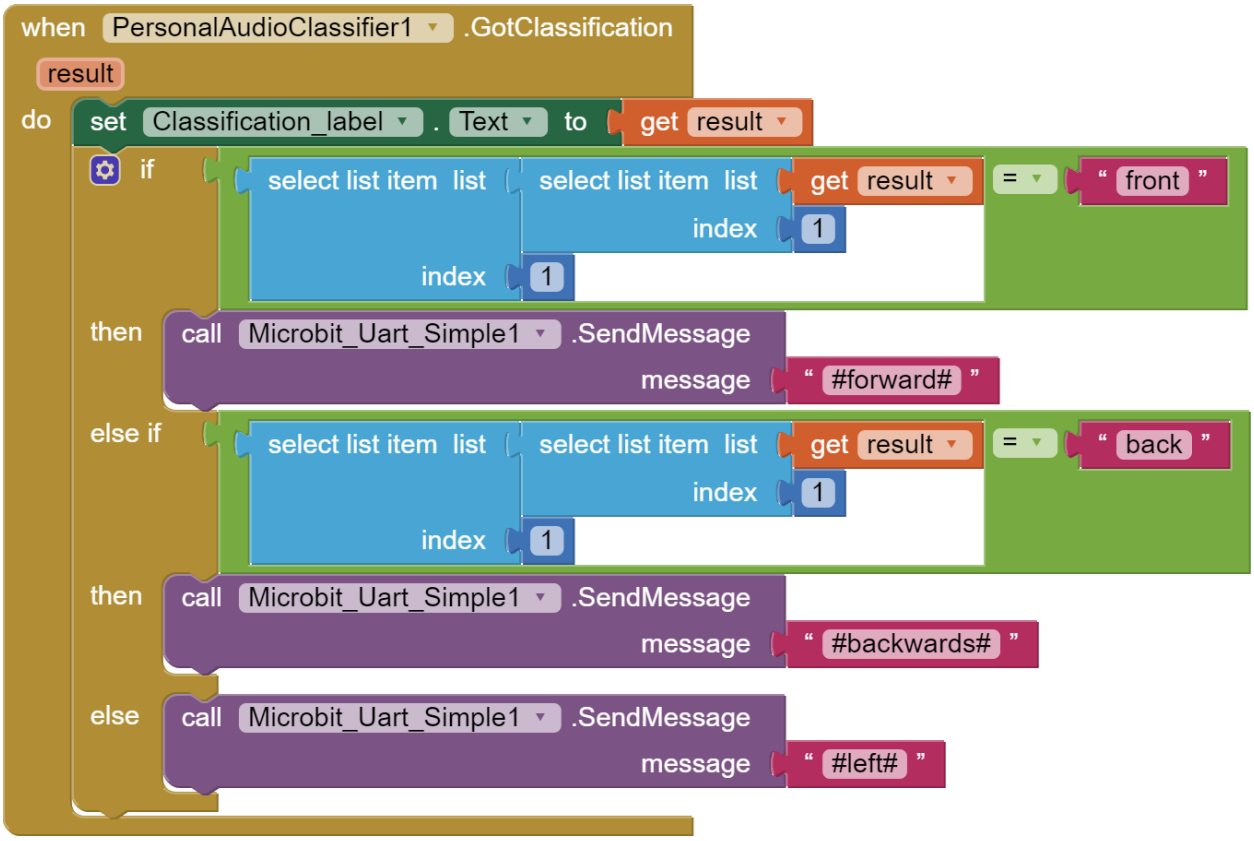


Figure 41: A script for programming Personal Audio Classifier when the trained model contained only two categories, such as front and back.

3.7.3 Experiment 4

To introduce this activity smoothly to your students you can use the corresponding worksheet document (*Students_worksheet_for_Activity_4.pdf*). Before doing so, make sure that they understand the purpose of this activity (i.e., to evaluate a trained model by integrating it into an application and to discover the limitations of AI).

After your students have created the application encourage each team to test the application created by other teams. Then encourage them to share their experiences in the plenary.

Some **questions** that you can pose to initiate the dialogue are:

- Did your application work correctly?
- Do you think different voices or different pronunciations affect the result?

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

- Can you think of any parameter that might lead to malfunction?
- What are the risks or dangers of using a trained model that contains errors?
- Sometimes, a trained model with too many categories can lead to malfunctions. Can you think of an alternative scenario for your robotic car where a smaller trained model (with fewer labels/categories) would be needed?

Through this activity your student will understand that:

- A lot of work has to be done for an AI system to interact in a more natural way
- Different pronunciations or tones of voice can lead to malfunctions
- There are serious risks in using an AI system in everyday life, if the dataset is not properly trained or contains biases

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.8 Activity 5: Introducing the idea of Societal Impact

3.8.1 Description

In this activity the students will be introduced to the 5th Big Idea, namely Societal Impact, by reflecting on the experiences they have gained while carrying out the previous four activities. In particular, they will be encouraged to think about the advantages, the disadvantages and the inherent risks of using AI services and tools, as well as of monitoring data, and making decisions based on specific datasets. This activity can be implemented separately or blended with the previous four. In this way, they will become aware of several ethical decisions that need to be taken into account when designing and using AI and IoT services and technologies.

A number of questions on ethical issues have already been addressed in previous activities. Here are some additional examples to initiate the dialogue in this direction:

1) You can encourage your students to think of a scenario where the robotic car collects more sensitive data (e.g. images of people to enable the car to recognize pedestrians) and transmit this data to the cloud or to other services.

- What are the advantages and disadvantages of this technology?
- What parameters should be taken into account regarding the security of this data?

2) Encourage your students to consider how a biased model can affect the way that an intelligent robot "thinks", leading to the construction of biased representations of the world. Encourage them to imagine a scenario where a driverless robotic car, used to rescue trapped people, is trained to recognize audio distress signals it receives, but is not properly trained. What would be the consequences of this misrepresentation of reality?

3) Imagine a case where a robotic car is trained to recognize only male voices and only baritone male voices. How can such a biased trained model affect the lives of several other people?

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

3.9 Material and Resources

Type of Resource	Title	Topic	Link
Pdf file	T2.4_Creating_the_robotic_car	Guidelines for creating the robotic car	
Pdf file	T2.4_WarmUp_ activities_for_the_robotic_car	Warm up programming activities for becoming familiar with the commands for the robotic car	
Pdf file	Circuit_card_Activity1	Material for helping students with the creation of the circuit for the 1st activity	
Pdf file	Half_baked_Activity1	Document with a half-baked solution of the code for the 1st activity	
Pdf file	T2.4_App_Inventor_Warm_Up	A warm up activity for becoming familiar with MIT App Inventor and create an application for controlling the robotic car remotely	
.aia file	Remote_control_Microbit	An App Inventor file for using the file that produced in the framework of the App Inventor Warm Up activity, and can be used for the 2nd Activity	
Pdf file	T2.4_Programming_the_robotic_car	A document with guidelines for creating the script in order to program the robotic car to follow the orders received by the designed application	
.aia file	Robotic_Car_SpeechRecognizer	An App Inventor file that can be used for the 4th Activity	

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Pdf file	Students_Worksheet_for_Activity_2	Students' worksheet for helping them toward implementing the 2nd Activity	
Pdf file	Students_Worksheet_for_Activity_4	Students' worksheet for helping them toward implementing the 4th Activity	

3.10 The Hardware for the robotic car

Figure 42 presents the basic electronic components that you need for creating the robotic car. In particular, you will need a BBC micro:bit microcontroller (**1**), a Kitronik Compact Motor Driver (**2**), a 3AA (or 4AA) battery holder (preferably with pre-attached wires) (**3**), 2 DC gear motors (preferably with pre-attached wires) (**4**) and 2 wheels (**5**).

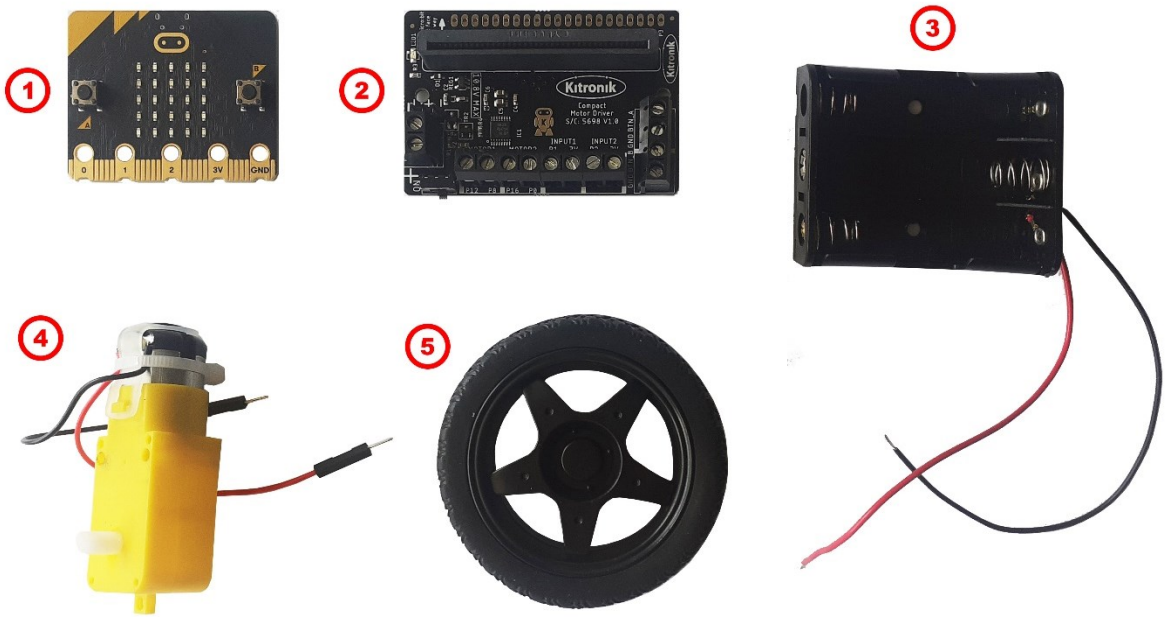


Figure 42: The electronic components needed for creating the robotic car

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.