

Lesson 8: mBot2 at your service

Subject: STEAM

Grade(s): 5th and up

Duration: 45 minutes

Difficulty: Intermediate

★ Lesson objectives

By the end of this lesson, students will be able to:

- Make the mBot2 carry out tasks using speech recognition and WIFI connection
- Combine speech recognition with variables for dialogs
- Send messages via cloud messaging
- use Boolean statements with sensor data

★ Overview

Speech recognition is the technology that converts the spoken word (but also entire sentences) back into text. It is a quite computational task to recognize even a single word just by the variation in frequency. And to be of practical use, speech recognition should work independently from an individual voice: having a high or low voice, slow or fast speaking should not make any difference. Speech recognition is a part of user interaction in consumer electronics (like smartphones, TVs, smart home), but also for assistive systems.

In this activity, you will program a short dialog with the mBot2 acting as a service robot. You can extend this task later to simulate a user dialog for other tasks as well.

Focus

By the end of this lesson, students will know:

- What speech recognition is and how to apply it
- How to make an internet connection using WiFi
- How to use speech recognition and develop a dialog with the user

- Send messages to remote computers via cloud messaging
- Use variables to describe a stage in a process (like ordering, user interaction)

Pre-lesson Checklist

What do you need?

- PC or laptop (with USB output) with the mBlock software installed, the web version (also for Chromebook), or a tablet with the mBlock app installed
- The mBot2 with a CyberPi
- A USB-C cable or Makeblock Bluetooth dongle

Lesson Plan

This lesson consists of four steps and takes a total of 45 minutes.

Duration	Contents
5 minutes	1. Warming up <ul style="list-style-type: none"> • Speech recognition in everyday life • Getting to know speech recognition with mBot2
10 minutes	2. Hands-on <ul style="list-style-type: none"> • Setting up a WiFi connection • Programming and applying speech recognition • Cloud messaging
25 minutes	3. Trying out <ul style="list-style-type: none"> • Programming mBot2 to become a robot waiter
5 minutes	4. Wrap-up <ul style="list-style-type: none"> • Showtime: show what you did with your robot in a fun, short movie for later discussion • If your teacher allows, share the end result on social media with the hashtag #mBot2WIFI • Reflection: what are you most proud of? What would you like to improve about your robot?

≡ Activities

1. Warming up (5 min)

Step 1: Warming up

This step consists of two parts:

1. Speech recognition in everyday life
2. Getting to know speech recognition with the mBot2

1. Speech recognition in everyday life

Speech recognition is the technology that can identify words and sentences in the sound of the human voice and translates this back to text. This is a very computational task, since what all the algorithms can look at is a change of frequency over time. But even this differs from person to person – just think of low or high voice, speaking fast or in a dialect. Even the mood can change the “tone” which the same sentence sounds like. The research of decades in computer science and linguistics allows these complex tasks to be incorporated more and more in everyday life.


Some recognition systems require a form of “training”, where the user reads specific texts known to the detecting algorithm – adapting a computer model to the individual tone of voice. Other only identify very specific words – they are used in user dialog systems mostly. Some systems can recognize and translate texts regardless of the context.

The CyberPi and the mBot2 can use voice recognition without the need for complex coding. However, since the output is “just” text, the program must analyze it and process further. Speech recognition is used in consumer electronics like TVs and smartphones or smart homes to give extra comfort for the user – asking for a connection, putting an appointment into the calendar, or having the room temperature and light changed just by telling.

Next to comfort only, speech recognition can be a great help for disabled people or provide extra safety, e.g., a driver in a car can approve a navigation change due to upcoming traffic jams just by voice, instead of interacting with knobs or a screen-based interface.

In this activity, you will make use of some of the features of voice recognition.

2. Get to know speech recognition with the mBot2



The mBot2 offers a speech recognition based on block coding – it is as simple as starting the recognition for a certain time (like 2,3 or 5 seconds) using a single block and retrieving the recognized text as a string by another block. The coding blocks will be explained below.

This voice recognition can understand many different languages and is not restricted to a small set of predefined words. Therefore, it is a very computational task - it is so demanding, that the mBot2 or the CyberPi can't handle it themselves... they need to call for support using the internet. That's why first of all, an Internet connection must be established, and a user account is required as well. Setting up a user account generates a "key" to sign in into the services used (for data usage and privacy see below)

The speech recognition records the audio like you did this in the 3rd activity, and then sends this recording to a very powerful network of computers for processing. They take care of the translation back to text and send the text back to the CyberPi/mBot2. The network of computers is usually referred to as "the cloud". It is not a specific computer anymore but rather a huge cluster of them shifting the computational tasks between them.

You can use the Internet for transmitting data between multiple CyberPis in different networks and locations as well (like in school and at home). This digital service is processed in the cloud as well.

Data usage and privacy note:

Speech recognition requires an internet connection for transferring the recorded audio and sending back the transcribed text. The recorded audio is processed on servers in Europe, the Microsoft Azure cloud, but it will not be stored permanently. For using the services, a "key" is required, that will automatically be generated by creating a user account on Makeblock.com and used internally by the coding blocks. There is no need to manually enter this key anywhere. Apart from an Email and a self-chosen password, no further personal information (like names, pictures, gender...) is mandatory.

The internet connection requires a WiFi access via SSID (that is the name of the wireless network) and password – certificate-based logins are not supported. The SSID and password are stored with the program in plain text, so using a temporary hotspot might be a good idea. Ideally, you can use a smartphone with a flat-rate data plan.

2. Hands-on (10 min)

Step 2: Hands-on:

This step consists of three parts:

1. Getting acquainted with the different code blocks of the WiFi connection
2. Programming and applying speech recognition
3. Messaging and Communication in LAN and Cloud

1. Setting up a WiFi connection

Open mBlock and connect the CyberPi in the 'Devices' tab. Switch to upload mode. Once connected look for "AI" (this means Artificial Intelligence) in the code blocks and click on it. You will now see the available functions including speech recognition (if they are greyed out, you forgot to switch to upload mode...).

To use speech recognition, you need to connect the CyberPi to the Internet. Below you will see the programming code that you will use in order to have the CyberPi connect to the Internet. As soon as the CyberPi is connected to the Internet you will see this on the display and the LEDs will turn green.

These examples use an event-based approach, so one thread in the program tries to enable the internet connection on start-up. The other threads start the recognition. But for now, it does not check if the Internet connection can be established before continuing – nor does it give any indication to the user about what is happening except if it was successful. You might want to change this at a later stage when trying out own ideas...

Make sure that you put in the name (SSID) and password for the WiFi network correctly (the picture shows an example!). These are both case sensitive. Connect the mBot2 to your computer with a USB cable and switch on the upload mode. Recreate the programming code from above and try if the code works. A tip is to do this with a hotspot from your cell phone, as a school network often only allows known devices to connect (white-listed MAC addresses).



Make sure that the data plan covers the internet usage! Now connect the CyberPi to your computer.

You have now connected the mBot2 to the Internet. You can now communicate with the mBot2 via the cloud. This is what you call Internet of Things (IoT).

If you are using a separate CyberPi to show the status (like the order sent to the kitchen or the delivery process), it must connect to the same WiFi network. This is to make sure that both devices (mBot2-Cyberpi and stand-alone CyberPi) are using the same channel in the WiFi network. The router/access point decides on the channels. This ensures that all devices are on the same channel for data exchange. You can also refer to LAN communication in lesson7.

2. Programming and applying speech recognition

You are now going to program and test the speech recognition. You are going to supplement the programming code in which you let the mBot2 connect to the WiFi with a code for speech recognition.

Code block:A green Scratch block with a Wi-Fi icon, labeled 'connect to Wi-Fi'. It has two input fields: 'ssid' and 'password', each with a 'password' label.

This block starts connecting attempts to the wireless network provided. You need a router or an access point that allows devices to connect with the network and access the Internet. Some schools might only allow known devices – each network adaptor (wired or wireless) has a unique number, the MAC address. If this number is unknown to these secured networks, they refuse the connection even if the password is correct.

The easiest way around this then is a temporary hotspot, e.g. by a smartphone with a flat-rate data plan that is used only for the CyberPi/mBot2 in a lesson (check your school's legal regulations).

Any blocks following this code block will be executed immediately, so it does not wait for a connection to be made ("non-blocking"). You need to code a waiting routine yourself with the next block:

Code block:A green Scratch block with a Wi-Fi icon, labeled 'network connected?'.

This Boolean block checks if the connection to the network could have been made – if so, it reports a True. You can use this together with a wait block:

An orange Scratch block with a Wi-Fi icon, labeled 'wait until', followed by a green Scratch block with a Wi-Fi icon, labeled 'network connected?'.

This combination stops all further code executing in the same thread until a connection is successfully made. You might want to make this more user-friendly at a later stage, informing about connection attempts, indicating a successful connection, or even adding a time-out (like a reboot if no connection can be made for 30 sec.).

Code block:A green Scratch block with a speaker icon, labeled 'speak'. It has a dropdown menu set to 'auto' and a text input field containing 'hello world'.

This code block will be used in an example below – it does the opposite of speech recognition: with this block you can make the CyberPi/mBot2 read out loud a text provided. Here the text is transferred to the cloud service, and the audio file synthesized is sent back. It has an auto-detection on the language used, so the spoken language should be the same as the text uses.

Code block:

Speech recognition is available in 12 different languages. If your native language is not supported, maybe you can use English. This code block starts an audio recording for the specific time and sends it to the cloud service for detection. For the results, you need the following code block:



Code block:

A Scratch 'speech recognition result' code block. It is a light green block with a speech bubble icon on the left. The text 'speech recognition result' is in bold.

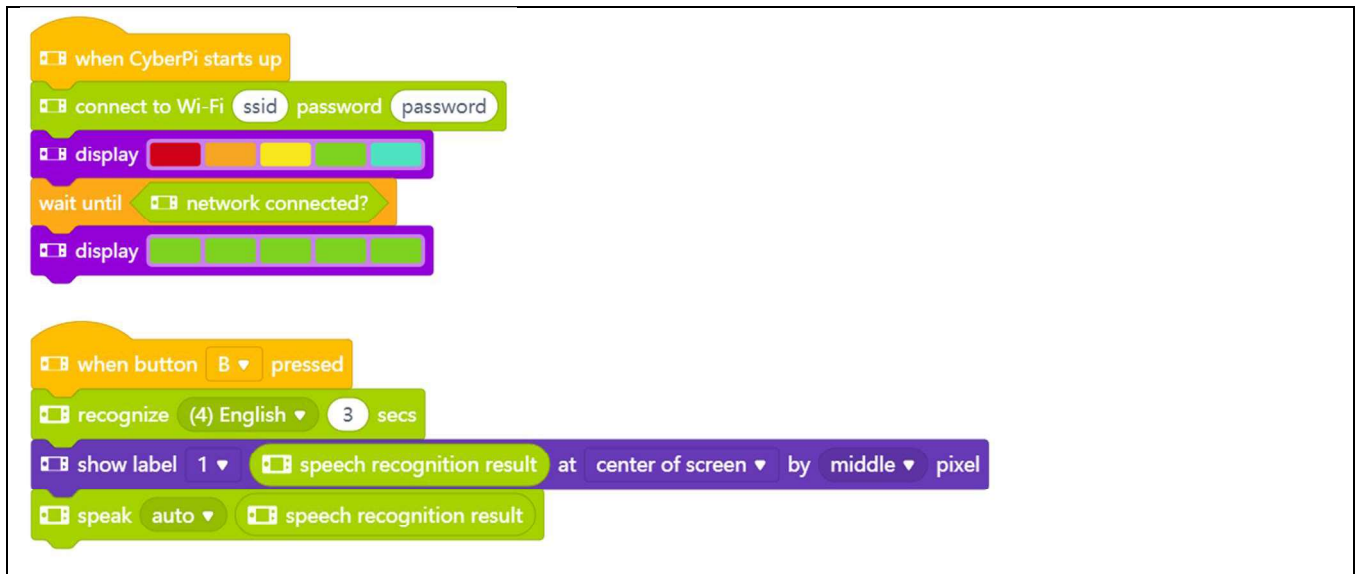
The results from the speech recognition can be accessed through this reporter block. You can save the result to a variable, in case you need to re-use it, or display it on the screen or process further with some analysis (like "does the text contain the word 'yes'?").

Code block:

A Scratch 'translate' code block. It is a light green block with a speech bubble icon on the left. The text 'translate' is in bold. To its right is a text input field containing the word 'hello'. Further right is the text 'into', followed by a dropdown menu showing 'Chinese' with a downward arrow.

The AI features even allow for language translation of texts. This code block is a reporter block that allows you to translate the text provided into a (supported) language of your choice. You could use this block together with the previous ones and the knowledge you gained from previous activities and build a voice-operated translator: press a button and translate 5 seconds of speech to a foreign language...you could even try different ones and test your foreign language skills 😊.

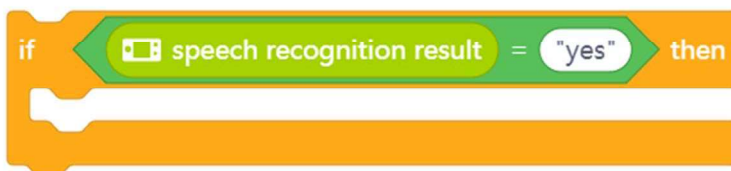
In the programming example shown here, the mBot2 is programmed so that when you press button B, the mBot2 recognizes speech for 3 seconds. The result is shown on the CyberPi display and the recognition is spoken. Recreate the programming code below and test it out. Check what you said and what was "understood" and translated back. Does it work? Then see what else you can do with speech recognition!



3 Messaging and Communication in LAN and Cloud

You can also use speech recognition to voice-control even multiple mBot2s. This can be done via a LAN connection. In lesson 7 you have already learned about the LAN connection and how to set it up. You use the result of the speech recognition to interact with another mBot2 or a CyberPi. On the "voice-commander" CyberPi, you could display a message like "Shall the robot drive?" ... By pressing a key, you start recognizing text and checking for a yes-or-no answer... If the answer is "yes", then the robot drives 10 cm. The "voice-commander"-CyberPi will send a broadcast message (LAN, like in lesson 7) to the robot.

But what happens if you answer "definitely yes" or "yes of course"? Then a direct matching with...



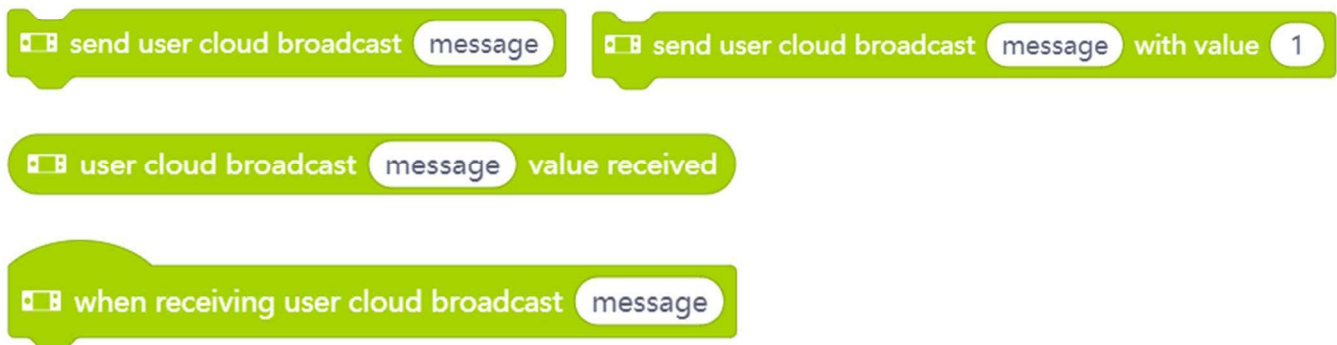
... would not work... the recognition result would be different (string "yes" is not equal to string "definitely yes").

In order to allow a broader set of answers all meaning "yes", you can simply check if the answer-string contains the word "yes":



Next to sending a message on the LAN, which requires the devices to be "local" to each other, you can use another cloud service to send messages to devices on remote locations – as long as they have an Internet connection.

These blocks work identical to the ones handling the LAN communication:

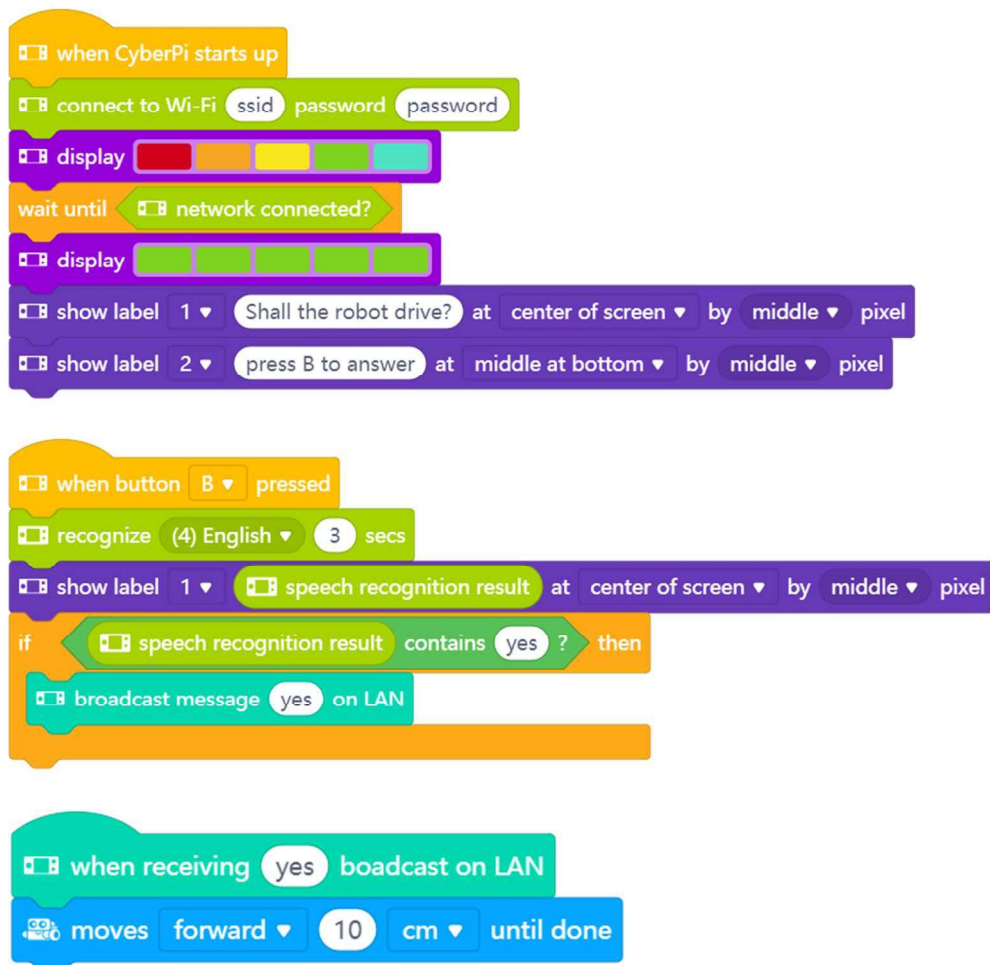


To make sure the remote side can communicate only with the devices you intend to, you need to have a Makeblock user account. This will create a "key" for the cloud communication – this key is hidden within the coding blocks and unique to a user. To make 2 remote devices communicate over the Internet, both need to be programmed from the same user account.

The cloud broadcast blocks have the advantage of being able to communicate worldwide, but they are not useful for speedy communication – it might take a few seconds for the message to arrive. The LAN-broadcasts on the other hand are fast but work only locally.

Below you see an example programming code that summarizes the above. Recreate the programming first and try if everything is working with the recognition and communicating between the CyberPi and another mBot2. For this exercise, you need to upload the same program code in upload-mode to both devices.

After you made sure everything works fine, you can modify and extend the example – maybe directional commands? Think of lesson1 and the exercise to navigate the mBot2 manually through a maze... can you do the same with voice-commands?



3. Trying out [25 minutes]

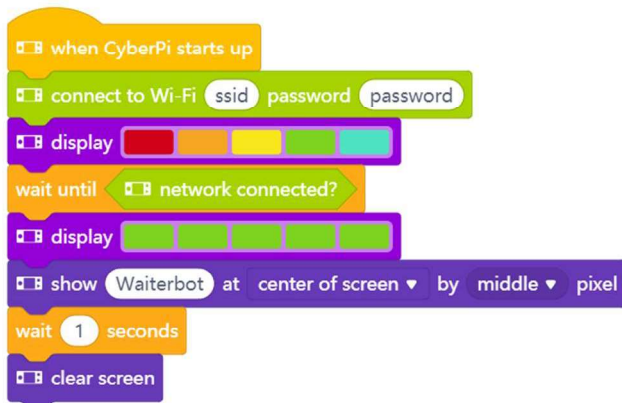
Step 3: Trying out

You now know how to make a WiFi connection, program speech recognition and deploy Cloud or LAN messaging. It is now time to get started yourself. You are going to program a polite service robot – a robot waiter. There are already restaurants in some countries where a robot comes to take the order and brings the food from the kitchen. Take a look at the following video: <https://www.youtube.com/watch?v=FFCPKmLAZb4>

The robot waiter you are going to make must be able to do a number of things:

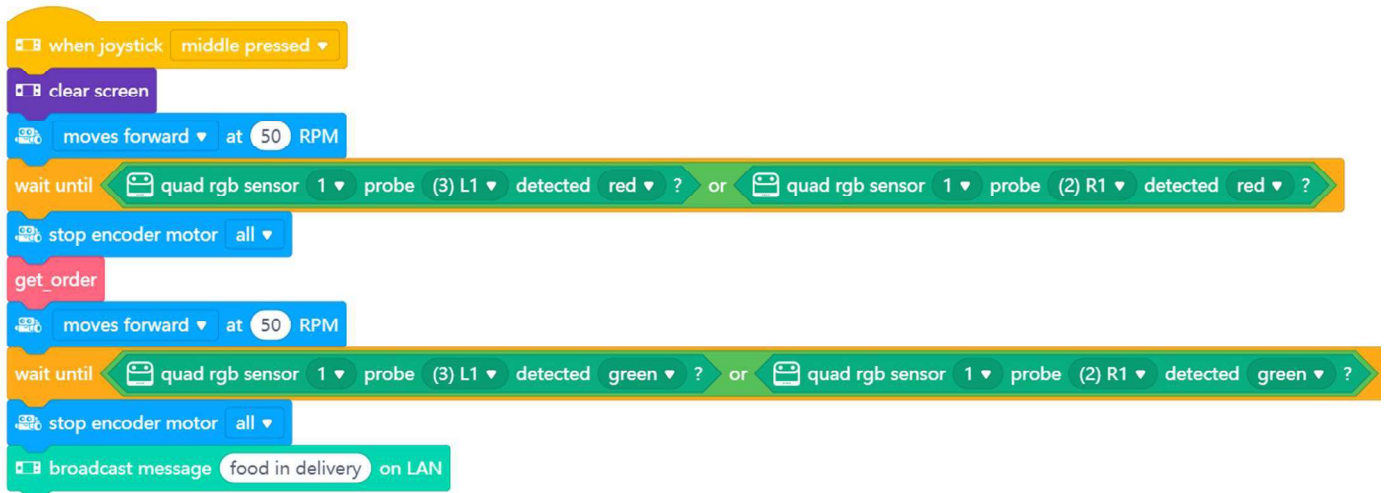
- Drive along a straight line
- Stop at a red area (your table), take an order and send the order to the kitchen
- Additional option: if a second mBot2 or CyberPi is available, then display the orders to the kitchen (so they can start preparing the food already)
- Continue driving until the mBot2 encounters a green surface (the kitchen) and stops to pick up the food from the kitchen
- Optional: if a second mBot2 or CyberPi is available, then give a notification that the food is being delivered

The start-up sequence is nearly identical to the previous examples – remember to put in the correct SSID and password:



The next code block is programmed in the same script field under the previous example. In this code block, you instruct the mBot2 to start driving and then stop at a red marker to take the order. After that, the mBot2 drives on to a green mark. The colour markers indicate the table and the kitchen. To start easier, we assume they are in a straight line. If you have everything working, you can extend this further and include the knowledge you gained from previous lessons and make the robot follow a line to the table (the provided map has red and green markers on the track that can be used for this purpose). Lesson 5 explains all about line following...

As an extra addition to voice recognition, staying with the straight line between table and kitchen, you can use cloud or LAN messaging to send the order to another mBot2 or CyberPi to display.

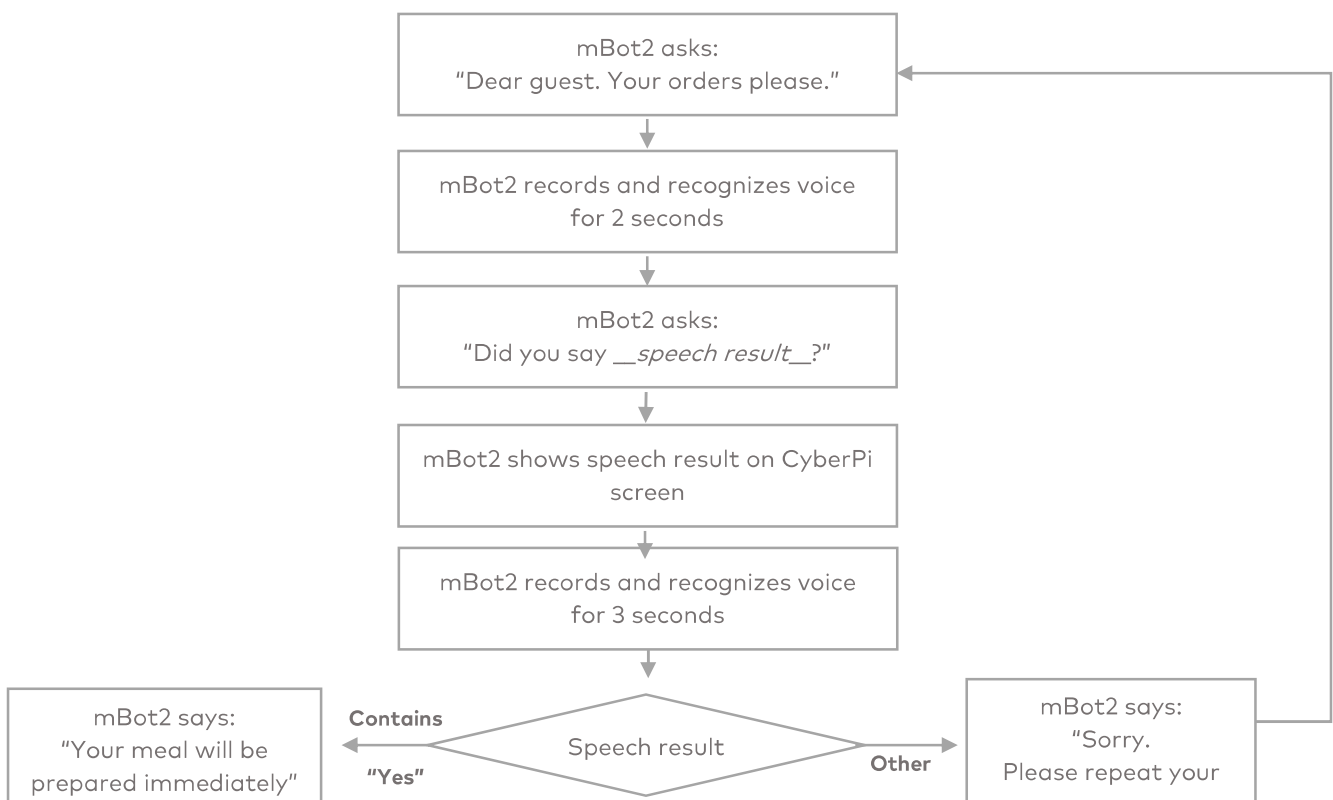


You have to create yourself the block 'get_order' in 'My blocks'. This helps keeping an overview of the main program – so it's just driving to coloured markers with getting the order in between.

To actually define what "get_order" means, let's first think about an order process:

1. Ask for the order
2. Recognize the answer
3. Check back that the understood order is correct
4. If yes, proceed (send order to kitchen and end the order sequence)
If no, ask again.

Checking is a sensible thing to prevent wrong orders. To translate these steps in a structured way, you can use a flow diagram like the following one:



From here, it is much easier to create the necessary coding blocks: you start with the code block 'define order'. You create this code block again in 'My blocks'.

In addition, you need to create two variables. In one variable we will store the recognition result from the order, so if everything was understood correctly, this can be sent to the kitchen (attention: checking back with yes/no – the answer to this would override the previous recognition result in the report block; that's why we need to store the recognized order in a variable).

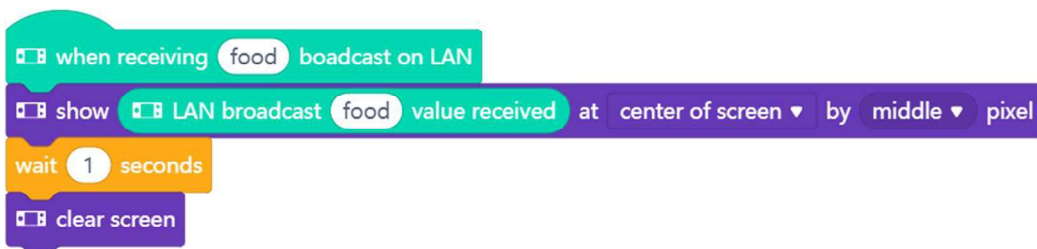
The second variable keeps track of the entire process: is the order complete (meaning, understood correctly)? If not, the ordering process will start all over again. This second use of the variable reports on process data (stage).



You can use the knowledge from previous lessons to help with the program, e.g. light up LEDs to make the user understand that the robot is about to speak (green) or listen (red), or you can use the screen on the waiter-mBot2 to display data. Use this at checkpoints in your program, so evaluate if your code works and give feedback to the users.

As an extra addition, you can use cloud or LAN messaging to send data to another mBot2. For a better learning experience, there are examples on how to solve specific tasks in this lesson. You can refer to them in case you have problems with your own code or get ideas on how to deal with a specific situation. In general, it is better to get a draft working and refine the code in iterations rather than trying to have everything done and programmed perfectly at once. Start with the driving sequence and instead of having the get_order-Block ready, just make a beep there to show that everything is working so far. Then work on the order process but keep the driving out. This way you can iterate faster without the robot having to drive between the markers every time you want to try your code. As final stages, combine them to solve the given task.

In the programming examples above, you also sent a message to a second mBot2 as an additional option. For example, in the 'order process' you have said that once the customer confirms the order the mBot2 sends a message via cloud or LAN messaging to a second mBot2 or CyberPi (for displaying the incoming orders in the kitchen). Now you have to program the second mBot2 to receive the message.



In the programming example below, the kitchen receives the order and displays it. Create the programming example below in the same script. Place the example next to the first programming example. This way you keep the script clear.

In the programming example 'controlling a robot' you have programmed that after detecting the green area a cloud message is sent to the customer to inform him that their order is coming. This is also programmed in the same script field next to the other blocks – you could use another CyberPi (as a smart device on the customer's table) to inform about the order status:

```

when receiving food in delivery broadcast on LAN
  play LED animation rainbow until done
  play yummy until done
  turn off LED all

```

When you have programmed everything, your script area will look like this:

```

when CyberPi starts up
  connect to Wi-Fi ssid password password
  display 
  wait until network connected?
  display 
  show Walterbot at center of screen by middle pixel
  wait 1 seconds
  clear screen

when joystick middle pressed
  clear screen
  moves forward at 50 RPM
  wait until quad rgb sensor 1 probe (3) L1 detected red ? or quad rgb sensor 1 probe (2) R1 detected red ?
  stop encoder motor all
  get_order
  moves forward at 50 RPM
  wait until quad rgb sensor 1 probe (3) L1 detected green ? or quad rgb sensor 1 probe (2) R1 detected green ?
  stop encoder motor all
  broadcast message food in delivery on LAN

define get_order
  set order_status to 0
  LED all displays
  speak auto Dear Guest! Your orders please?
  repeat until order_status > 0
    LED all displays
    recognize (4) English 2 secs
    turn off LED all
    set oder to speech recognition result
    show oder at center of screen by middle pixel
    LED all displays
    speak auto join Did you say oder
    LED all displays
    recognize (4) English 1 secs
    turn off LED all
    show speech recognition result at center of screen by middle pixel
    if speech recognition result contains yes ? then
      set order_status to 1
      speak auto Your meal will be prepared immediately.
      broadcast message food with value oder on LAN
    else
      speak auto Sorry. Please repeat your order

```


You need to upload this code to all the mBot2s and/or CyberPis you want to use. It can be the same code because they are automatically connected via the LAN or Cloud connection – and the one sending the message won't react upon it.

Create a straight line for the mBot2 to drive along and make sure there is a wide red and green marker. Upload the code and try whether the robot waiter is processing the customers' orders. Of course, there is always room for improvement. Adjust the code where you think it could be better or prettier. Maybe you start with a line-follower that can take the orders? Or you extend this exercise further and have a dedicated waiter mBot2 taking the orders and a delivery mBot2 that will bring the food? You can cooperate with other teams in your class and integrate their robots...

Discuss possible extensions like these in the class when reviewing the progress so far and elaborate a use case with multiple robots – it doesn't need to be a restaurant...

4. Wrap up (5 min)

Step 4: Wrap up

Did you succeed in programming the waiter-bot?

In this lesson, you learned about speech recognition and speech synthesis and how to apply it via a cloud service. You managed to connect the mBot2 to the Internet and so send data locally or even remotely.

Creating a dialog (like for ordering a meal) was part of the exercise, and you learned to filter the text answer for specific trigger words to have a more 'natural' conversation (not expecting the user to answer with specific responses only).

It is now time for a brief reflection. Think on your own and discuss with the group:

- What do you think worked out well?
- What could be better?
- Which parts of the lesson did you find easy and which did you find more difficult?
- What would you like more explanation about?
- Who could help you with that?