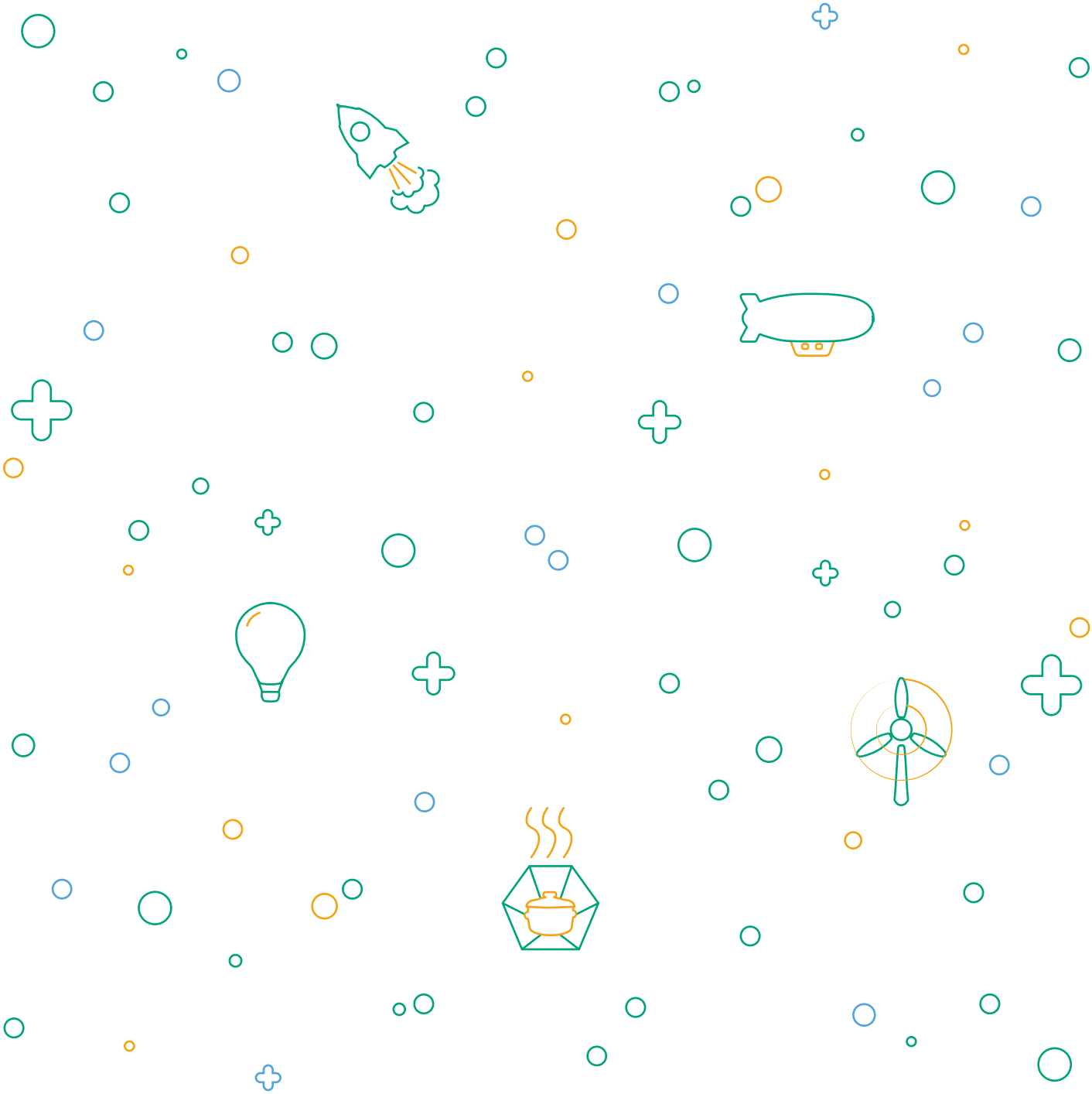
partnership with

life.augmented

FLY TO MARS!

EDUCATIONAL BOOKLET

MARTIEN ROBOT



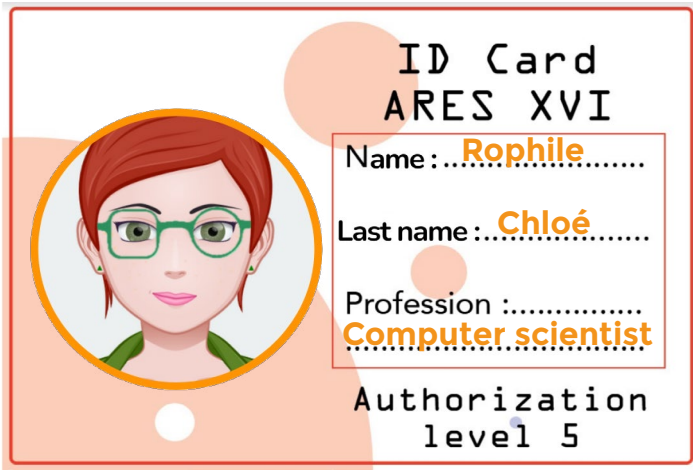
vitta science

EDUCATIONAL BOOKLET

MARTIAN ROBOT

Written by
@P7i7Plum3 and Damien Vallot
Layout and illustration by
Laura Venezia and Diana Khalipina

• 2021 Edition •
Printed in Italy



OUR TWO MARS COLONISTS ARE READY TO FACE ALL CHALLENGES!

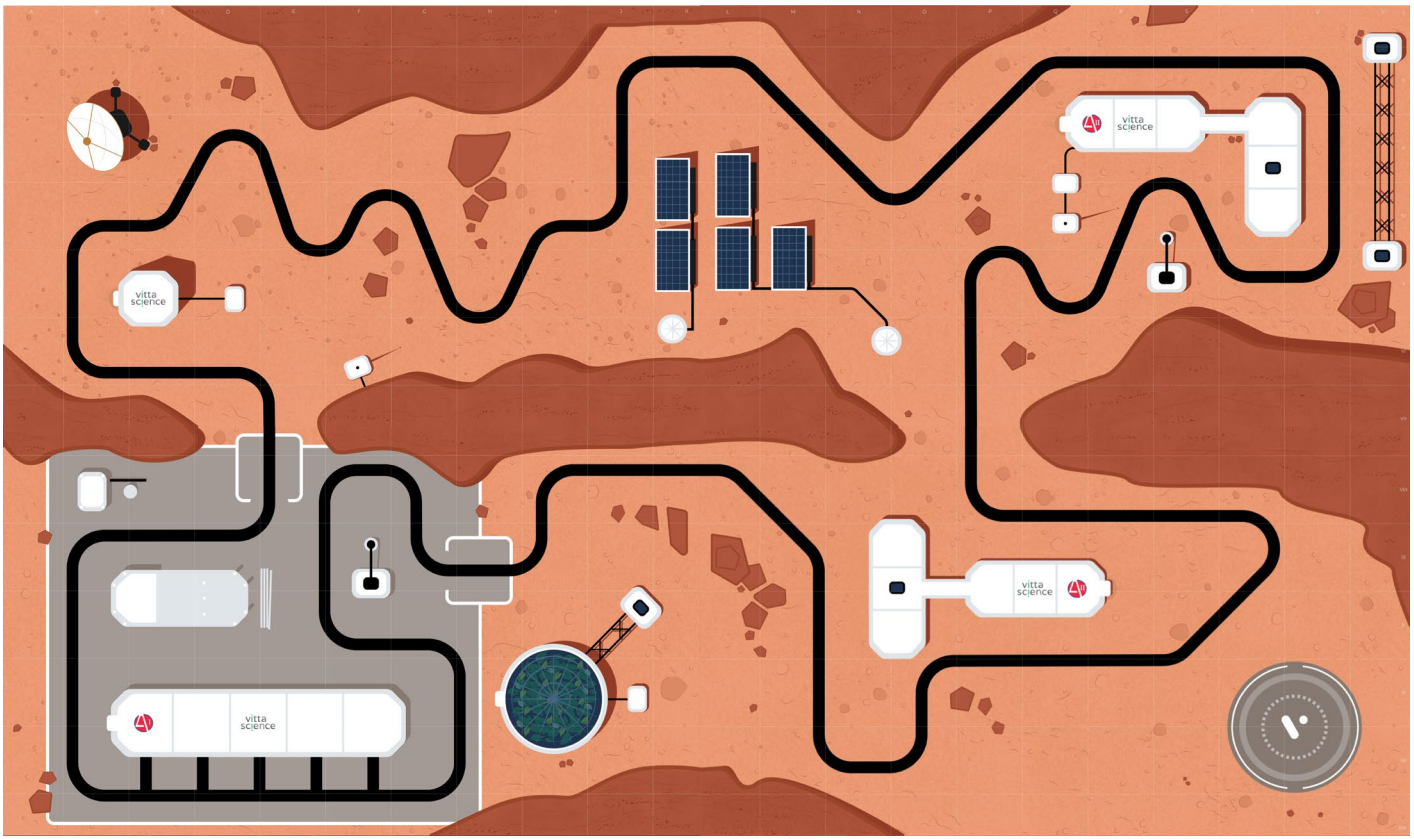
The year is 2041. The success of the Ingenuity Mars mission in 2021 is still fresh in everyone's mind. This NASA flying robot, which was deployed alongside the Perseverance rover, validated the concept of a robot attached to a control center on Mars.

Following the success of this mission, many others have been successfully carried out. Ever since the installation on Mars of the underground base Ares I in 2039, we are seeking to colonize the surface.

For this, a swarm of robots was deployed in 2040 on the surface to patrol, analyze and build the future Ares II base planned for 2042. The MSA (Martian Space Agency) based on Earth manages the whole project with the help of the first Martian colonists on site.

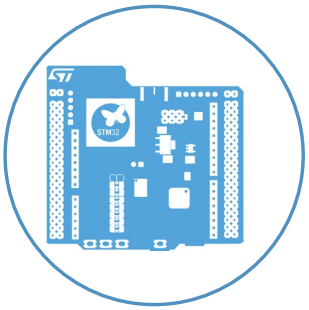
In the tasks, we will have to deal with two of our most experienced colonists from the Ares I base. Indeed, following an incident on the base, they are facing several challenges!

WHERE THE OPERATIONS TAKE PLACE

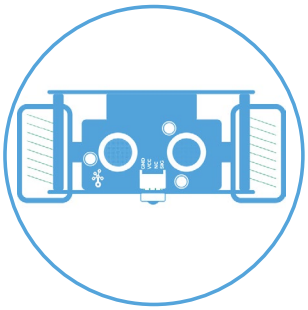


EQUIPMENT NEEDED FOR THE BUILDING AND USE OF THE MARTIAN ROBOT KIT

ELEMENTS OF THE KIT:

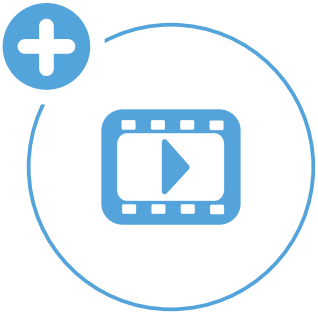


ST NUCLEO-WB55RG board



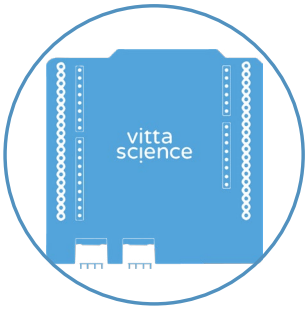
AlphaBot robot

Track for robot

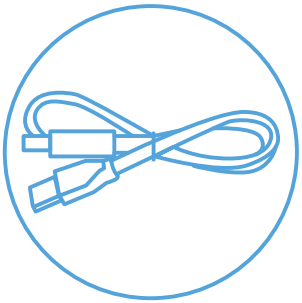


Digital resources found online

ST VL53L0X distance sensor



Nucleo Shield for Alphabot



Elements for the installation

A computer

WARNINGS CONCERNING THE USE OF THE KIT

CAUTION!

The kit includes small elements that should not be swallowed (risk of choking).

+7

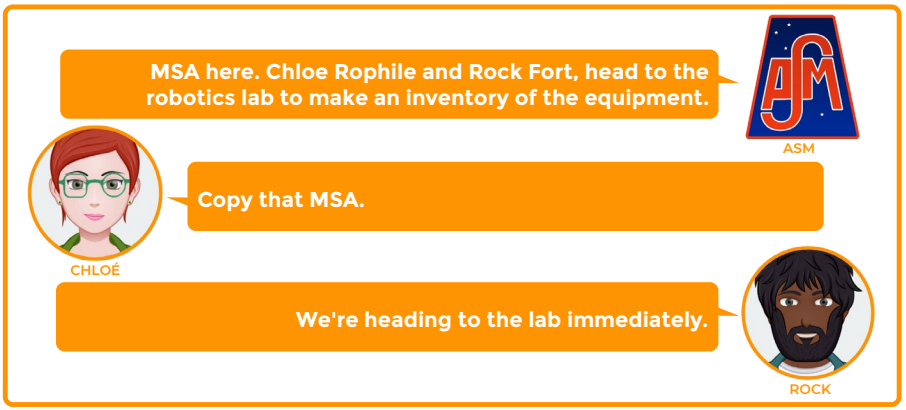
Minimum age

Doesn't suit children under the age of 7.

INSTRUCTIONS FOR SORTING AND RECYCLING

The following graphics explain the instructions that help you sort out the various elements of this kit for recycling. For more information, check out the website www.consignesdetri.fr

CONTENTS



PAGE
38

PAGE
34

Task 2: Controlling the engines

- Task 2-1: Controlling the engines
- Task 2-1: Learn to turn
- Task 2-1: Learn to make a trajectory

Task 3: Testing the obstacle detector

- Task 3-1: Discovering the ultrasound sensor
- Task 3-2: Discovering the Time of Flight (ToF)
- Task 3-3: Distance sensors and engines

PAGE
41

PAGE
44

Task 4: Real time piloting

- Task 4-1: The remote control
- Task 4-2: Testing in real "Martian" conditions

Task 5: Moving in a square mode

- Task 5-1: Moving in a square mode
- Task 5-2: Detecting obstacles before moving

PAGE
59

PAGE
50

Task 6: Line tracker

- Task 6-1: Following a line
- Task 6-2: Following a line with 3 sensors
- Task 6-3: Following a line and avoiding obstacles

Task 7: Role-playing in a complex situation

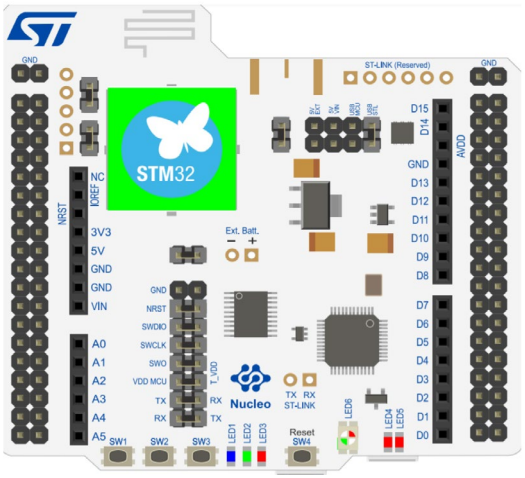
- Task 7: Finally, a routine!

Workshop • 1 depends on the tutor's estimation
Presentation of the Vittascience microcontroller and interface

The "connected plant" kit contains all the necessary equipment to build an electronic components circuit that can automatically water a plant.
It comes with a ST NUCLEO-WB55RG board developed by STMicroelectronics. This board has a STM32 microcontroller with Bluetooth low energy BLE.

• Presentation of the NUCLEO-WB55RG 15 min

The following image shows the entries and exits of the NUCLEO-WB55RG board, which can be used as a support to various electronic components circuits along with the provided components.

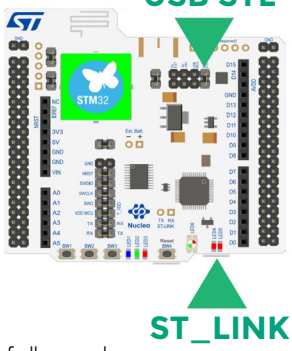


This board is programmed through the Vittascience interface in MicroPython.
Before using the NUCLEO-WB55RG board for the first time, it is required to upload a firmware to program it (see box).

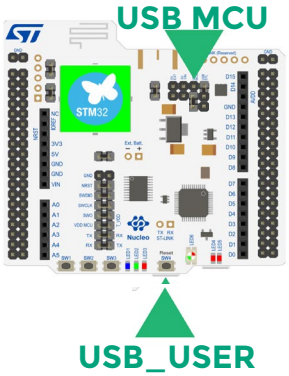
ABOUT THE UPLOAD OF THE PROGRAM:

In order to program the board in MicroPython either by code or by bloc from the Vittascience website, the right firmware should be uploaded.

(1)



(2)



Here are the steps to be followed:

1. To flash the board, make sure the jumpe (metallic piece wrapped in black plastic) is positioned on the USB STL above the board on the same level as the supply sources. If it is not the case, move the jumper to see USB STL (see image1).
2. Plug the USB cable in the STLINK port to upload the firmware. 2 red led lights will come on.
3. Download the firmware on the address: https://stm32python.gitlab.io/fr/docs/Micropython/Telechargement, then drag and drop it on your board that will appear as a USB named "NOD-WB55". Warning: do not unzip the file.
4. When the download is finished, a green led light will come on (led 6, to the right of the Reset button).
5. Unplug the USB cable.
6. Move the previously used jumper (see step 1) to the USB MCU. See image (2).
7. Re-plug the cable in the USBUSER port (the other USB). See image (2).
8. Connect the board to the computer; the red led light number 5 will come on, this shows that the board is well connected.
9. Use the Vittascience interface to program the board:
    - it is preferable to use a Chromium browser,
    - go to Vittascience - "Program",
    - select the STM32 interface,
    - click on "Connect" and select the board,
    - the loaded file main.py is executed continuously.

Tip : Encountering a problem? Have a question? We are ready to answer you on: support@vittascience.com

• Programming the board ⏱ according to the supervisor's estimations

You can find all the details about how the online programming interface works on Vittascience.com.

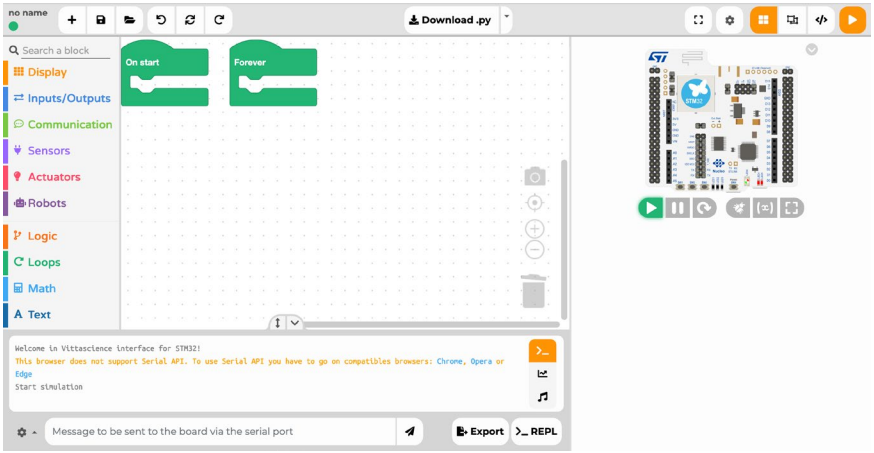
You can also program the board using Arduino software (C++ language). Tutorials for this software are available in the resources library of our website Vittascience.com.

1 • Creating an account

First of all, we advise you to create an account on our website. It is not necessary in order to get the kit, but it will allow you to save and share your programs, resources and feedback.

Visit our website Vittascience.com and click on the green icon at the top right to sign up.

2 • Interface



The interface allows you to program in bloc with a simultaneous transcription in Python.

Caution : The ST NUCLEO-WB55RG board is necessary in order to run the program once the sensors are in position.
You can find resources and programs on our website Vittascience.com to help you learn how to program the board.

Select the port: Connecter when you connect the board to the computer the interface will automatically detect to which port it is connected. The drop-down menu allows you to select the correct port if several cards are connected to the computer.

Transfer the program to the card: Téléverser the code is run on the card as soon as the transfer is completed.

Undo or Redo: he previous or next actions.

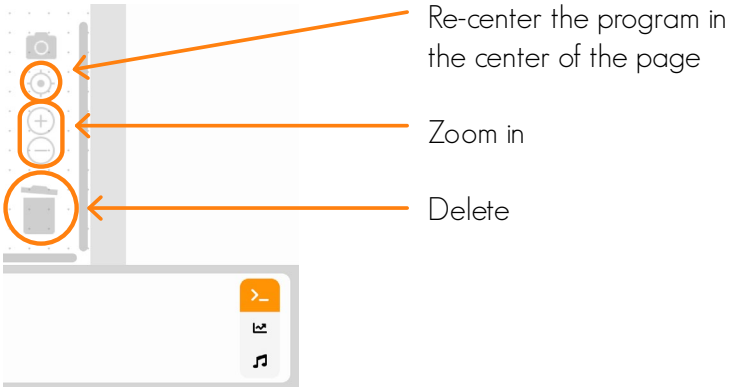
Start a new project: to start a new blank project, click on this button.

Save the project : to save your project, click on this button. For those who have an account, it is possible to share the program with the community.

Open an existing project: if you want to open programs you have already made, or have your students work on a framework you have created, they can access it by clicking on this button.

Code in Python: if you want to code directly in Python.

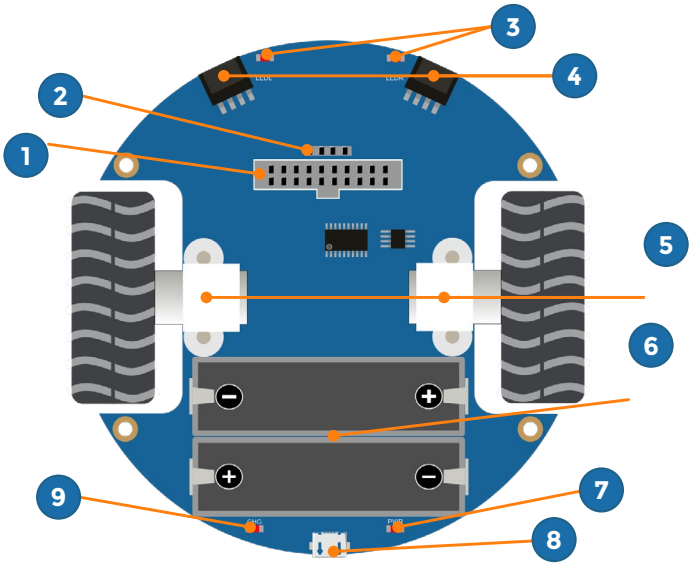
Quick access:



Tip : This programming interface is designed to be very easy to use, do not hesitate to test it and suggest it to your students.

Overview of AlphaBot2-Base

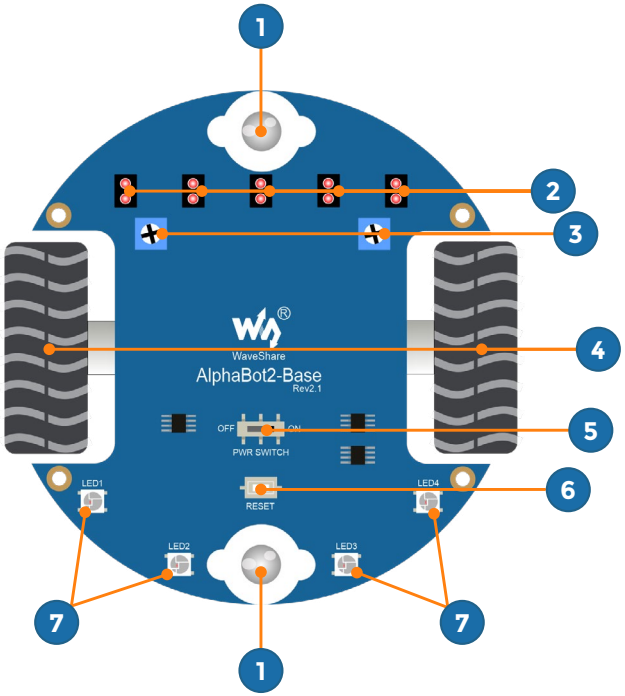
• Figure 1 : Lower base, frontside



Source : AlphaBot2-Ar - Waveshare Wiki

1 • AlphaBot2 control interface (female)
2 • Ultrasonic module interface
3 • Obstacle avoiding indicators
4 • Reflective infrared photoelectric sensor, for obstacle avoiding
5 • Micro gear motor reduction rate 1:30, 6V/600RPM
6 • Batteries holder for rechargeable Li-ion 14500 batteries. Beware, depending on Alphabot2 hardware revision, the +/- polarities can be reversed.
7 • Power indicator
8 • 5V USB battery charging port
9 • Battery charging indicator

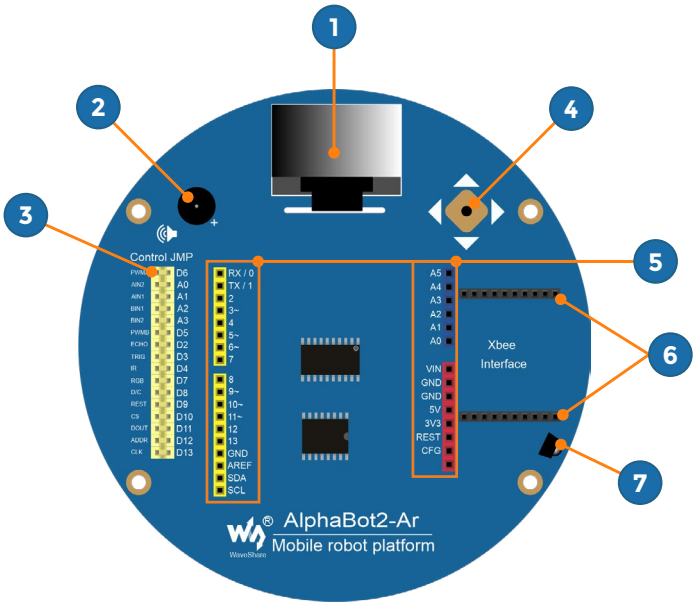
• Figure 2 : Lower base, backside



Source : AlphaBot2-Ar - Waveshare Wiki

1 • Omni-directional wheel
2 • Reflective infrared photoelectric sensor, for line tracking
3 • Potentiometer for adjusting obstacle avoiding range. You need to adjust them with a screw-driver to make the infrared sensors (figure 1, point 4) working.
4 • Rubber wheels diameter 42mm, width 19mm
5 • Power switch
6 • Reset switch
7 • Adressable RGB LEDs

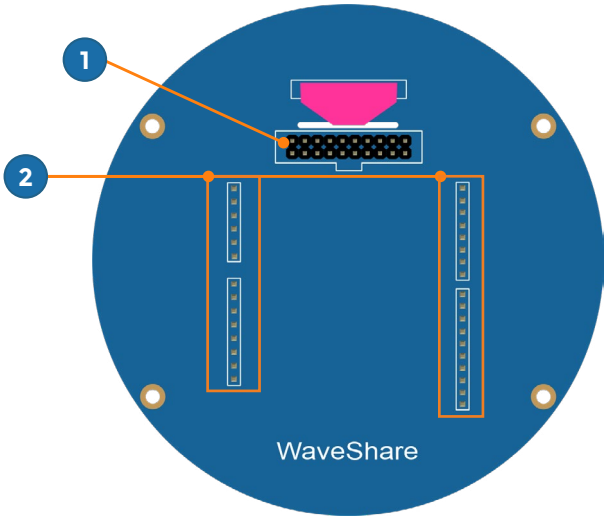
• Figure 3 : Upper base, frontside



Source : AlphaBot2-Ar - Waveshare Wiki

1 • OLED 128x64 display
2 • Buzzer
3 • Arduino peripheral jumpers
4 • Joystick
5 • Arduino expansion header (female)
6 • Xbee connector (female), not used in our kit.
7 • IR receiver (for remote control)

• Figure 4 : Upper base, backside

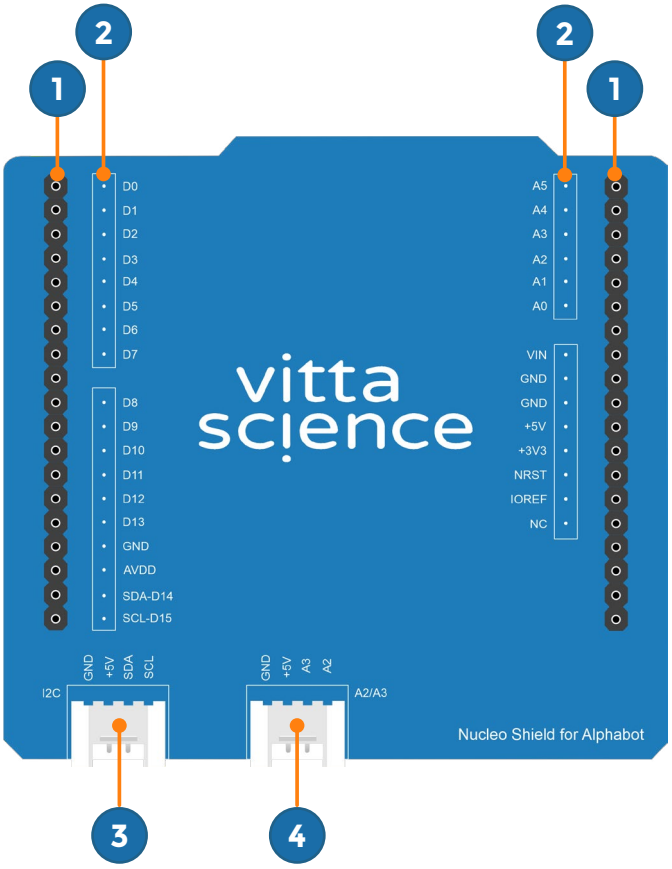


Source : AlphaBot2-Ar - Waveshare Wiki

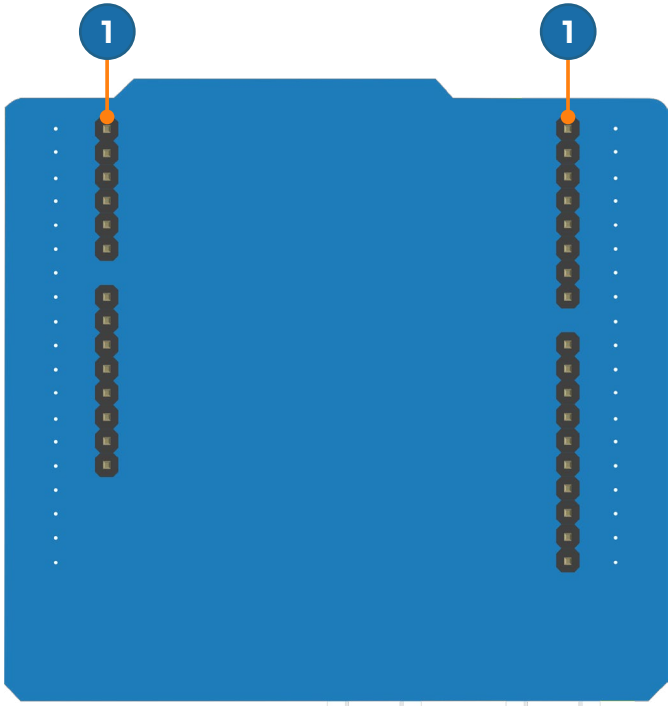
1 • AlphaBot2 control interface (male)
2 • Arduino expansion header (male)

Overview of Nucleo Shield for Alphabot

• **Figure 5 : Nucleo Shield for Alphabot, frontside**



• **Figure 6 : Nucleo Shield for Alphabot, backside**

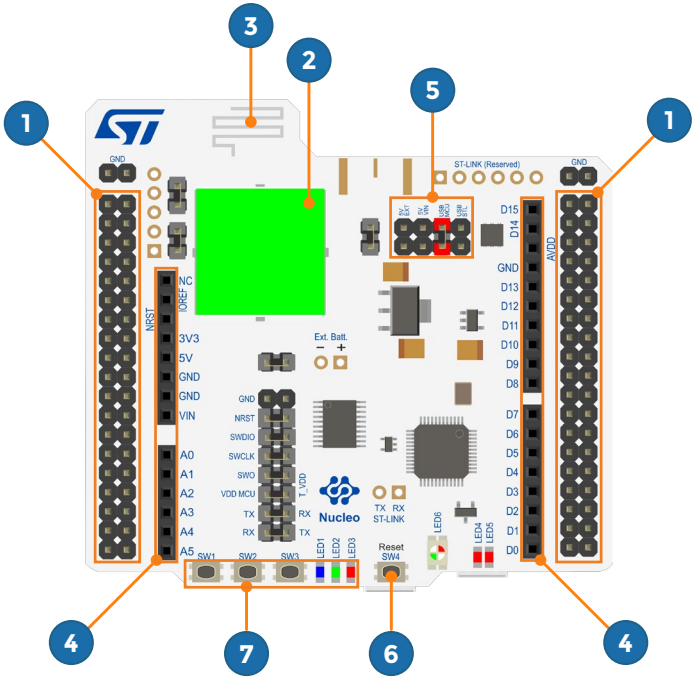


1 • Nucleo - to - Arduino bridge header (female)
2 • Arduino expansion header labels
3 • Grove connector to I2C bus
4 • Grove connector to analog input

1 • Arduino expansion header (male)

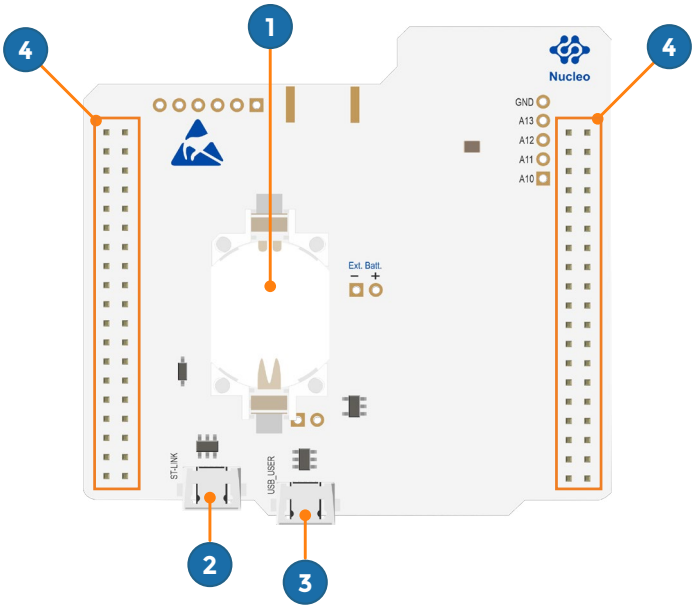
Overview of Nucleo-WB55 board

• **Figure 7 : Nucleo-WB55 frontside**



1 • ST-Morpho expansion header (male)
2 • STM32WB55 System-on-Chip embedding Bluetooth Low Energy radio
3 • Bluetooth Low Energy antenna
4 • Arduino expansion header (female)
5 • Power & programming setup header (male)
6 • Reset switch
7 • User switches (3) and user LEDs (3)

• **Figure 8 : Nucleo-WB55 backside**

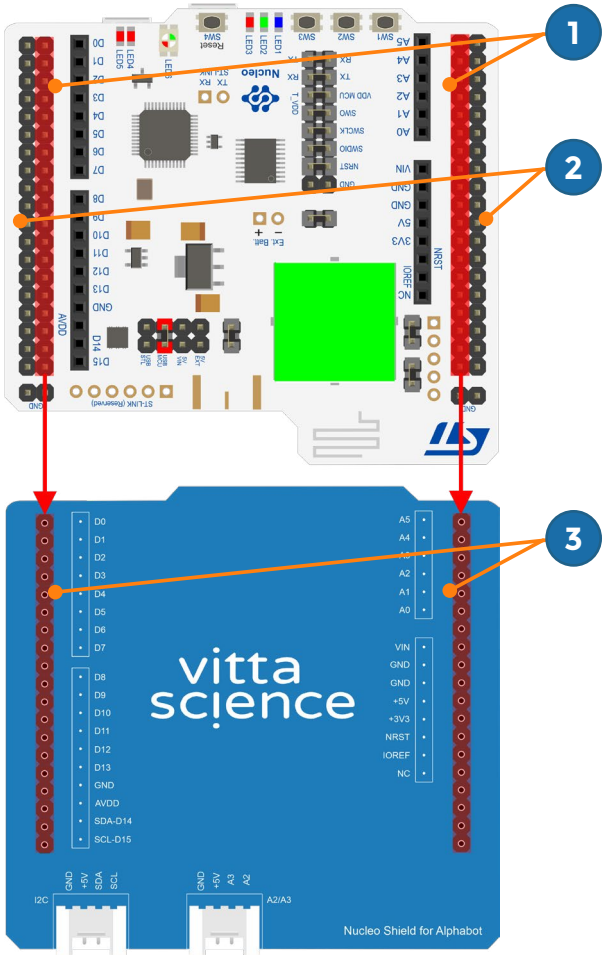


1 • CR2023 battery socket (depends on versions)
2 • Micro-USB connector to ST-LINK (firmware programming)
3 • Micro-USB connector to USB _ USER (link to MicroPython REPL)
4 • ST-MORPHO header (male)

Assembly instructions ⏱ according to the supervisor's estimations

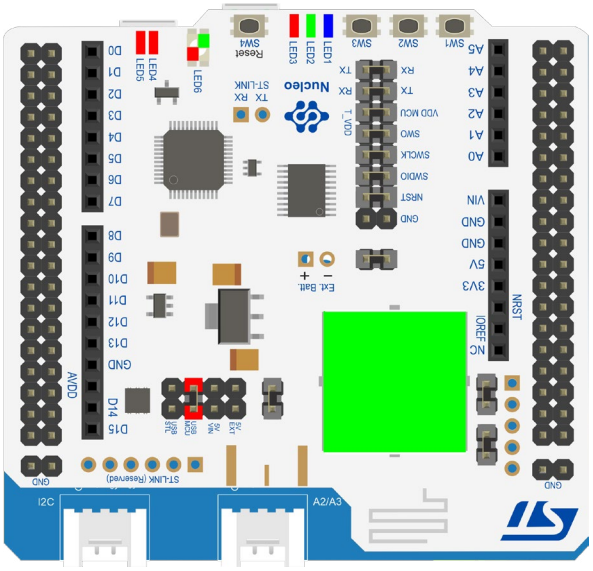
Connecting Nucleo-WB55 to Nucleo shield for Alphabot

• **Figure 9 : Nucleo-WB55 & Nucleo shield for Alphabot headers matching**



1 • Inner columns of ST-Morpho expansion header
2 • Outer columns of ST-Morpho expansion header
3 • Nucleo - to - Arduino bridge

• **Figure 10 : Nucleo-WB55 plugged on Nucleo shield for Alphabot (top view)**



• **First, plug Nucleo-WB55 board on top of Nucleo Shield for Alphabot.**

To achieve this, the two inner columns of Nucleo-WB55 ST-Morpho expansion header must fit inside the Nucleo-to-Arduino bridge header. This is illustrated on figure 9.

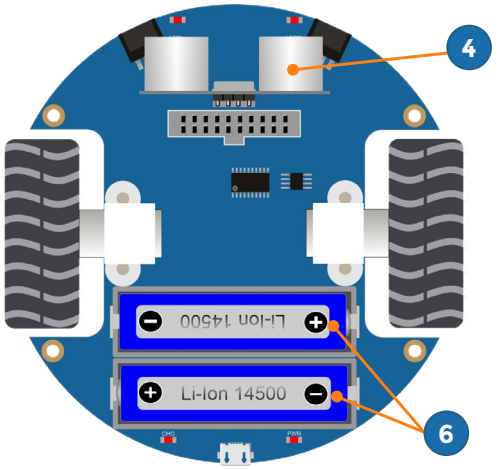
Doing that :

- Take care of the Nucleo-WB55 orientation
- Take care making no overlay mistake between male and female connectors during this operation. Check also that all teeth of inner columns of Morpho header are fully and firmly inserted inside Nucleo-to-Arduino bridge header.
- Notice that teeth from outer columns of Morpho header will be floating parallel to Nucleo-to-Arduino header.

If you completed this step properly, top view of the result should look like figure 10.

Alphabot base setup

• Figure 11 : Alphabot lower base setup (top view)



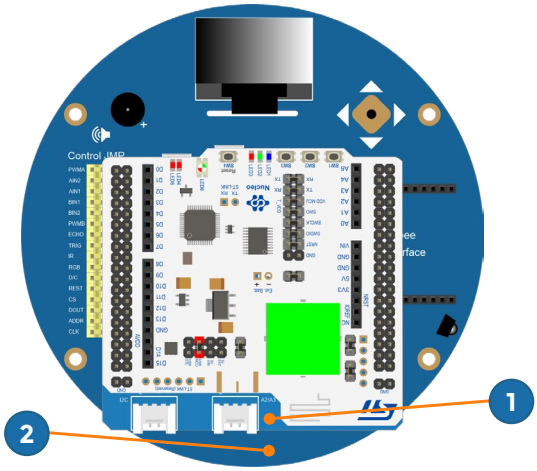
1 • Plugged ultrasonic sensor
2 • Plugged Li-Ion rechargeable 14500 batteries. Beware : polarities potentially inverted on your robot, check !

• Second, prepare the Alphabot lower Base

To achieve this, plug the two rechargeable Li-ion 14500 batteries inside batteries holder and the ultrasonic sensor inside its header. This is illustrated on figure 11.

Doing that :
• Take care using the right kind of batteries, i.e., rechargeable Li-ion 14500 batteries.
If you completed this step properly, top view of the result should look like figure 11.
Beware, AA batteries won't work !

• Figure 12 : Alphabot upper base setup (top view)



1 • Nucleo shield for Alphabot plugged on Arduino expansion header, on top of Alphabot upper base.
2 • Alphabot upper base (below)

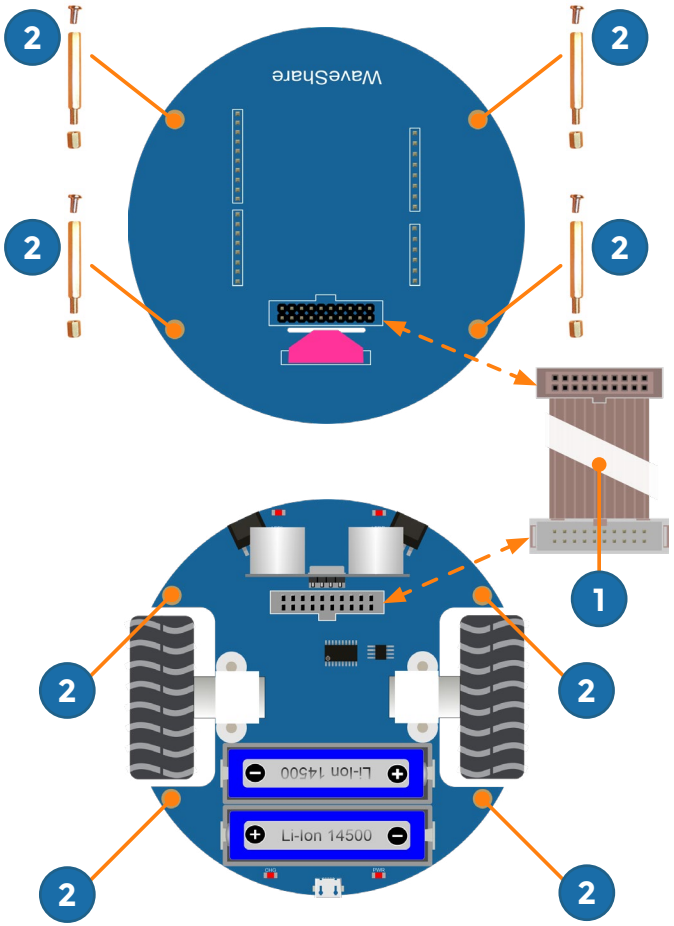
• Third, prepare the Alphabot upper base

To achieve this, plug the female Arduino connectors on the backside of the Nucleo shield for Alphabot inside the Arduino expansion header on the frontside of Alphabot upper base.

Doing that :
• Take care achieving the right overlay for connectors. Each male-connector tooth should fit inside its corresponding female plug and be fully and firmly inserted inside it.
If you completed this step properly, top view of the result should look like figure 12.

Connecting Alphabot lower-base to upper -base

• Figure 13 : Connecting lower-base to upper -base



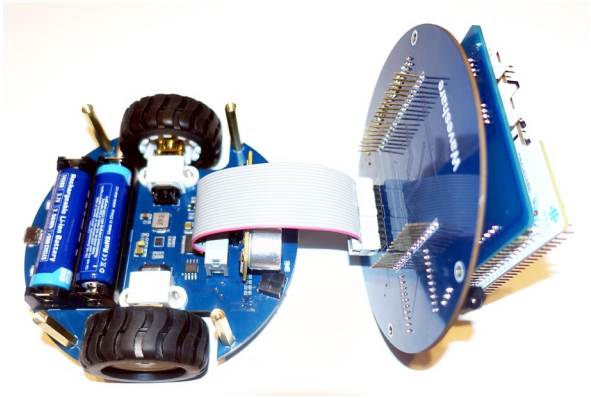
1 • Flexible connector map.
2 • Holes and spacer screws
3 • Holes for spacer screws

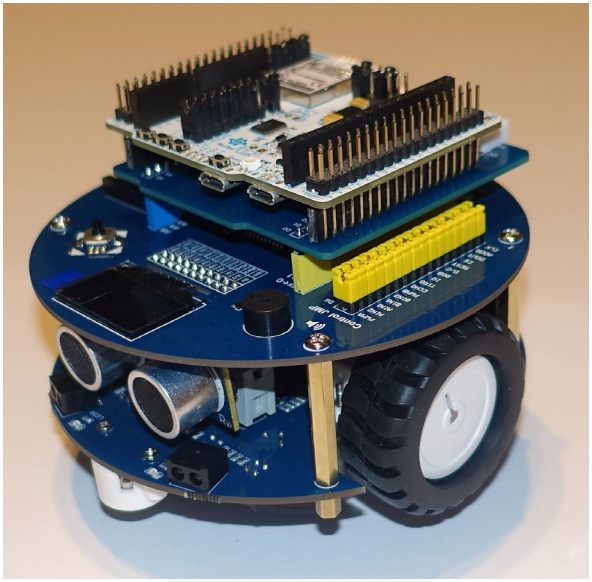
• Thourth, connect Alphabot layers

To achieve this,
1 • Plug the female end of flexible connector map to the AlphaBot2 control interface on the backside of the upper base.
2 • Plug the male end of flexible connector map to the AlphaBot2 control interface on the front side of lower base.
3 • Finish assembly of upper base and lower base putting the 4 spacer-screws in the corresponding holes.

Doing that :
• Take care pulling firmly the ends of flexible map inside both upper and lower base connectors.
Figure 13 is illustrating this step.

Pictures of assembled Martian robot





• Picture 1 : Connecting Alphabot lower-base to upper-base.

• Picture 2 : Assembled Martian robot.

• Last step, pictures of assembled robot

Picture 1 and picture 2 will give you a visual confirmation you did no mistake while following the previous assembly instructions.

Now, it's time to share the last technical points to use properly the robot !

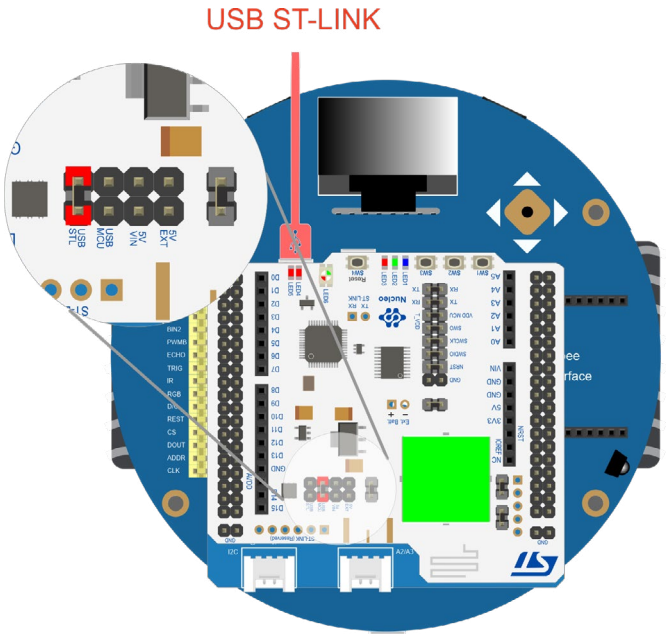
Programming the Martian robot

according to the supervisor's estimations

MicroPython firmware updating setup

1 • Put the power switch under the robot to « OFF »
2 • Put the jumper highlighted in red on below figure on the « USB STL » position.
3 • Connect your board to your computer using « USB STL » connector of the Nucleo-WB55 board.
4 • From your computer, drag and drop the new firmware on the virtual USB drive associated to your Nucleo-WB55 board.
5 • Wait for the copy operation to complete.

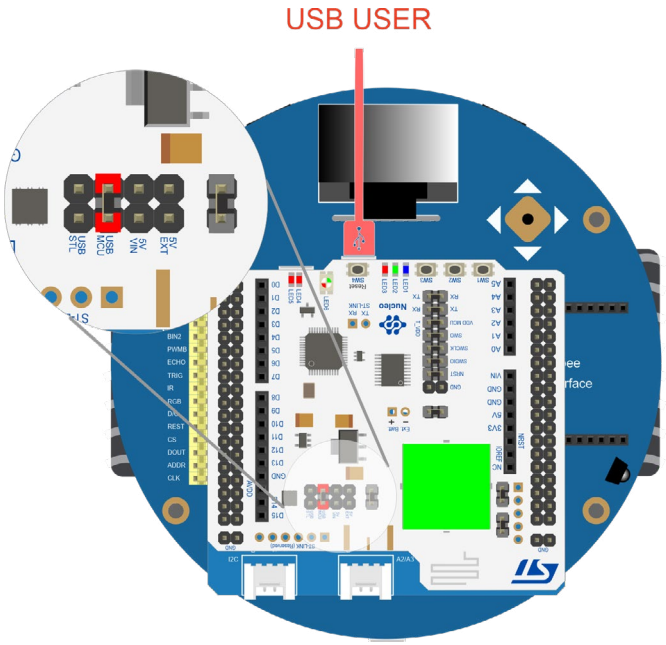
To operate correctly, the robot need the firmware release 1.17 or a later one. If you need to update the firmware, please download it exclusively from Vittascience web site.



MicroPython programming setup

1 • Put the power switch under the robot to « OFF »
2 • Put the jumper highlighted in red on below figure on the « USB MCU » position.
3 • Connect your board to your computer using « USB USER » connector of the Nucleo-WB55 board.
4 • Connect the USB cable to your computer and start Vittascience programming interface.

You can now connect to the Vittascience interface and upload your programs inside the Nucleo-WB55 driving the robot.

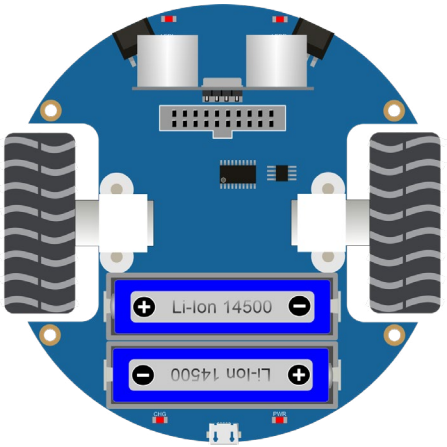


Operating the Martian robot

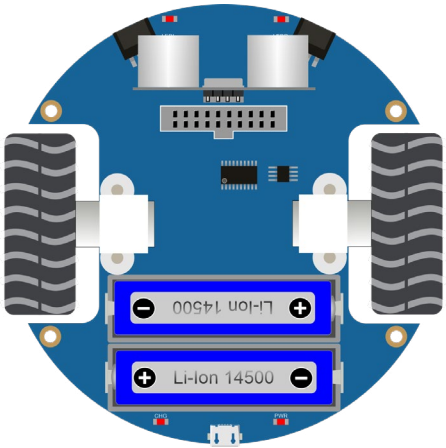
according to the supervisor's estimations

Charging the robot batteries

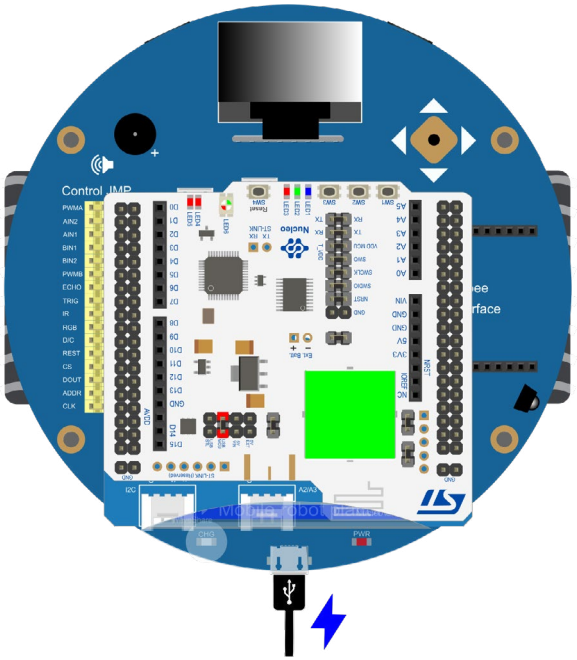
1 • If the orientation of batteries holder on your robot is the one shown on figure 16-a, put the power switch under the robot to « OFF ».
2 • If the orientation of batteries holder on your robot is the one shown on figure 16-b, put the power switch under the robot to « ON ».
Put the power switch under the robot to « OFF ».
3 • Plug only one USB cable to your robot, in the 5V USB battery charging port of the Alphabot base.
4 • The CHG LED should light until the batteries are fully charged. If the LED is flashing quickly, please verify the USB cable and the batteries are well plugged.
5 • When the batteries are full, the CHG LED is off.



• Figure 16-b



• Figure 16-a



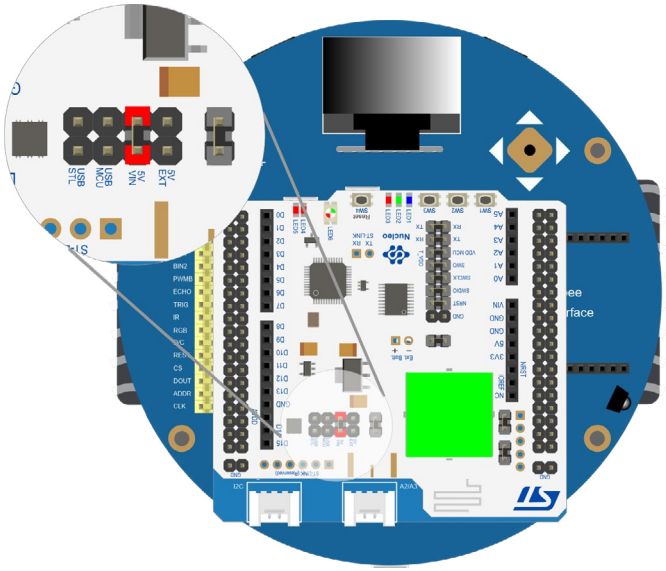
• Figure 17

Starting the robot

1 • Unplug any USB cable connected to your robot.
2 • Put the jumper highlighted in red on below figure on the «5V VIN» position. Now, the Nucleo-WB55 board will draw power from the two batteries of the Alphabot base.
3 • Put the power switch under the robot to « ON », the PWR LED on its base should light.

If the batteries of the Alphabot are charged enough, the PWR LED of the base of the robot will light and the robot will start executing the last program you uploaded inside the Nucleo-WB55 using Vittascience programming interface.

If the robot behaviour is not the one you expected, check your program and/or consider charging the batteries.



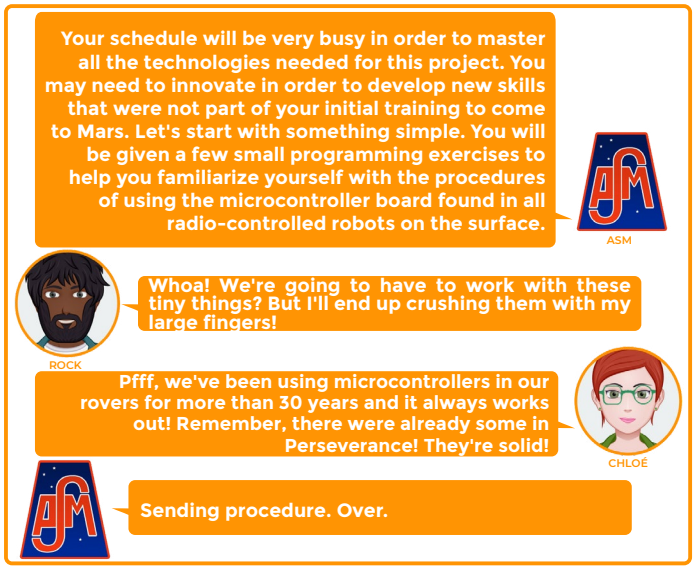
Task • 1 ⏱ according to the supervisor's estimations

Discovering and using the equipment

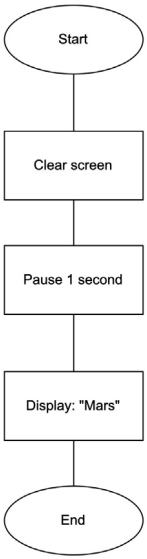
Discovering the controller board of the future mini rover (very simple task to master the program transfer and the interface of the Vittascience platform.)

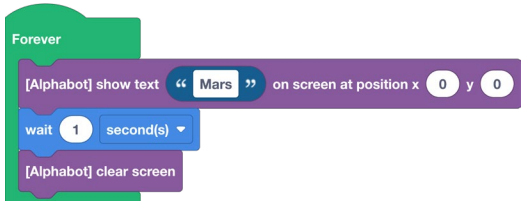
• Task 1-1: Display text on the OLED screen

To start off this first task we will display the text "Mars" on the OLED screen of the robot.



Here are, respectively, the flow chart, the code for programming by bloc and the Python code.






```
1 import machine
2 from stm32_ssd1306 import SSD1306, SSD1306_I2C
3 import utime
4
5 oled = SSD1306_I2C(128, 64, machine.I2C(1))
6
7 while True:
8   oled.text('Mars ', 0, 0)
9   oled.show()
10  utime.sleep(1)
11  oled.fill(0)
12  oled.show()
```

In the Vittascience programming interface, you can choose to program either in Python or in block code. As a first step and according to your level, we advise you to use the block code which is easier to begin with.

The blocks are in the left menu, as a start, only the following two will be needed.

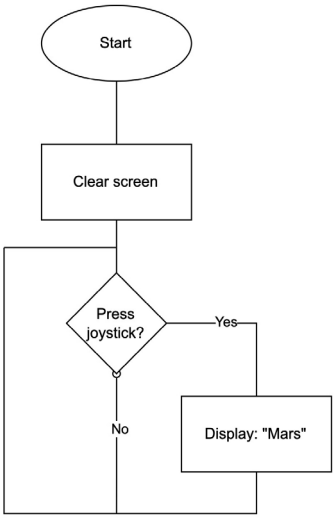
Robots | ⇄ Inputs/Outputs

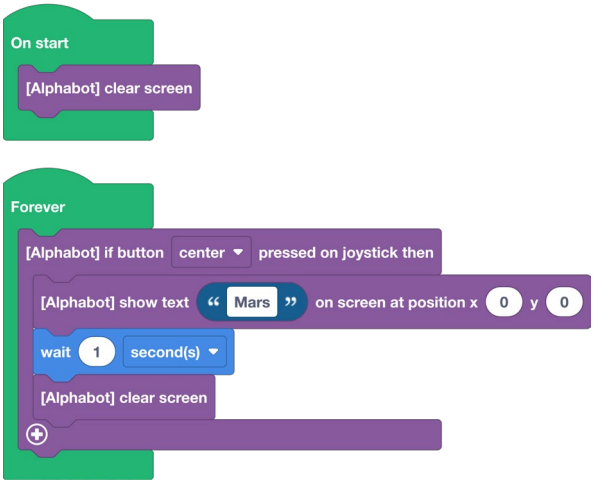
• Task 1-2 : Using the joystick

Now that we have understood how the basic functions of the board work to display a text, we will interact with the screen using the joystick, as a first step.

We are going to keep the display of "Mars" as the basic code, but now it will be displayed only when we press the joystick of the robot. For this task we will need new blocks which are in the following menus:

Robots | ⇄ Inputs/Outputs | Loops



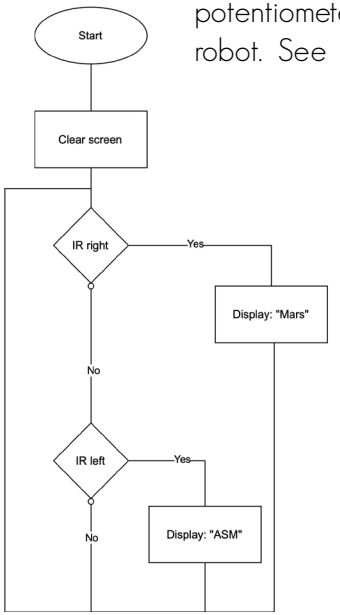



```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  joystickButton = alphabot.getJoystickValue()
  if joystickButton == "center":
    oled.text('Mars ', 0, 0)
    oled.show()
    utime.sleep(1)
    oled.fill(0)
    oled.show()
```

• Task 1-3: Use of the front infrared sensors

Instead of the joystick, which is only a basic pressure sensor (press or not) we are going to use the frontal infrared sensors of the robot.
For this task, we will complete the previous code with the following function: "if I pass my hand in front of the right sensor, the OLED screen will display "Mars" for 1 second. If I pass my hand in front of the left sensor, then it displays "MSA"." (Again, the information received will be basic "I am receiving or not")
Here are respectively, the flowchart, the block programming code and finally the Python code.
As we can see, in the code there is an "erase screen" block, in the startup loop, it makes sure that the screen is empty when the robot starts up.

Caution : If the infrared sensors do not seem to be working, adjust their sensitivity using the potentiometers on the underside of the robot. See page 18, figure 2, point 3.



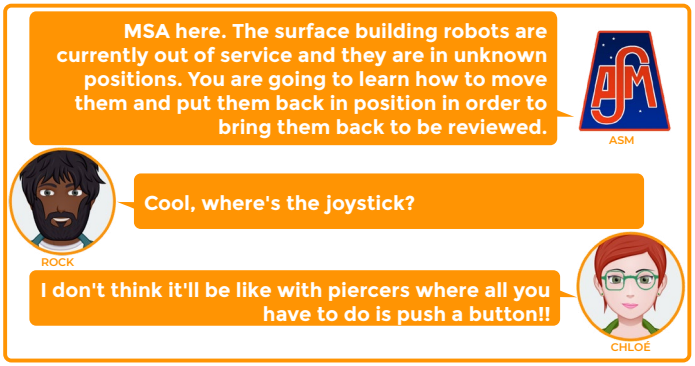

```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  detection = alphabot.readInfrared()
  if detection == alphabot.RIGHT_OBSTACLE:
    oled.text('Mars ', 0, 0)
    oled.show()
    utime.sleep(1)
    oled.fill(0)
    oled.show()
  detection = alphabot.readInfrared()
  if detection == alphabot.LEFT_OBSTACLE:
    oled.text('ASM', 0, 0)
    oled.show()
    utime.sleep(1)
    oled.fill(0)
    oled.show()
```

Task • 2 according to the supervisor's estimations

Controlling the engines

• Task 2-1: Controlling the engines

It is about time we moved our little robot. For now we are only going to make simple movements like move forward/ turn.



There are many ways to control the wheels of the robot according to the 3 code blocks:

Bloc 1 : Allows you to control each engine, whether it is left or right, independently. It also allows you to control the rotation direction and the speed of rotation.

Bloc 2 : Allows you to control the rotation direction and rotation speed of the two left and right engines simultaneously.

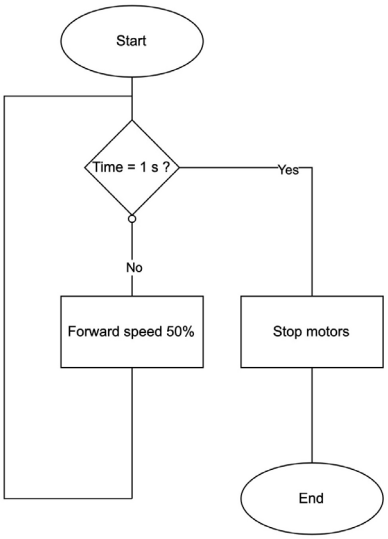
Bloc 3 : Allows you to stop the engines independently or simultaneously.

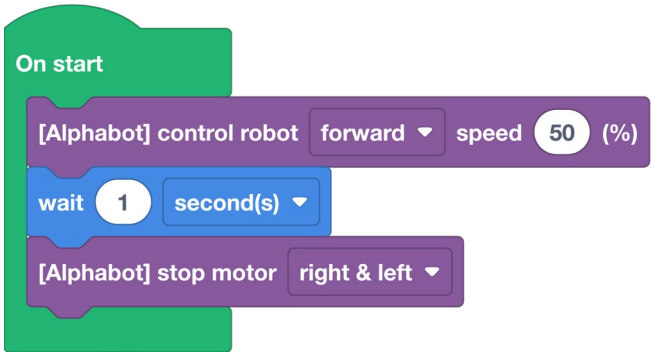
Bloc 4 : Allows you to activate the engine that lets the robot turn to the left or to the right and to control the power of the engine.

For our first program we are just going to make the robot move forward for 1 second.

To do so we will need blocks which can be found in:

Here are respectively, the flowchart, the block programming code and the Python code.






```python
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime

alphabot = AlphaBot_v2()

alphabot.moveForward(50)
utime.sleep(1)
alphabot.stop()

while True:
  pass
```

• Task 2-2 : Learn to turn

A robot that moves forward is good, but also dangerous. We'll have to teach it to turn and for that we'll use the independent control of each engine.

For turning, we have two methods to choose from: the tank method or the car method. The tank turns its wheels in the opposite direction on each side which allows it to turn on the spot. The car, with its differential system, usually has one wheel spinning faster than the other which allows it to turn.

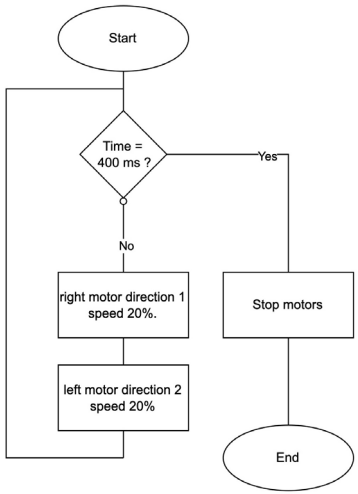
Each technique has its own advantages and disadvantages depending on its use. The opposite direction technique allows the robot to turn faster but at the expense of the overall speed of movement. The other method presents the same problem but the other way around of course.

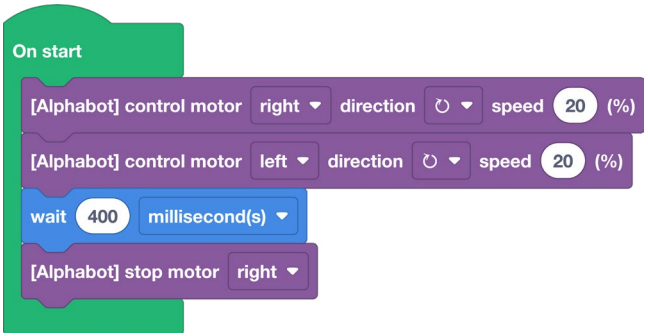
In our situation we are going to move in Mars, a hostile environment, so caution and maneuverability is more important. So we will start by using the tank method with a 90 ° turn.

To do so we will need blocks that can be found in:

Robots ⇄ Inputs/Outputs

Here is in order the flowchart, then the programming code by block and finally the Python code.






```python
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime

alphabot = AlphaBot_v2()

alphabot.setMotors(right=20)
alphabot.setMotors(left=-20)
utime.sleep_ms(400)
alphabot.stop()

while True:
  pass
```

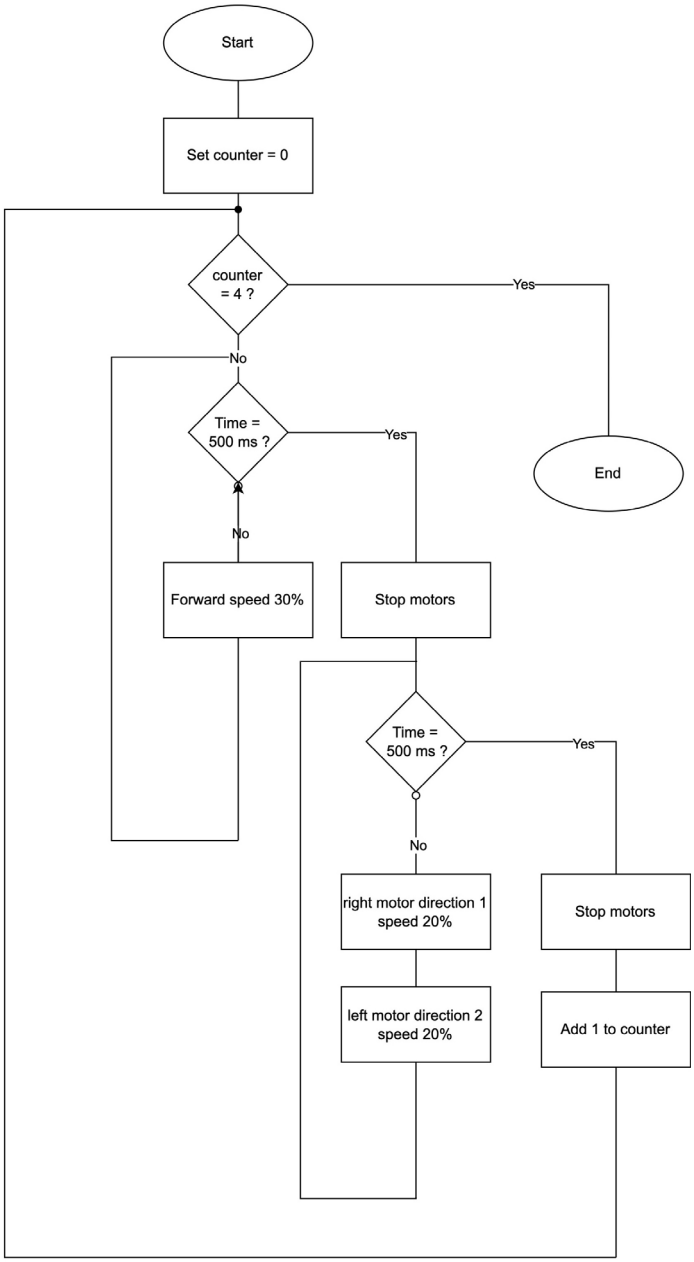
The program will not be perfect; we will have to adjust the time in order to obtain a 90° angle. You can do it by trial and error or use a proportionality table. Do not hesitate to reduce the speed in order to better control the direction.
We now have full control of our robot and can move it wherever we want.

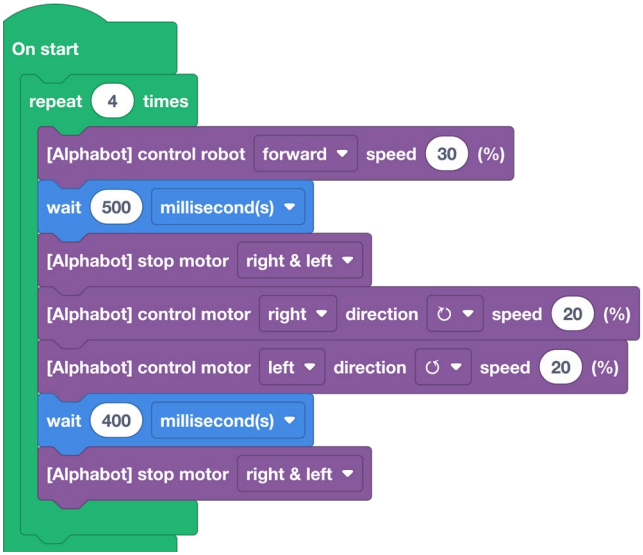
• Task 2-3 : Learn to make a trajectory

We know how to move forward/backward in a straight line and how to turn in a 90° angle. Demonstrate your skills by making the robot move in a square.
To do so, we will need the block that is in:

Robots ⇄ Inputs/Outputs

Caution : The flowchart becomes more complicated to write because of the loop that will count how many times we have done a series of actions. A counter variable will therefore appear.



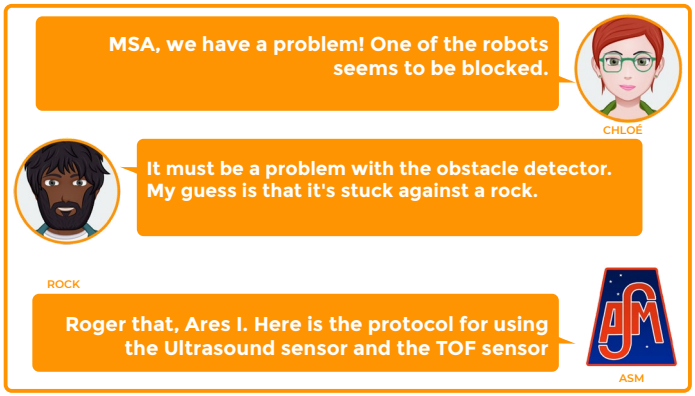



```
1  import machine
2  from stm32_alphabot_v2 import AlphaBot_v2
3  import utime
4
5  alphabot = AlphaBot_v2()
6
7  for count in range(4):
8    alphabot.moveForward(30)
9    utime.sleep_ms(500)
10   alphabot.stop()
11   alphabot.setMotors(right=20)
12   alphabot.setMotors(left=-20)
13   utime.sleep_ms(400)
14   alphabot.stop()
15
16 while True:
17   pass
```

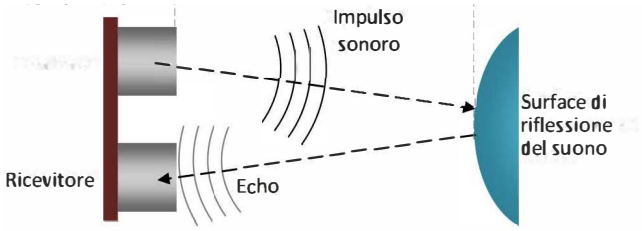
Task • 3 ⏱ according to the supervisor's estimation

Testing the obstacle detector

• Task 3-1: Discovering the ultrasonic sensor



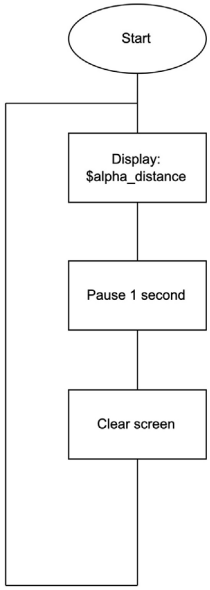
The ultrasonic sensor can detect an obstacle in front of it and even measure a distance.



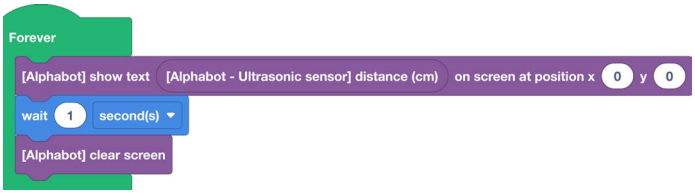
On the one hand, there is a transmitter that sends out a sound wave. When the wave hits an obstacle, the obstacle sends the same wave back to the transmitter. Then a calculator "counts" how long the sound wave took to go and come back. Knowing the speed at which sound travels in the air, one can deduce the distance (V=d/t). We deduce d.

For this mission we are going to be constantly displaying the distance of an object on the OLED screen.

To do so we will need blocks which are in:

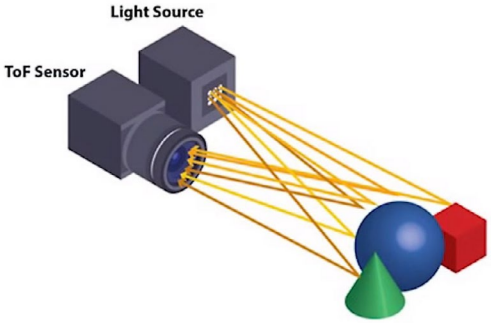



```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  oled.text(str(alphabot.readUltrasonicDistance()), 0, 0)
  oled.show()
  utime.sleep(1)
  oled.fill(0)
  oled.show()
```



• Task 3-2: Discovering Time of Flight (ToF)

The Time of Flight sensor allows you to detect an obstacle in front of it and even to measure a distance.

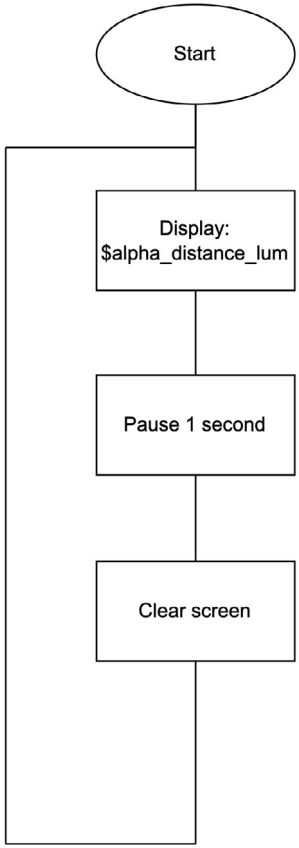


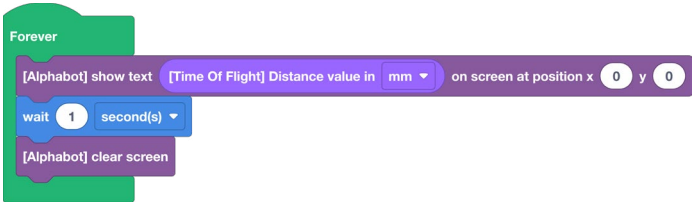
There is a transmitter that sends a laser beam. When the beam hits an obstacle, the obstacle sends it back. Then a calculator "counts" how long the laser pulse took to go and come back. Knowing the speed at which a laser travels in the air (speed of light), we can deduce the distance (V=d/t).

For this task, we will be constantly displaying the distance of an object on the OLED screen. The Time of Flight sensor must be connected to the I2C socket of the shield

To do so we will need blocks which are in:

Robots | Inputs/Outputs | Loops

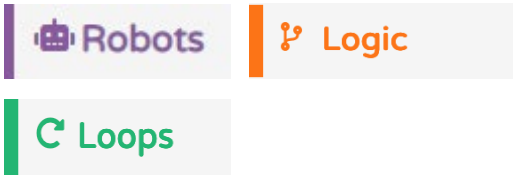




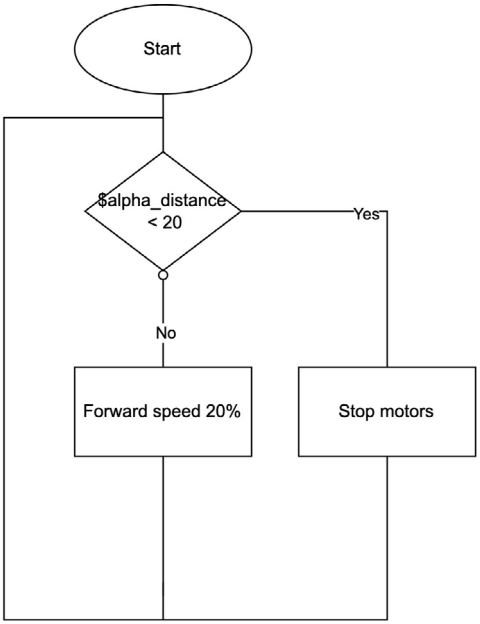
• Task 3-3 : Distance sensors and engines

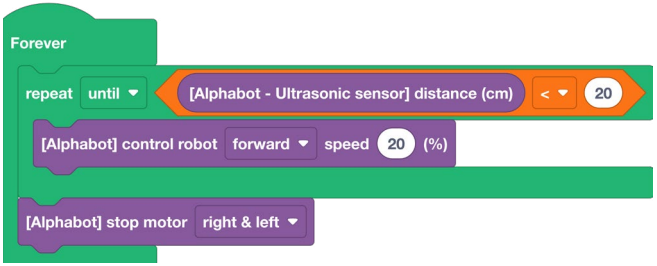
We are going to use this ultrasonic sensor to move the robot. The robot will have to move until the distance between the robot and a wall is the right distance, for example 30 cm.

To do so we will need the blocks that are in:



In this task we are going to use the logic comparators, which allow us to check if a piece of information is true or false. Once this verification is done, we can start this or that action. In our case, the question will be: Is the distance between my robot and the wall less than 20 cm?






```
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()

oled.fill(0)
oled.show()

while True:
  while not alphabot.readUltrasonicDistance() < 20:
    alphabot.moveForward(20)
  alphabot.stop()
```

If you measure the distance obtained after the robot has moved, there may be a slight difference. This is due to two factors:

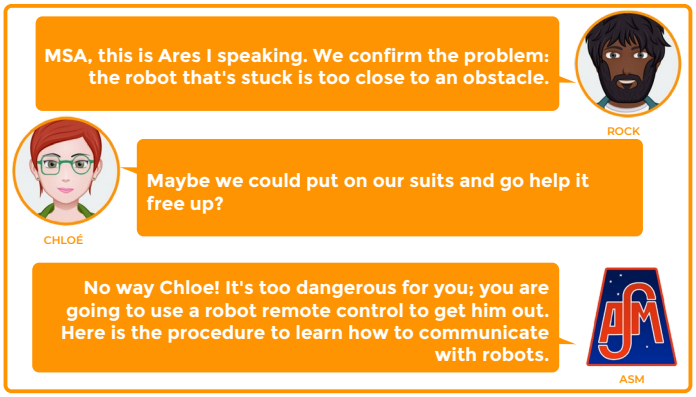
- The first factor is due to human error, we forget that we should not measure the distance from the wheels onward but rather from the distance sensor because it takes the measurement.
- The second factor is due to the robot's inertia and the speed of calculation. The robot cannot stop instantly and the higher the speed, the greater the error due to braking. Also, the computation time of the board, which is negligible most of the time, can cause a slight deviation of a few mm if the speed is high.

It is entirely possible to use the ToF sensor instead of the Ultrasonic sensor, the program will be identical except for the name of the sensor used.

Task • 4

according to the supervisor's estimations

Real time piloting



In order to control the small robot we will use the dedicated remote control.

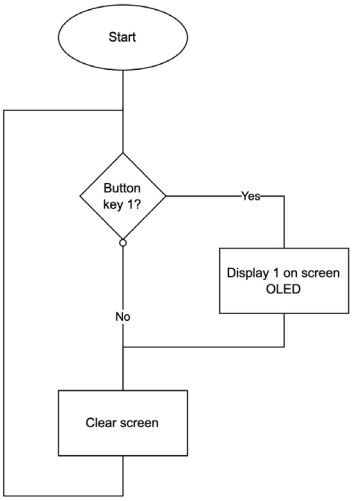
The idea is that the robot receives information and reacts according to the information it receives.

• Task 4-1: The remote control

In the first task, we will be displaying a piece of information on the board according to the key of the remote control. For example, if we press the key 1 the screen should display 1.

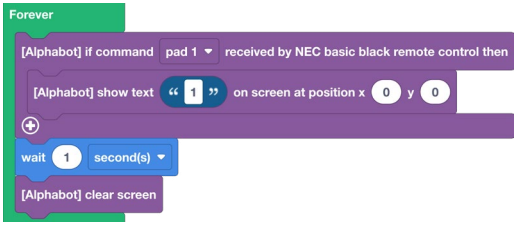
To do so we will need blocks that are in:

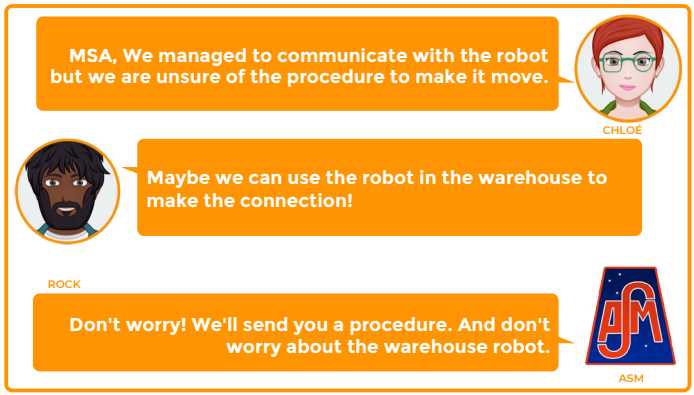
Robots | Loops




```python
import machine
from stm32_ssd1306 import SSD1306, SSD1306_I2C
from stm32_alphabot_v2 import AlphaBot_v2
import utime
from stm32_nec import NEC_8, NEC_16
import gc

oled = SSD1306_I2C(128, 64, machine.I2C(1))
alphabot = AlphaBot_v2()
ir_current_remote_code = None

def remoteNEC_basicBlack_getButton(hexCode):
  if hexCode == 0x0c: return "1"
  elif hexCode == 0x18: return "2"
  elif hexCode == 0x5e: return "3"
  elif hexCode == 0x08: return "4"
  elif hexCode == 0x1c: return "5"
  elif hexCode == 0x5a: return "6"
  elif hexCode == 0x42: return "7"
  elif hexCode == 0x52: return "8"
  elif hexCode == 0x4a: return "9"
  elif hexCode == 0x16: return "0"
  elif hexCode == 0x40: return "up"
  elif hexCode == 0x19: return "down"
  elif hexCode == 0x07: return "left"
  elif hexCode == 0x09: return "right"
  elif hexCode == 0x15: return "enter_save"
  elif hexCode == 0x0d: return "back"
  elif hexCode == 0x45: return "volMinus"
  elif hexCode == 0x47: return "volPlus"
```

```
  elif hexCode == 0x46: return "play_pause"
  elif hexCode == 0x44: return "setup"
  elif hexCode == 0x43: return "stop_mode"
  else: return "NEC remote code error"

def remoteNEC_callback(data, addr, ctrl):
  global ir_current_remote_code
  if data < 0:  # NEC protocol sends repeat codes.
    print('Repeat code.')
  else:
    print('Data {:02x} Addr {:04x} Ctrl {:02x}'.format(data, addr, ctrl))
    ir_current_remote_code = remoteNEC_basicBlack_getButton(data)

classes = (NEC_8, NEC_16)
ir_remote = classes[0](alphabot.pin_IR, remoteNEC_callback)

oled.fill(0)
oled.show()

while True:
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "1":
    oled.text('1', 0, 0)
    oled.show()
  utime.sleep(1)
  oled.fill(0)
  oled.show()
```

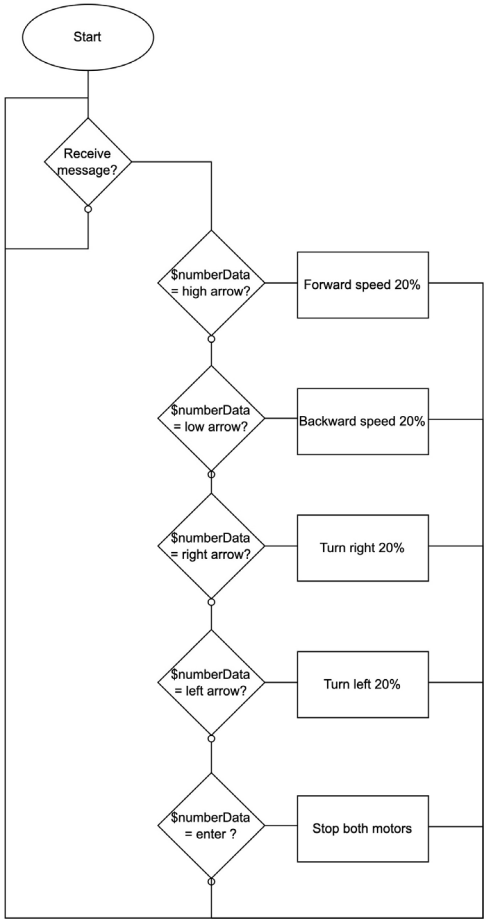


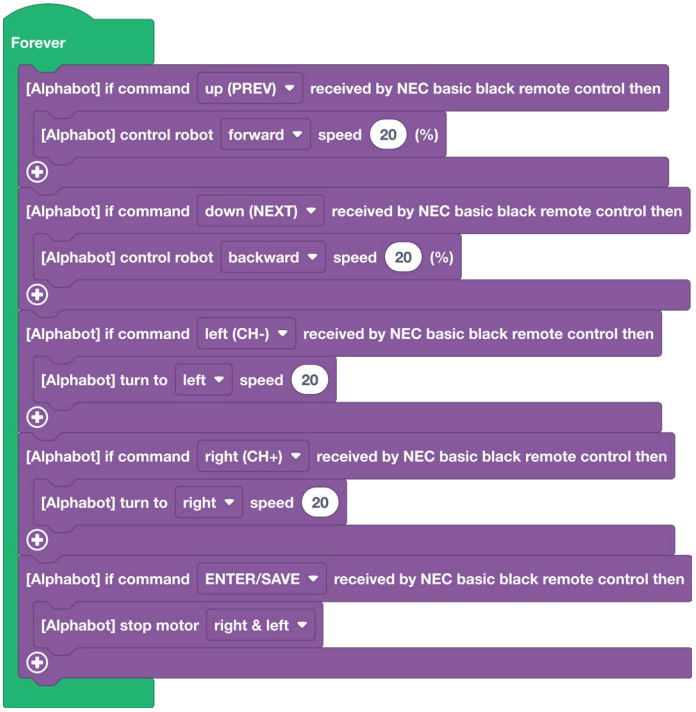


• Task 4-2: Testing in real "Martian" conditions

We are now going to drive our little robot in the warehouse. To do so we will use the programs we made earlier.

The goal is to move the robot forward with the top arrow, turn it with the right and left arrows, move it backwards with the bottom arrow and stop it with the Enter button on the remote control.






```
while True:
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "up":
    alphabot.moveForward(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "down":
    alphabot.moveBackward(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "left":
    alphabot.turnLeft(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "right":
    alphabot.turnRight(20)
  utime.sleep_ms(150)
  gc.collect()
  if ir_current_remote_code == "enter_save":
    alphabot.stop()
```

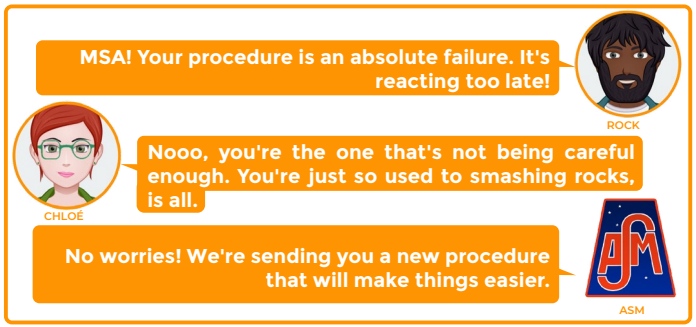
The Python code is intentionally incomplete, the beginning of the program is identical to the previous one and we can only find the function calls here.

After this simulation, we realize that it is very complex to control the robot with the remote control; a more efficient solution is to program the robot to make its decisions autonomously according to its environment.

With a simple command, the robot will have to adapt to fully accomplish the task it was given. For example, go 100 meters north and it will have to connect the meeting point which is located 100 meters to the north thanks to its programming.

Task • 5 ⏱ according to the supervisor's estimation

Moving in square mode



• Task 5-1: Moving in square mode

For this task we will start programming in square mode. The robot will make small movements as if it were on a chessboard or a checkers board. This will simplify the programming to start with and will allow us to tell the robot for example to move 3 squares forward and 2 to the right.

To begin with, we will perform the previous tasks with the 90° rotation, which will be perfect for moving on a square, and then we will measure how much time it takes to move straight ahead by one square.

We are going to set the size of the square to 10 cm, which allows us to make a grid of 4 squares by 3 squares on an A3 sheet.
The task code we will use is that of the 2-1 and 2-2, as well as the one for the task 4-2.

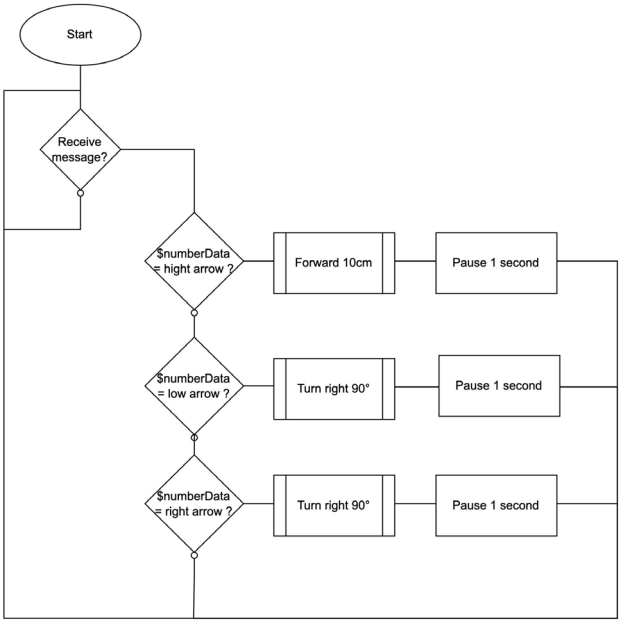
We will avoid re-coding everything. So we are going to use the code we've already made and we will improve it, as is often the case in computer science. To do so, we will use the following functions:

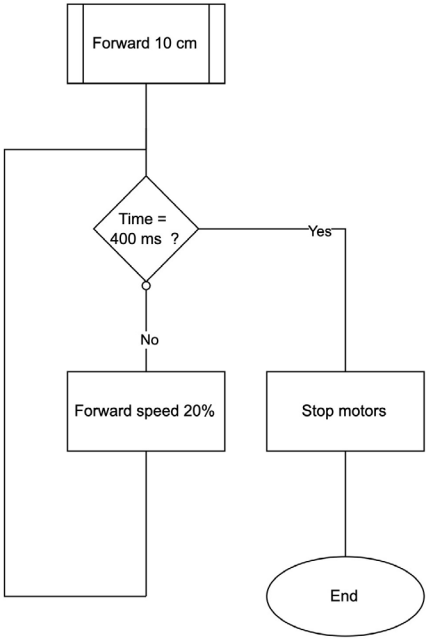
To proceed, we are going to create three functions: one to move forward, one to turn right and one to turn left.

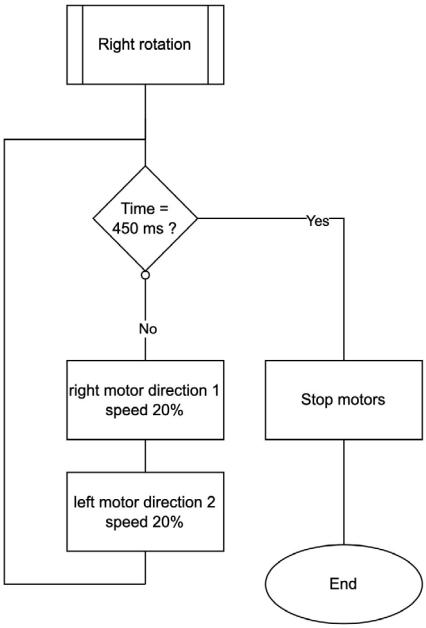
There are no changes from task 2-2 except that instead of putting the blocks in the "Repeat indefinitely" or "On startup" loop, we will put them in a function block to which we will give a distinguishable name so as not to get lost in the code.

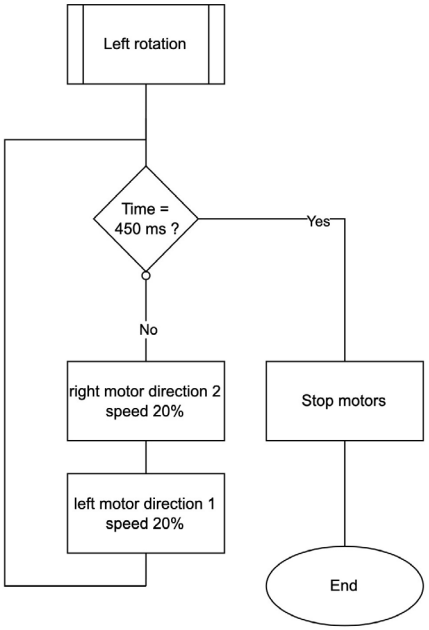
We will add a condition block so that the robot executes the function when the right button on the remote control is pressed, and then we will add a second of delay in order to have the time to press a key on the remote control in a dead zone (unaffected key).

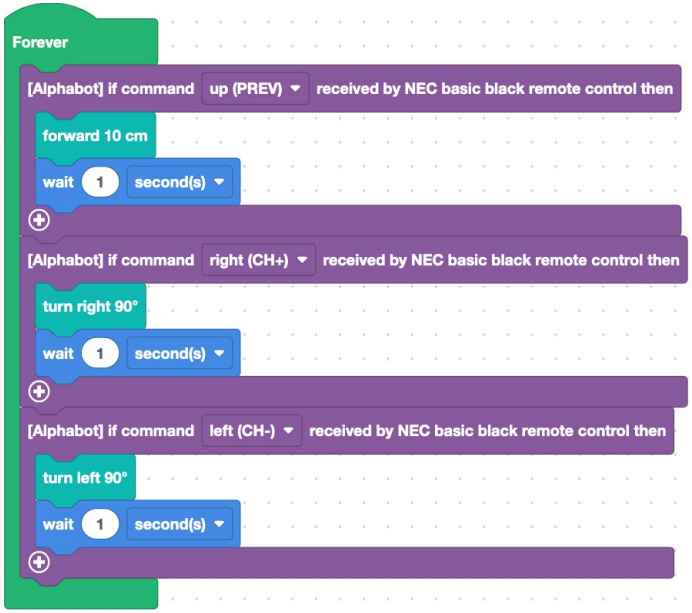
This gives us the flowcharts shown on the following pages:

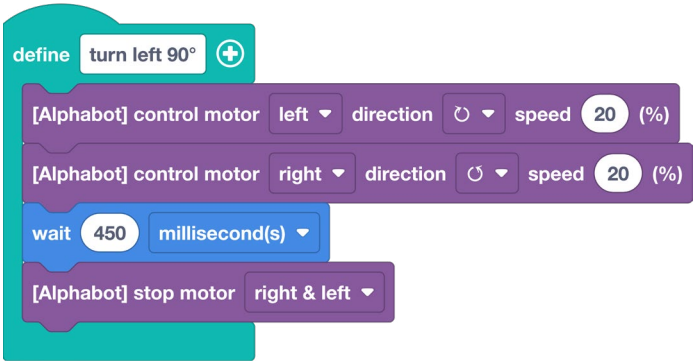

Start
Receive message?
$numberData = hight arrow ?
Forward 10cm
Pause 1 second
$numberData = low arrow ?
Turn right 90°
Pause 1 second
$numberData = right arrow ?
Turn right 90°
Pause 1 second

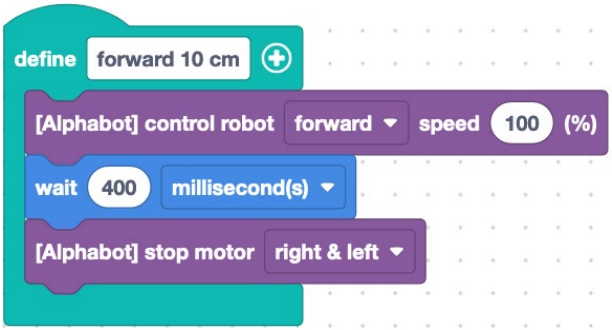

Forward 10 cm
Time = 400 ms ?
Yes
No
Forward speed 20%
Stop motors
End

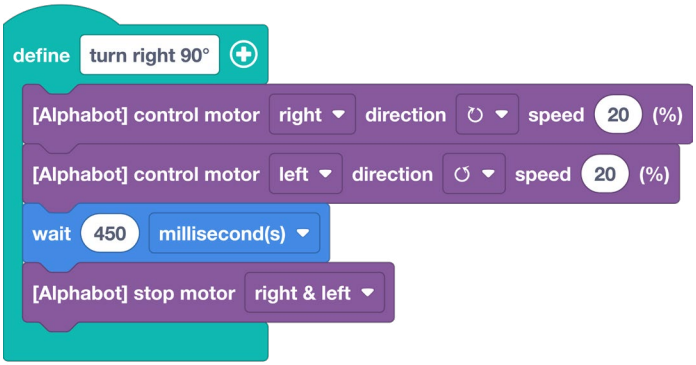

Right rotation
Time = 450 ms ?
Yes
No
right motor direction 1 speed 20%
left motor direction 2 speed 20%
Stop motors
End


Left rotation
Time = 450 ms ?
Yes
No
right motor direction 2 speed 20%
left motor direction 1 speed 20%
Stop motors
End


Forever
[Alphabot] if command up (PREV) received by NEC basic black remote control then
forward 10 cm
wait 1 second(s)
[Alphabot] if command right (CH+) received by NEC basic black remote control then
turn right 90°
wait 1 second(s)
[Alphabot] if command left (CH-) received by NEC basic black remote control then
turn left 90°
wait 1 second(s)


define turn left 90°
[Alphabot] control motor left direction speed 20 (%)
[Alphabot] control motor right direction speed 20 (%)
wait 450 millisecond(s)
[Alphabot] stop motor right & left


define forward 10 cm
[Alphabot] control robot forward speed 100 (%)
wait 400 millisecond(s)
[Alphabot] stop motor right & left


define turn right 90°
[Alphabot] control motor right direction speed 20 (%)
[Alphabot] control motor left direction speed 20 (%)
wait 450 millisecond(s)
[Alphabot] stop motor right & left

```
45 def tourner_gauche_90_C2_B0():
46   alphabot.setMotors(left=20)
47   alphabot.setMotors(right=-20)
48   utime.sleep_ms(450)
49   alphabot.stop()
50
51 def tourner__droite_90_C2_B0():
52   alphabot.setMotors(right=20)
53   alphabot.setMotors(left=-20)
54   utime.sleep_ms(450)
55   alphabot.stop()
56
57 def avancer_10cm():
58   alphabot.moveForward(20)
59   utime.sleep_ms(400)
60   alphabot.stop()
```

```
62 while True:
63   utime.sleep_ms(150)
64   gc.collect()
65   if ir_current_remote_code == "up":
66     avancer_10cm()
67     utime.sleep(1)
68   utime.sleep_ms(150)
69   gc.collect()
70   if ir_current_remote_code == "right":
71     tourner__droite_90_C2_B0()
72     utime.sleep(1)
73   utime.sleep_ms(150)
74   gc.collect()
75   if ir_current_remote_code == "left":
76     tourner_gauche_90_C2_B0()
77     utime.sleep(1)
78
```

Now that we have our functions for programming, the task gets easier, we just need to call the function in the main program and it will execute itself in full.

As you can see, this considerably simplifies the program, no need to make lines and lines of coding with the principle of functions anymore.

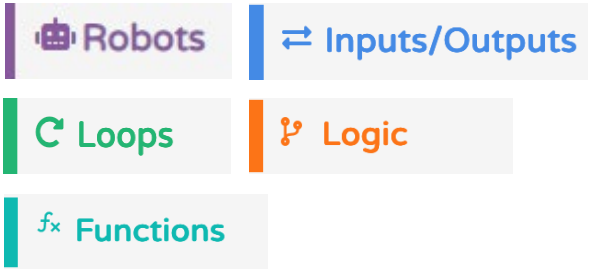
• Task 5-2: Detecting an obstacle before moving

In this task we will add the elements of the third task. As moving without looking where you are going can, at some point, cause problems. In everyday life and for humans, the eyes are the main organs of obstacle detection. Indeed, our 3-dimensional vision allows us to spot obstacles and to evaluate their distance. In our case, the ultrasonic sensor or Time of Flight will play this role.

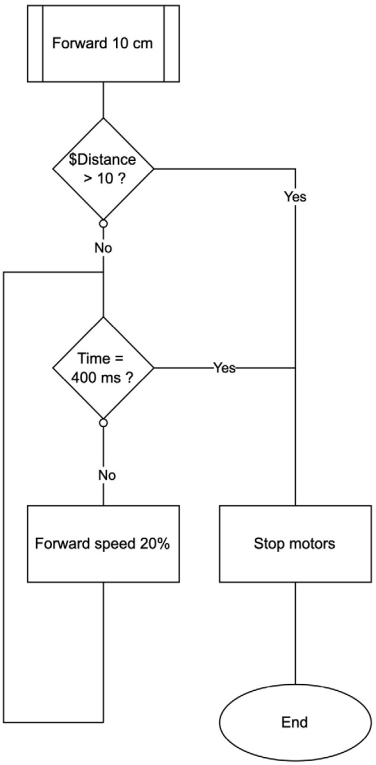
The goal of this task will be to move the robot on the checkerboard and check if the square it has to move to is free before each move. And if it is not the case, the robot should not move onto the blocked square and instead display a cross on the board.

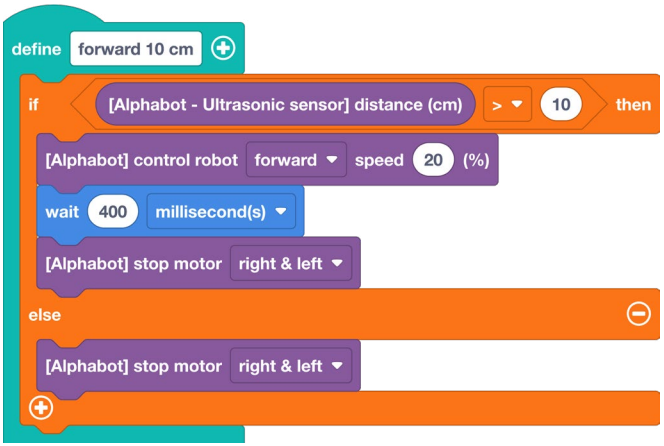
To do this we will only modify the move function so that it detects obstacles.

To do so, we will need the blocks that are in:



Only the changes in the function "Move 10 cm forward" are presented here.

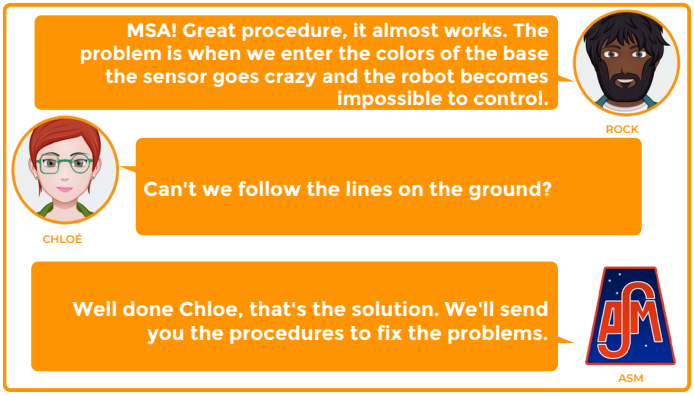




```
45 def avancer_10_cm():
46   if alphabot.readUltrasonicDistance() > 10:
47     alphabot.moveForward(20)
48     utime.sleep_ms(400)
49     alphabot.stop()
50   else:
51     alphabot.stop()
```

In the Python code, the function related to the ultrasound sensor is not added here. If needed, it is available in task 3.

We have already come a long way, our robot is now able to move on its own, avoiding obstacles or rather ignoring them.



Task •6 ⏱ according to the supervisor's estimation
Line tracker

• Task 6-1: Following a line

The line follower integrated into the AlphaBot robot is made up of 5 infrared sensors located under the chassis of the robot.

The principle of operation is as follows: The line follower is composed of 5 infrared sensors that each have an infrared emitting LED and a sensitive photo-transistor. The LED sends a signal and the photo-transistor detects it. The dark absorbs the infrared radiation, the receiver does not detect the signal sent when the sensor is above the line, so it returns 0. In the opposite case, the signal is reflected by the white, the sensor then returns 1.

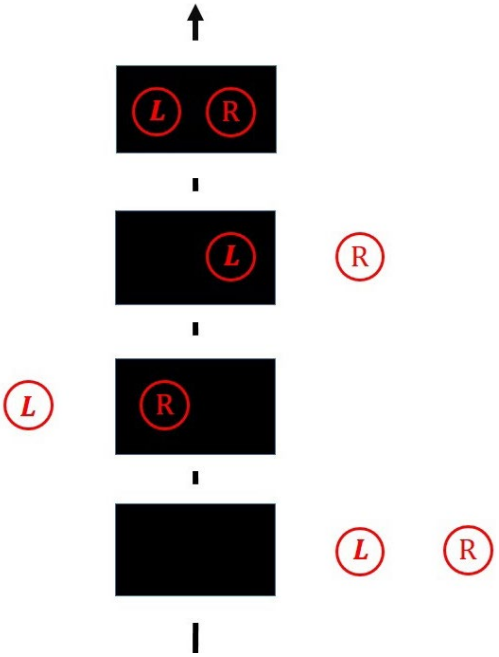
At first we will only use two sensors to follow the line.

- **Case 1:** The 2 sensors are above line.
  -> Return 0,0

- **Case 2:** Line-L is above the line and and Line-R is on the outside.
  -> Return 0,1

- **Case 3:** Line-L is outside and Line-R is above the line.
  -> Return 1,0

- **Case 4:** The 2 sensors are outside the black line.
  -> Return 1,1



Here we can see how the sensors operate and the information that we get from the microcontroller board.

With this information, we will be able to think about what each engine does based on the information coming from the sensors.
To make the task easier, we are going to use a logical table in which there will be two columns for the sensors and two columns for the engines. We are going to use the code 0 for switched off engine or white color and 1 for active engine and black color.

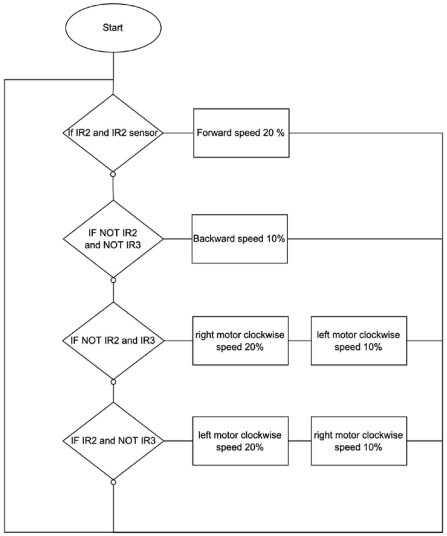
Left sensor	Right sensor	Left engine	Right engine
0 / white	0 / white	?	?
0 / white	1 / black	1 / activated	0 / turned off
1 / black	0 / white	0 / turned off	1 / activated
1 / black	1 / black	1 / activated	1 / activated

We realize that in the first case, there is no more line and in this case we must implement a strategy to find the line. This can be, for example, "move 5 cm forward and then turn around in a circle until you find the line" or simply "back up" because you went too fast and lost the line.

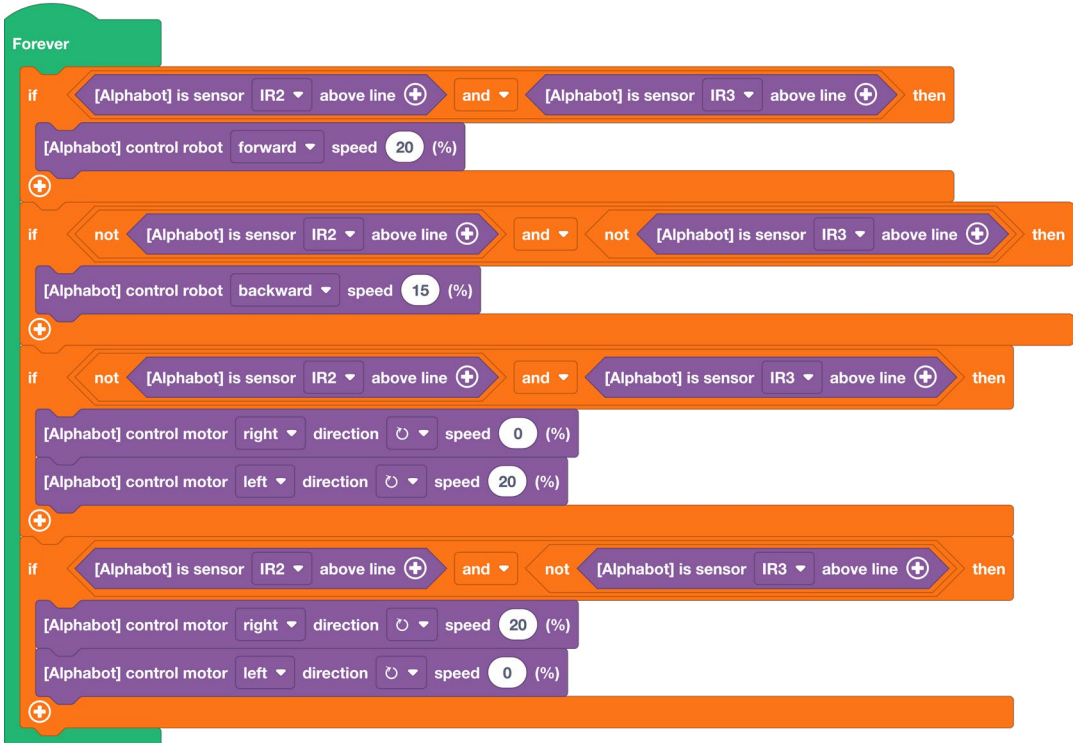
Let's try to implement this.

To do so we will need the following blocks:

In the program, we can see at the beginning in the block "At startup" a "stop the engine" sign. This is a mini initialization, it allows us to be sure that we are starting with the engine off. In this initialization, we can add many things to be sure to start the program in good conditions; for example a "clear screen" option, to be sure that nothing remains on the screen. We can add a pause to give you time to position the robot before it starts.




```
while True:
  if isSensorAboveLine(alphabot, sensor='IR2') and isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.moveForward(20)
  if not isSensorAboveLine(alphabot, sensor='IR2') and not isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.moveBackward(15)
  if not isSensorAboveLine(alphabot, sensor='IR2') and isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.setMotors(right=0)
    alphabot.setMotors(left=20)
  if isSensorAboveLine(alphabot, sensor='IR2') and not isSensorAboveLine(alphabot, sensor='IR3'):
    alphabot.setMotors(right=20)
    alphabot.setMotors(left=0)
```

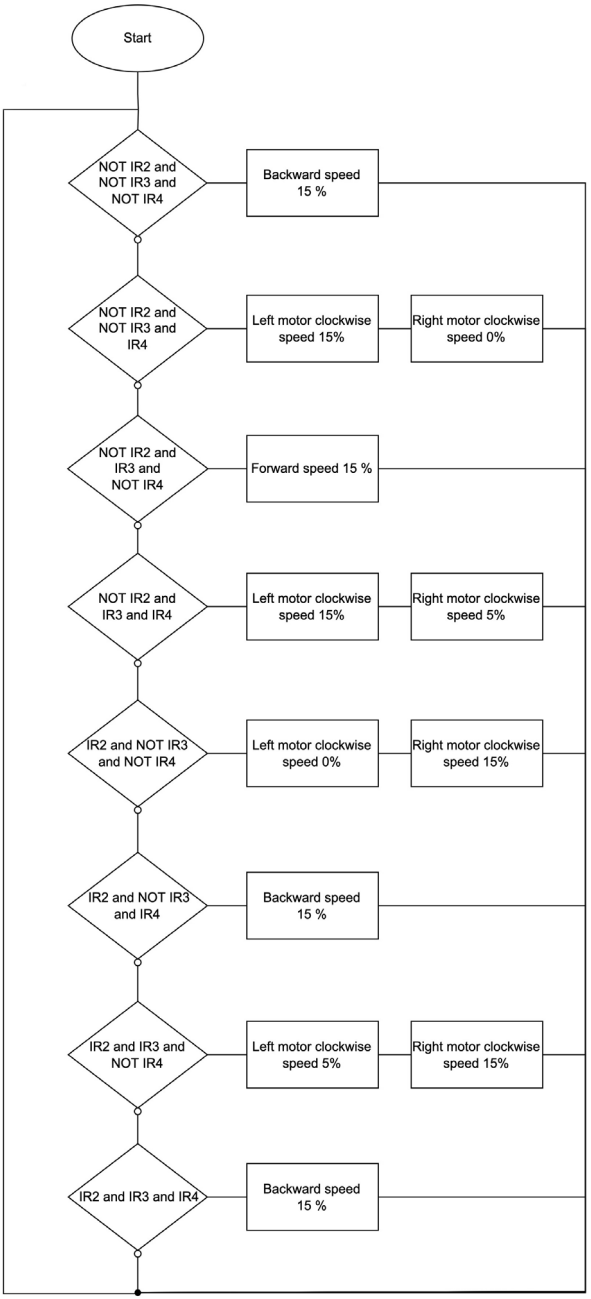


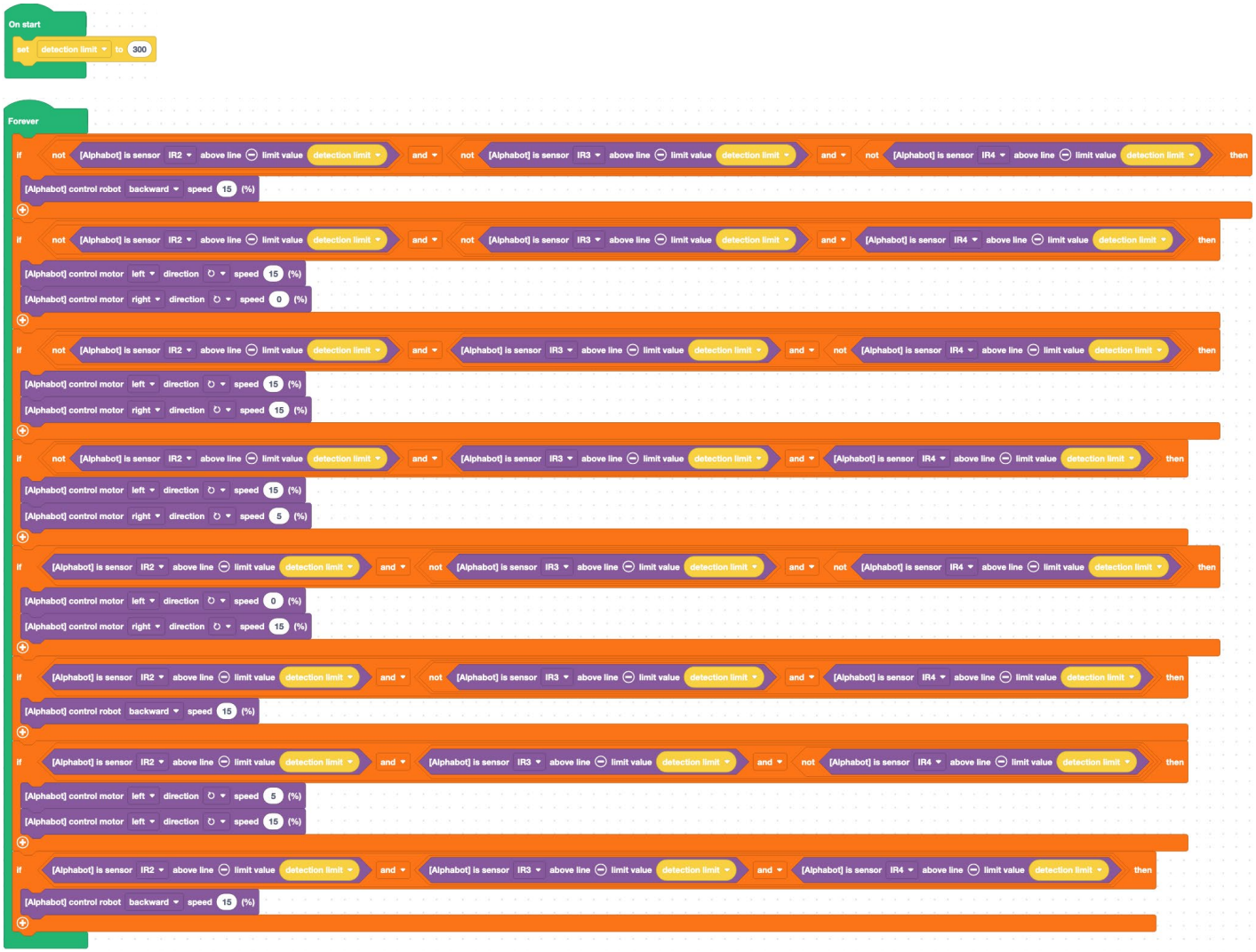
• Task 6-2: Following a line with 3 sensors

Now let's try to improve our program by using three line sensors: left, center and right of the robot.
We will use the following code: 0 for white and 1 for black line:

Left sensor	Center sensor	Right sensor	Left engine	Right engine
0	0	0	-15%	-15%
0	0	1	15%	0
0	1	0	15%	15%
0	1	1	15%	5%
1	0	0	0	15%
1	0	1	-15%	-15%
1	1	0	5%	15%
1	1	1	-15%	-15%

```
while True:
  if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveBackward(15)
  if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=15)
    alphabot.setMotors(right=0)
  if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveForward(15)
  if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=15)
    alphabot.setMotors(right=5)
  if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=0)
    alphabot.setMotors(right=15)
  if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveBackward(15)
  if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
    alphabot.setMotors(left=5)
    alphabot.setMotors(right=15)
  if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
    alphabot.moveBackward(15)
  oled.fill(0)
  oled.show()
```


Start
NOT IR2 and NOT IR3 and NOT IR4
Backward speed 15 %
NOT IR2 and NOT IR3 and IR4
Left motor clockwise speed 15%
Right motor clockwise speed 0%
NOT IR2 and IR3 and NOT IR4
Forward speed 15 %
NOT IR2 and IR3 and IR4
Left motor clockwise speed 15%
Right motor clockwise speed 5%
IR2 and NOT IR3 and NOT IR4
Left motor clockwise speed 0%
Right motor clockwise speed 15%
IR2 and NOT IR3 and IR4
Backward speed 15 %
IR2 and IR3 and NOT IR4
Left motor clockwise speed 5%
Right motor clockwise speed 15%
IR2 and IR3 and IR4
Backward speed 15 %


On start
set detection limit to 300
Forever
if not [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 15 (%)
[Alphabot] control motor right direction speed 0 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 15 (%)
[Alphabot] control motor right direction speed 15 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 15 (%)
[Alphabot] control motor right direction speed 5 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 0 (%)
[Alphabot] control motor right direction speed 15 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 5 (%)
[Alphabot] control motor left direction speed 15 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)

• Task 6-3: Following a line and avoiding obstacles

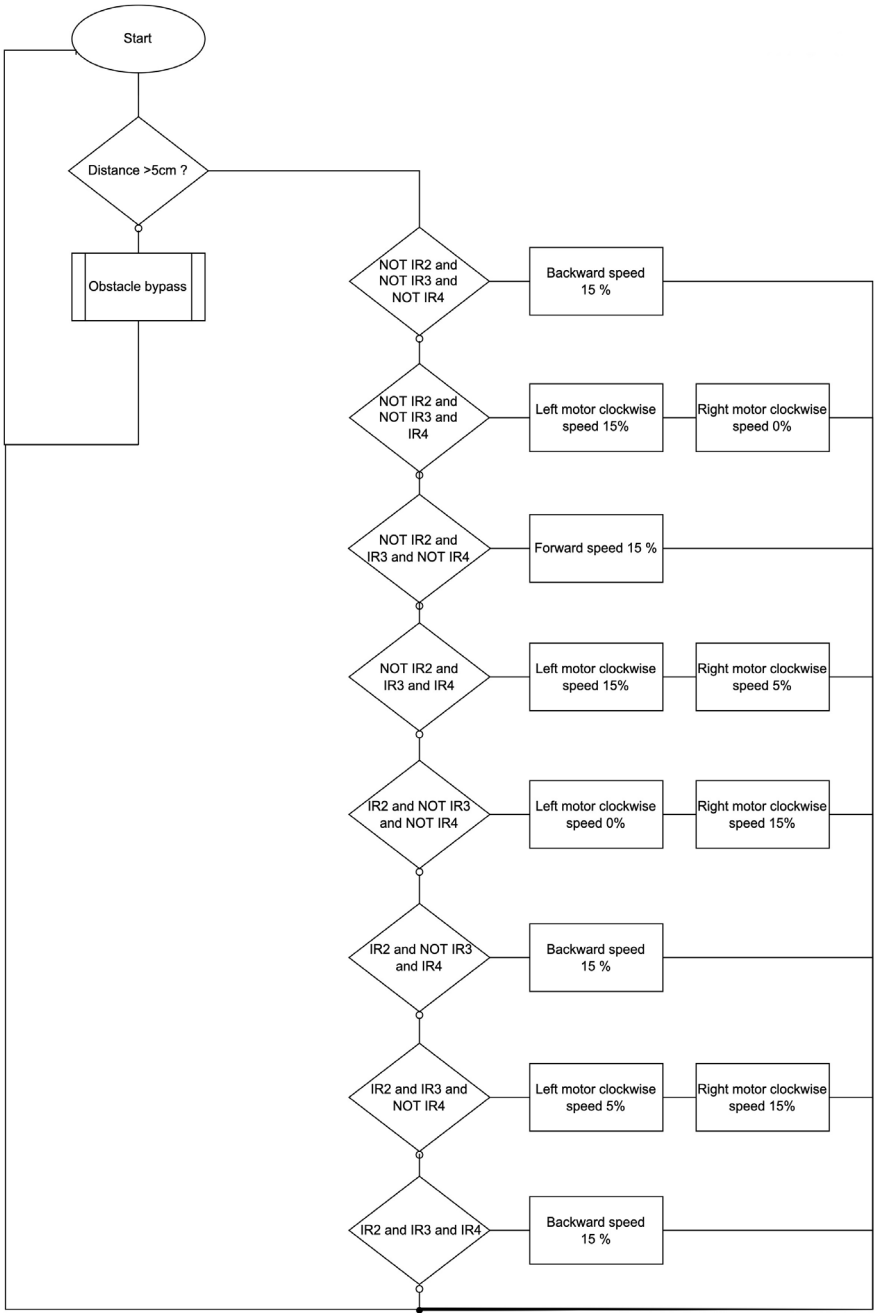
For this task, we are going to imagine that the robot will detect an object while following the line. In this case, it will have to stop following the line, go around the object and then find the line on the other side of the obstacle.

In our case, we will limit the size of the hindering object to 10 cm in diameter.
We are going to use the previous work from task 5 where we used the functions. We are going to keep the functions "Move 10 cm forward" and "Turn to the right" or to the left at a 90° angle. We are just going to add an avoid obstacle function and we will improve our initialization with a pause.

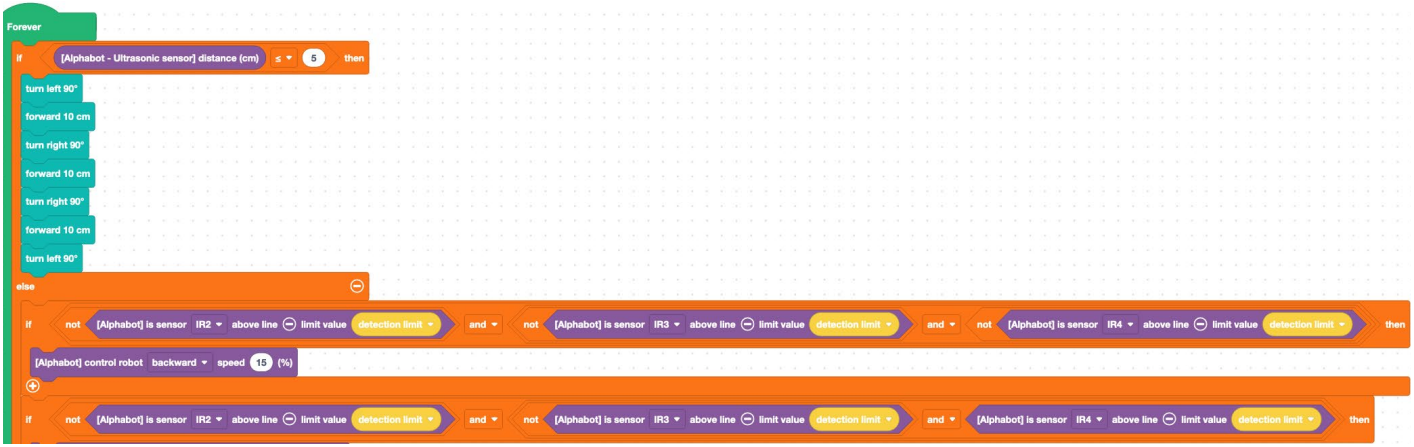
Let's try to implement this.
To do so, we will need the following blocks:

The algorithms and the program will become more consistent but we find many parts from old programs. The same goes for the algorithms, very little change, just complementary additions. The algorithms or codes are not present in full, only the important parts have been reused.

```
while True:
  if alphabot.readUltrasonicDistance() <= 5:
    tourner_gauche_90_C2_B0()
    avancer_10_cm()
    tourner_droite_90_C2_B0()
    avancer_10_cm()
    tourner_droite_90_C2_B0()
    avancer_10_cm()
    tourner_gauche_90_C2_B0()
  else:
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=15)
      alphabot.setMotors(right=0)
    if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
```


Start
Distance >5cm ?
Obstacle bypass
NOT IR2 and NOT IR3 and NOT IR4
Backward speed 15 %
NOT IR2 and NOT IR3 and IR4
Left motor clockwise speed 15%
Right motor clockwise speed 0%
NOT IR2 and IR3 and NOT IR4
Forward speed 15 %
NOT IR2 and IR3 and IR4
Left motor clockwise speed 15%
Right motor clockwise speed 5%
IR2 and NOT IR3 and NOT IR4
Left motor clockwise speed 0%
Right motor clockwise speed 15%
IR2 and NOT IR3 and IR4
Backward speed 15 %
IR2 and IR3 and NOT IR4
Left motor clockwise speed 5%
Right motor clockwise speed 15%
IR2 and IR3 and IR4
Backward speed 15 %

• 58


Forever
if [Alphabot - Ultrasonic sensor] distance (cm) ≤ 5 then
turn left 90°
forward 10 cm
turn right 90°
forward 10 cm
turn right 90°
forward 10 cm
turn left 90°
else
if not [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then

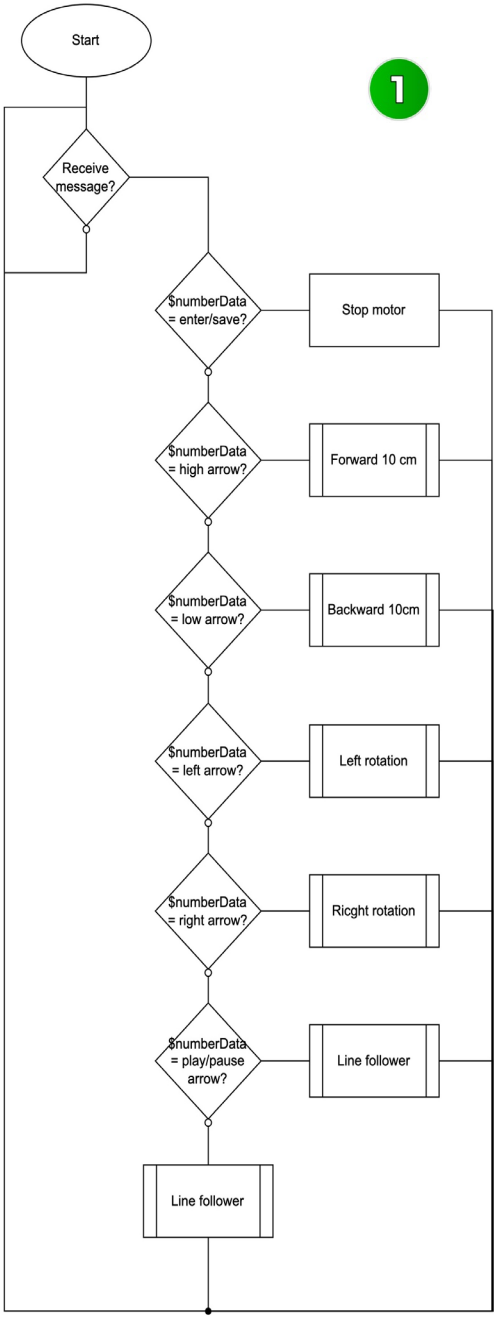
Task • 7 ⏱ according to the supervisor's estimation

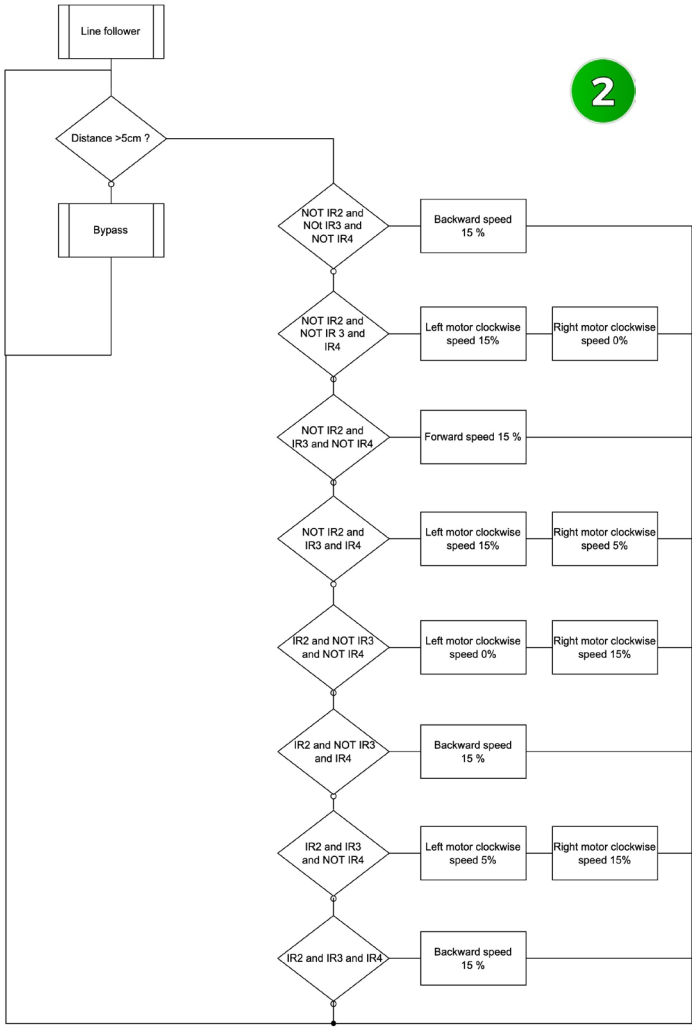
Finally, a routine!

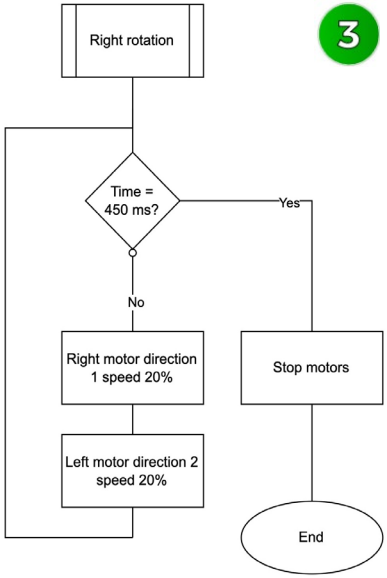
Role-playing in a complex situation: The robot follows the loop continuously but a sudden drop in temperature occurs. The robot stops all its activities and must take shelter. To do this, we switch to manual control and plan a return to the Ares I center or to the indicated coordinates. The movements will be secured by using the ultrasonic sensors or the ToF sensor to check if a movement is correct. The remote monitoring will be done with the remote control. Planning several types of movements will be necessary.

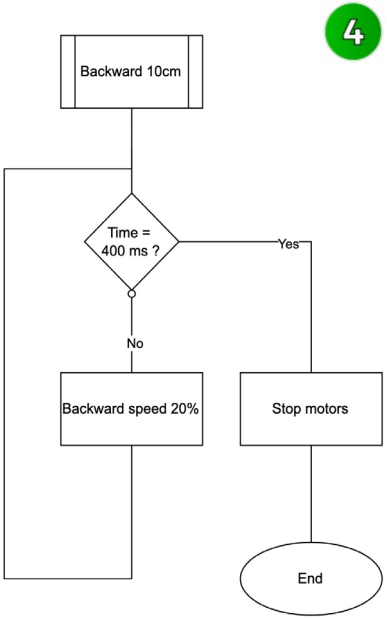
Be careful, the number of buttons on the remote control are limited and you have to leave a free button to simulate a problem and another one to restart the robot as a line follower.

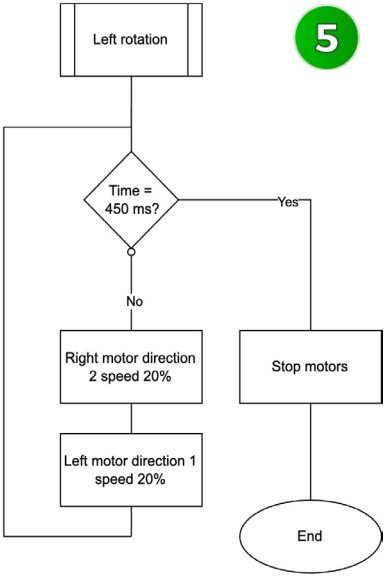
Attention, the whole code is presented here. Although it is not too complex, it becomes significant in size!

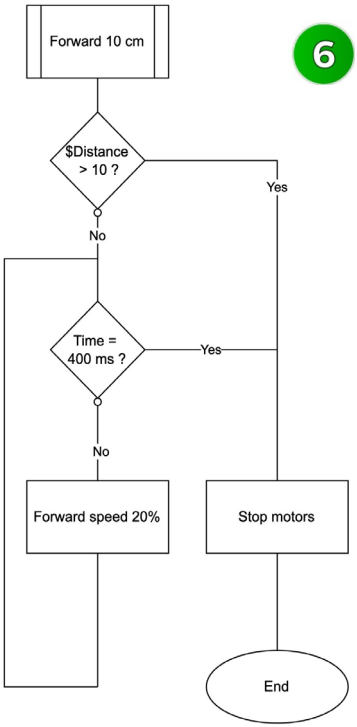


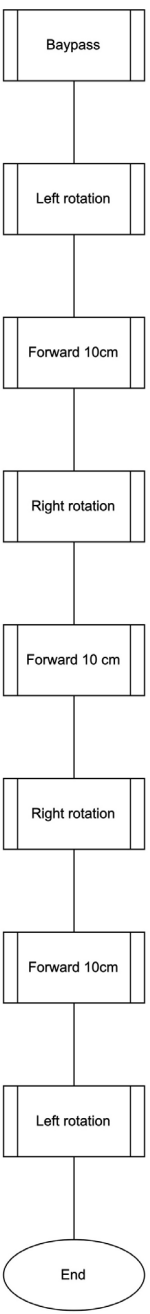

Line follower
Distance >5cm ?
Bypass
2
NOT IR2 and NOt IR3 and NOT IR4
Backward speed 15 %
NOT IR2 and NOT IR 3 and IR4
Left motor clockwise speed 15%
Right motor clockwise speed 0%
NOT IR2 and IR3 and NOT IR4
Forward speed 15 %
NOT IR2 and IR3 and IR4
Left motor clockwise speed 15%
Right motor clockwise speed 5%
IR2 and NOT IR3 and NOT IR4
Left motor clockwise speed 0%
Right motor clockwise speed 15%
IR2 and NOT IR3 and IR4
Backward speed 15 %
IR2 and IR3 and NOT IR4
Left motor clockwise speed 5%
Right motor clockwise speed 15%
IR2 and IR3 and IR4
Backward speed 15 %


Right rotation
3
Time = 450 ms?
Yes
No
Right motor direction 1 speed 20%
Left motor direction 2 speed 20%
Stop motors
End

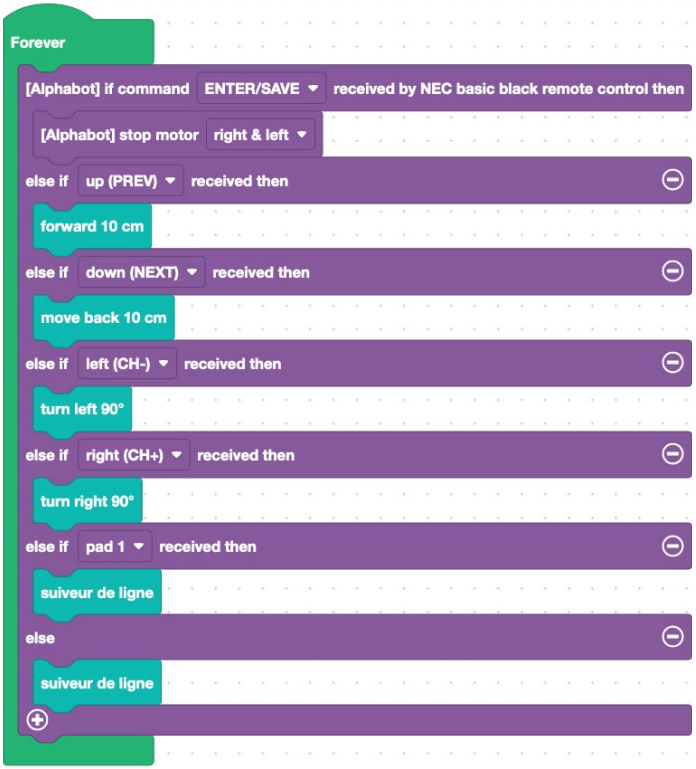

4
Backward 10cm
Time = 400 ms ?
Yes
No
Backward speed 20%
Stop motors
End


Left rotation
5
Time = 450 ms?
Yes
No
Right motor direction 2 speed 20%
Stop motors
Left motor direction 1 speed 20%
End

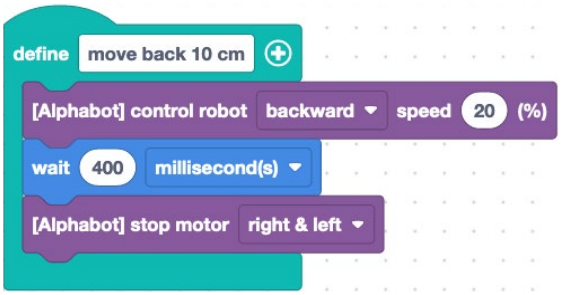

Forward 10 cm
6
$Distance > 10 ?
Yes
No
Time = 400 ms ?
Yes
No
Forward speed 20%
Stop motors
End

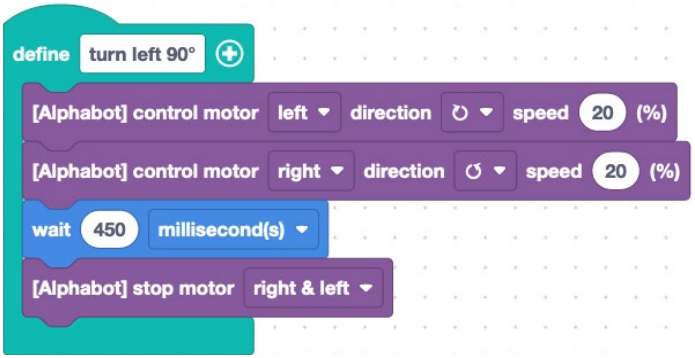

Baypass
7
Left rotation
Forward 10cm
Right rotation
Forward 10 cm
Right rotation
Forward 10cm
Left rotation
End

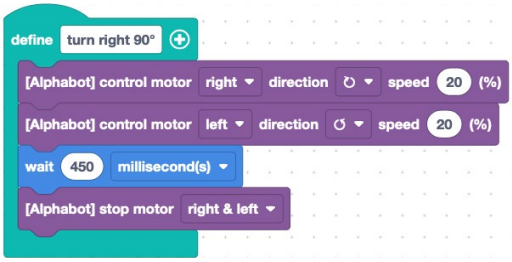

On start
wait 1 second(s)
[Alphabot] calibrate line finder
set detection limit to 300

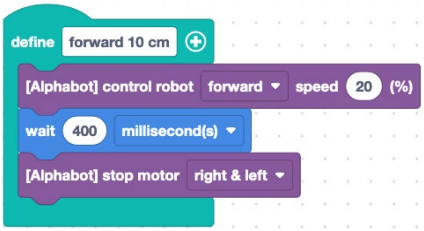

Forever
[Alphabot] if command ENTER/SAVE received by NEC basic black remote control then
[Alphabot] stop motor right & left
else if up (PREV) received then
forward 10 cm
else if down (NEXT) received then
move back 10 cm
else if left (CH-) received then
turn left 90°
else if right (CH+) received then
turn right 90°
else if pad 1 received then
suiveur de ligne
else
suiveur de ligne

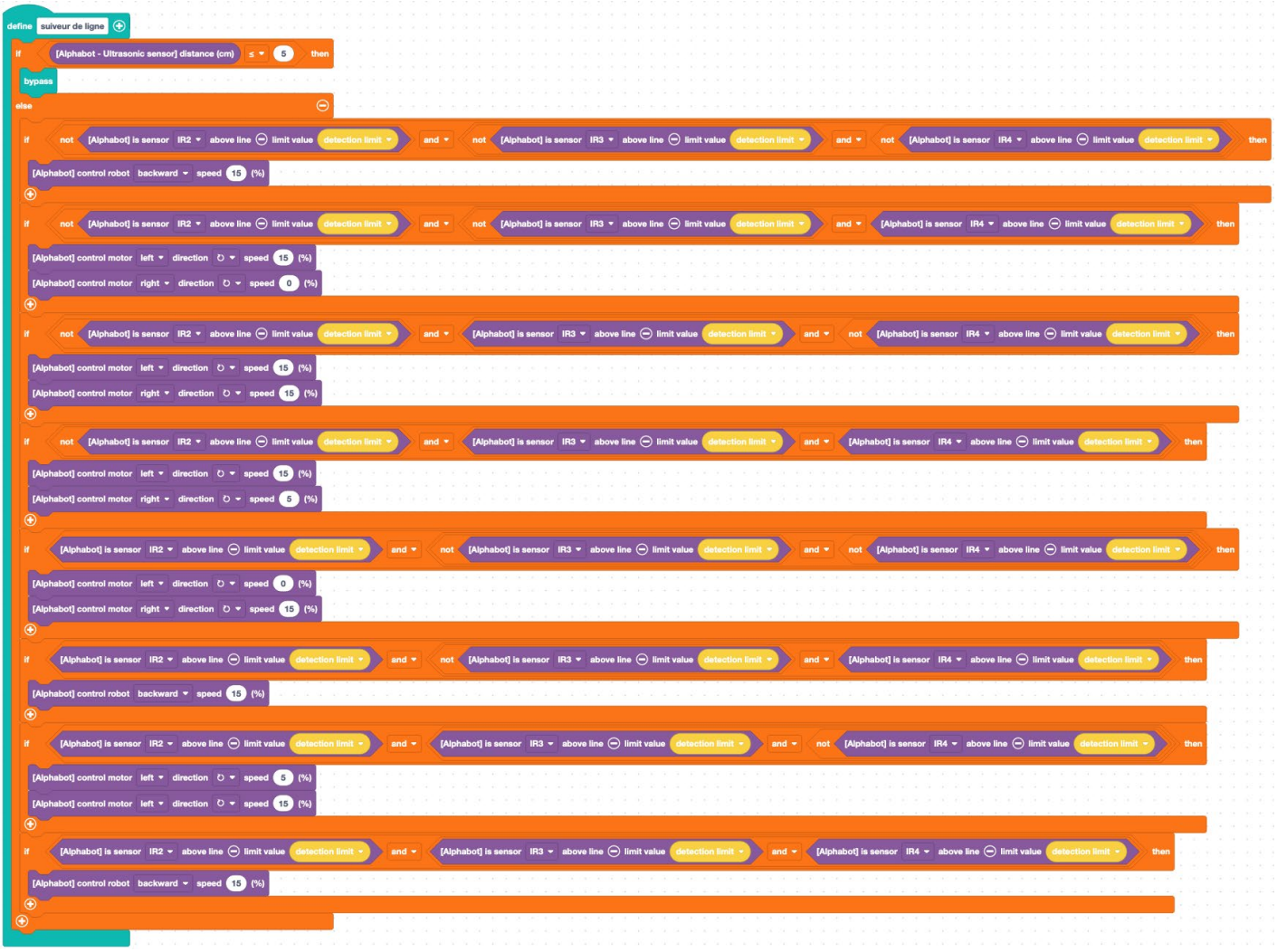

define bypass
turn left 90°
forward 10 cm
turn right 90°
forward 10 cm
turn right 90°
forward 10 cm
turn left 90°


define move back 10 cm
[Alphabot] control robot backward speed 20 (%)
wait 400 millisecond(s)
[Alphabot] stop motor right & left


define turn left 90°
[Alphabot] control motor left direction speed 20 (%)
[Alphabot] control motor right direction speed 20 (%)
wait 450 millisecond(s)
[Alphabot] stop motor right & left


define turn right 90°
[Alphabot] control motor right direction speed 20 (%)
[Alphabot] control motor left direction speed 20 (%)
wait 450 millisecond(s)
[Alphabot] stop motor right & left


define forward 10 cm
[Alphabot] control robot forward speed 20 (%)
wait 400 millisecond(s)
[Alphabot] stop motor right & left


define suiveur de ligne
if [Alphabot - Ultrasonic sensor] distance (cm) ≤ 5 then
bypass
else
if not [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 15 (%)
[Alphabot] control motor right direction speed 0 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 15 (%)
[Alphabot] control motor right direction speed 15 (%)
if not [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 15 (%)
[Alphabot] control motor right direction speed 5 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 0 (%)
[Alphabot] control motor right direction speed 15 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and not [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and not [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control motor left direction speed 5 (%)
[Alphabot] control motor left direction speed 15 (%)
if [Alphabot] is sensor IR2 above line limit value detection limit and [Alphabot] is sensor IR3 above line limit value detection limit and [Alphabot] is sensor IR4 above line limit value detection limit then
[Alphabot] control robot backward speed 15 (%)

```
import machine
from stm32_alphabot_v2 import AlphaBot_v2
import utime
from stm32_nec import NEC_8, NEC_16
import gc
from stm32_ssd1306 import SSD1306, SSD1306_I2C

alphabot = AlphaBot_v2()
ir_current_remote_code = None
oled = SSD1306_I2C(128, 64, machine.I2C(1))

def remoteNEC_basicBlack_getButton(hexCode):
  if hexCode == 0x0c: return "1"
  elif hexCode == 0x18: return "2"
  elif hexCode == 0x5e: return "3"
  elif hexCode == 0x08: return "4"
  elif hexCode == 0x1c: return "5"
  elif hexCode == 0x5a: return "6"
  elif hexCode == 0x42: return "7"
  elif hexCode == 0x52: return "8"
  elif hexCode == 0x4a: return "9"
  elif hexCode == 0x16: return "0"
  elif hexCode == 0x40: return "up"
  elif hexCode == 0x19: return "down"
  elif hexCode == 0x07: return "left"
  elif hexCode == 0x09: return "right"
  elif hexCode == 0x15: return "enter_save"
  elif hexCode == 0x0d: return "back"
  elif hexCode == 0x45: return "volMinus"
  elif hexCode == 0x47: return "volPlus"
  elif hexCode == 0x46: return "play_pause"
  elif hexCode == 0x44: return "setup"
  elif hexCode == 0x43: return "stop_mode"
```

```
  elif hexCode == 0x44: return "setup"
  elif hexCode == 0x43: return "stop_mode"
  else: return "NEC remote code error"

def remoteNEC_callback(data, addr, ctrl):
  global ir_current_remote_code
  if data < 0:  # NEC protocol sends repeat codes.
    print('Repeat code.')
  else:
    print('Data {:02x} Addr {:04x} Ctrl {:02x}'.format(data, addr, ctrl))
    ir_current_remote_code = remoteNEC_basicBlack_getButton(data)

classes = (NEC_8, NEC_16)
ir_remote = classes[0](alphabot.pin_IR, remoteNEC_callback)
alphabot.calibrateLineFinder()

def suiveur_de_ligne():
  if alphabot.readUltrasonicDistance() <= 5:
    Contournement()
  else:
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    if not alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=15)
      alphabot.setMotors(right=0)
    if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveForward(15)
    if not alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=15)
      alphabot.setMotors(right=5)
    if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=0)
      alphabot.setMotors(right=15)
    if alphabot.TRSensors_readLine(2) <= 300 and not alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and not alphabot.TRSensors_readLine(4) <= 300:
      alphabot.setMotors(left=5)
      alphabot.setMotors(right=15)
    if alphabot.TRSensors_readLine(2) <= 300 and alphabot.TRSensors_readLine(3) <= 300 and alphabot.TRSensors_readLine(4) <= 300:
      alphabot.moveBackward(15)
    oled.fill(0)
    oled.show()

def Contournement():
  tourner_gauche_90_C2_B0()
  avancer_10_cm()
  tourner_droite_90_C2_B0()
  avancer_10_cm()
  tourner_droite_90_C2_B0()
  avancer_10_cm()
  tourner_gauche_90_C2_B0()
```

```
84 def tourner_gauche_90_C2_B0():
85   alphabot.setMotors(left=20)
86   alphabot.setMotors(right=-20)
87   utime.sleep_ms(450)
88   alphabot.stop()
89 
90 def tourner_droite_90_C2_B0():
91   alphabot.setMotors(right=20)
92   alphabot.setMotors(left=-20)
93   utime.sleep_ms(450)
94   alphabot.stop()
95 
96 def avancer_10_cm():
97   if alphabot.readUltrasonicDistance() > 10:
98     alphabot.moveForward(20)
99     utime.sleep_ms(400)
100    alphabot.stop()
101  else:
102    alphabot.stop()
103 
104 def reculer_10():
105  alphabot.moveBackward(20)
106  utime.sleep_ms(400)
107  alphabot.stop()
108 
109 utime.sleep(3)
110 
111 while True:
112  utime.sleep_ms(150)
113  gc.collect()
114  if ir_current_remote_code == "enter_save":
115    alphabot.stop()
116  elif ir_current_remote_code == "up":
117    avancer_10_cm()
121    tourner_gauche_90_C2_B0()
122  elif ir_current_remote_code == "right":
123    tourner_droite_90_C2_B0()
124  elif ir_current_remote_code == "play_pause":
125    suiveur_de_ligne()
126  else:
127    suiveur_de_ligne()
```

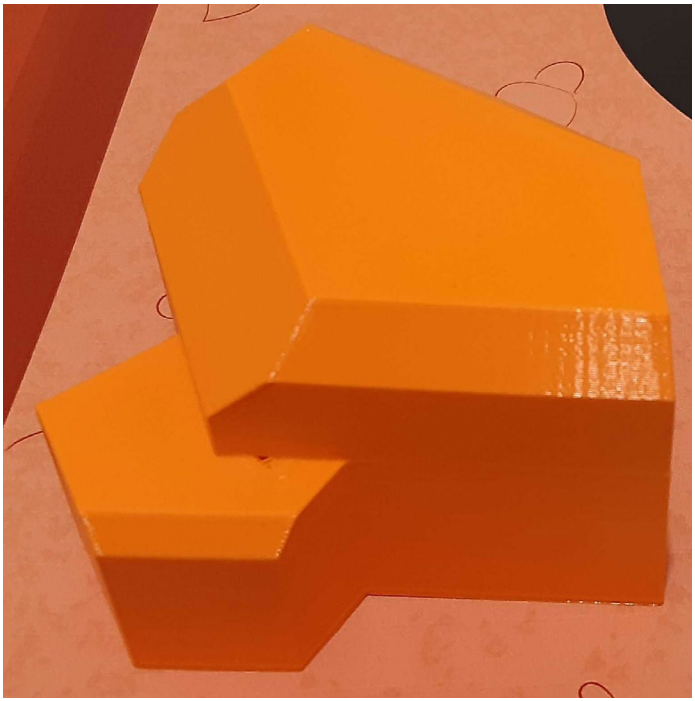
Furthermore

In order to complement the programming of this booklet, you can add the following activities over the course of the year or in parallel, on other levels .

Creating 3D objects:

- A mini crater that adapts to the course
- A rock
- A Martian base
- A solar panel

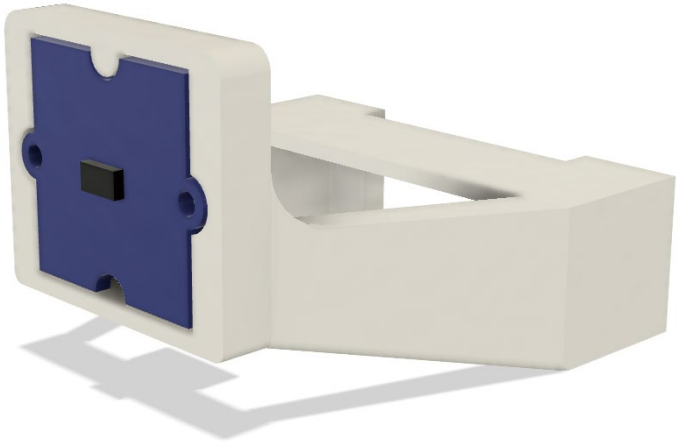
A more important project would be to create a station for the robot. You could start from the idea of a garage with an automatic door that will allow you to use the ultrasonic sensor to detect the presence of the robot at the opening and closing of the station's entry lock. In this project, there is programming, but also design and manufacturing with machines, tools, 3D printing or laser cutting.



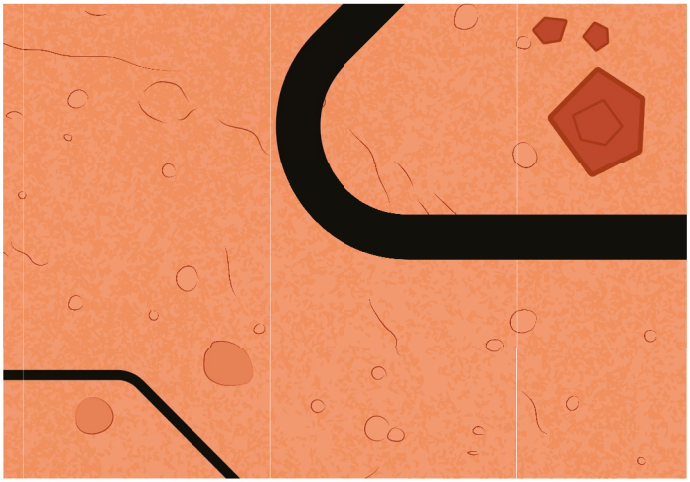
An example of a 3D printed Martian rock, which makes for an ideal obstacle for the distance sensor!

Furthermore

If you want to replace the ultrasonic sensor with the Time of Flight sensor, here is a support to 3D print.
Find it on the website vittascience.com/learn

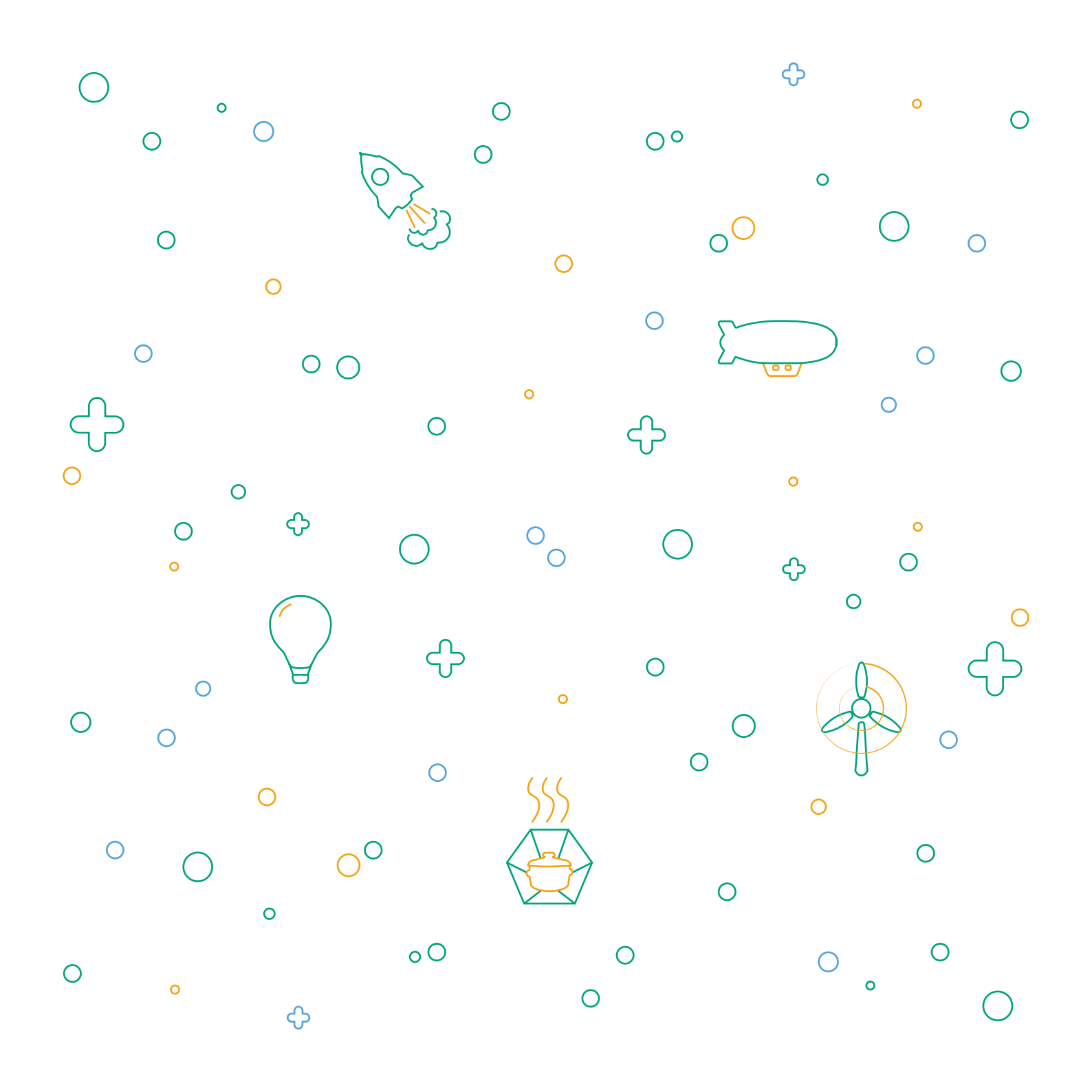


To close the track of the Martian robot, print this sheet in A4 format and place it on the track.
Find it on the website vittascience.com/learn



Notes

Notes



This is the Martian robot building guide.
It explains in detail each step and each task necessary to cary out the experiment. This guide is not exhaustive, your imagination and the resources available on our website can help you enhance the experience!

This booklet is subject to change, we invite you to to consult the version available online which is updated regularly on : www.vittascience.com/learn
(search for "Martian robot" to find the user guide)

vitta science | life.augmented

STEM your way
Innovation depends on you

IPCEI on Microelectronics