



Robot Programming Python & Robockly WIFI-203



NeuLog 



Robot Programming Python & Robockly WIFI-203

2_11

© All rights reserved.

The material in this book may not be copied, duplicated, printed, translated, re-edited or broadcast without prior agreement in writing.

For further information contact info@neulog.com

Contents

Chapter 1 – Control and Robots.....	1
1.1 Robots	1
1.2 Control systems.....	2
1.3 SENSE autonomous.....	3
1.4 WIFI-203 Wireless coding unit.....	4
1.5 Blockly and Robockly.....	5
1.6 Python	5
1.7 Starting the WIFI-203	7
Experiment 1.1 – Direct Mode	9
1.1.1 SENSE movement.....	12
1.1.2 The SENSE sensors	13
Experiment 1.2 – First Programs	15
1.2.1 First program – forward	17
1.2.2 Program download	20
1.2.3 Forward and backward.....	21
1.2.4 Turning left and right.....	22
1.2.5 Challenge exercises – Moving in a square.....	25
Experiment 1.3 – Interactive Programs	26
1.3.1 Variables	27
1.3.2 Repeat until	30
1.3.3 Endless loop	34
1.3.4 Movement between two lines	35
1.3.5 Challenge exercise – Between a wall and a line (I).....	36
Experiment 1.4 – Functions and Variables.....	37
1.4.1 Variable instead of constant.....	39
1.4.2 Functions	42
1.4.3 Challenge exercise – Between a wall and a line (II).....	45
1.4.4 Moving along a black line.....	45
1.4.5 Challenge exercise – Along a complex black line	47
Experiment 1.5 – Conditions and Decisions	48
1.5.1 The If instruction.....	48
1.5.2 OFF and ON with different values.....	53
1.5.3 AND condition	55
1.5.4 OR condition	56

II

1.5.5	Challenge exercise – Along two lines.....	57
1.5.6	Movement along a wall.....	57
1.5.7	Challenge exercises – Along walls	59
Challenge 1.6 – Counting.....		60
Challenge 1.7 – Automatic movement.....		60
Challenge 1.8 – Loops		60
Challenge 1.9 – Loops and procedures.....		61
Challenge 1.10 – "Don't touch me" robot.....		61
Challenge 1.11 – Robots in a convoy		61
Challenge 1.12 – Movement in a labyrinth		62
Challenge 1.13 – Exiting a circle.....		63
Challenge 1.14 – Moving along corridors		63
Experiment 1.15 – Editing Python program		64
1.15.1	Python built-in library	64
Chapter 2 – Brain Units.....		72
2.1	Brain units	72
2.2	NeuLog sensors as brain units	73
2.3	Changing Brain unit ID	74
Experiment 2.1 – Sound Sensor		75
2.1.1	Challenge exercise – Wait for sound	77
Experiment 2.2 – Motion Sensor.....		78
3.2.1	Challenge exercise – Moving in a distance range.....	81
Experiment 2.3 – Brain Tracking Unit		82
2.3.1	IR Transmitter	82
2.3.2	Brain tracking unit.....	82
2.3.3	Challenge exercise – Tracking a robot with IR transmitter	86
Experiment 2.4 – Brain Gripper Arm.....		87
2.4.1	Brain gripper arm	87
2.4.2	Challenge exercises – The SENSE with gripper arm	93
Experiment 2.5 – Robot and Science experiment.....		94
2.5.1	The NeuLog light sensor	94

III

2.5.2	Light intensity vs distance.....	95
2.5.3	The SENSE as USB module	95
2.5.4	Challenge exercise – Magnetic fields vs distance	99
Chapter 3 – Autonomous vehicle challenges		100
3.1	Autonomous vehicles	100
3.2	Programming tips	100
Challenge 3.1 – Along black lines		101
3.1.1	Left and right along a black line	101
3.1.2	Smooth movement along a black line	102
3.1.3	Adding Forward movement	103
3.1.4	Along a black line with a stop in front of an obstacle	104
Challenge 3.2 – AGV – Automatic Guided Vehicle		105
Challenge 3.3 – AGV between stations.....		107
Challenge 3.4 – Along a building block.....		108
3.4.1	Left and right along walls	109
3.4.2	Smooth movement along a black line	110
3.4.3	Adding Forward movement	110
3.4.4	Along a wall with a stop in front of an obstacle	112
Challenge 3.5 – Along a building block and bypass cars.....		113
Challenge 3.6 – Autonomous museum guard		114
Challenge 3.7 – Along a building block with stop sign		115
Challenge 3.8 – Along a building block with stop for pedestrian.....		115
Challenge 3.9 – Building block guard		116
Challenge 3.10 – Two buildings guard		117
Challenge 3.11 – Taxi driver		118
Challenge 3.12 – Taxi driver with passenger.....		119
Challenge 3.13 – Home vacuum cleaner robot		120

Chapter 1 – Control and Robots

1.1 Robots

The world today is a world of embedded computer systems. We find them in media systems, watches, phones, remote control, cars, and many more electronics. A few years ago, we did not see terms such as wearable computing or internet of things.

Everyday a surprising new product or application appears and months later, we cannot realize how we lived without it. Modern systems are based more and more on machine learning and artificial intelligence.

The robotic systems which are part of the embedded computer system, perform independent activities such as: search, manipulation, identification, activation, protection etc.

Many systems combine a certain kind of artificial intelligence in operating and communication between machines.

The robotic system includes the controller, building components, wheels, gears, motors, sensors, and more.

Each robotic system includes a controller that allows it to operate in accordance with different operating programs. The robot developer writes these programs on a computer and forwards them to the controller.

Building a robotic system creates a challenge to acquire knowledge in various technology areas (electronics, computers, mechanics, electricity, etc.).

There are many types of robots such as arm robots, mobile robots, walking robots and more.

The SENSE robots are a series of robots and "brain" units for study, programming and making robots with wide variety of robot applications.

The sense autonomous is a robot enables us to program many robot applications and functions such as movement on a line, movement along walls, tracking, AGV (Automatic Guided Vehicle), autonomic car, autonomic guard vehicle, autonomic taxi driver, environment monitoring, manipulating car and more. All these applications are described as exercises in this book.

1.2 Control systems

A robot is a computerized control system.

A "Control system" may be defined as a group of components, which can be operated together to control multiple variables, which govern the behavior of the system.

Examples:

- Air-conditioning systems control the temperature in the room.
- A greenhouse control system controls temperature, humidity, light, and irrigation.
- A speed control system maintains a steady motor speed regardless of the changing load on the motor.
- A light control system can maintain a steady level of light, regardless of the amount of available sunlight. The control system turns lamps ON or OFF according to the requirements.

Three basic units are in every computerized control system:

1. **Input unit** – the unit that reads the system sensors like temperature, light, distance, touch switch, etc. and feeds information into the control unit.
2. **Control unit** – the "BRAIN" of the control system, which contains the system program in its memory and performs the program instructions and processes the received data.
3. **Output unit** – the unit that operates the system actuators such as motors, lamps, pump, and fan as the results of the inputs and the program "decisions".

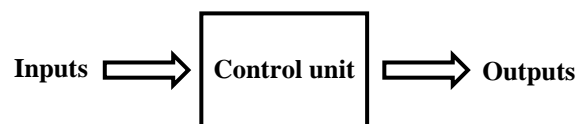


Figure 1-1

The control unit is connected to a computer for programming and downloads a program to the control unit flash memory.

Disconnecting the control unit from the computer and connecting a power source such as a battery to it will create an independent system.

1.3 SENSE autonomous



SENSE autonomous is a mobile robot for applications such as:

- Movement along black line or white line.
- Movement along walls or in a labyrinth.
- **Autonomous vehicle** such as: AGV, autonomous car, autonomous guard vehicle, autonomous taxi driver, autonomous manipulator.
- Following a moving body holding IR transmitter using tracking module.
- **Environmental monitoring** and measurement robot with NeuLog sensors.

The **SENSE autonomous** has the following built in:

- Base unit
- 3 connectors for NeuLog sensors or brain units
- 5 IR range sensors
- 1 line sensor
- Pivot wheel
- 2 motors with wheels
- A controller for the base sensors, motors, and independent operation
- A flash memory for the user programs
- USB connector for connection to PC or MAC

The sense autonomous comes with an adapter for external battery. Such battery can be a standard Power Bank with USB outlet.

You may also have the NeuLog battery module BAT-202, which can be plugged directly into one of the SENSE sockets



When connecting such battery to the sense autonomous and disconnecting it from the PC, the sense autonomous becomes an independent robot running on its internal program in its flash memory.

In this book, we shall call the sense autonomous in short **SENSE**.

1.4 WIFI-203 Wireless coding unit

WIFI-203 is an embedded Linux controller. It is a Wi-Fi module housed in a rigid plastic packaging with colored label.

The module works wirelessly with any computer platform: computers, tablets, IPADs and smart phones.

The module has two connectors for communication with NeuLog sensors or with brain I/O units. The module includes flash memory used as hard disk for program files.



There is no downloaded application or software installation as they are built directly into the Wi-Fi module which transmits its own closed wireless network.

All current popular browsing programs can be used with the Wi-Fi module to allow you to use whichever browser you and your students feel most comfortable with.

The Wi-Fi module can be powered by the NeuLog battery module or by a USB power source.

You'll notice that on the front of the WIFI-203 module there is a LED indicator that includes four color LEDs. Underneath the indicator, there is a legend that explains what each color represents:

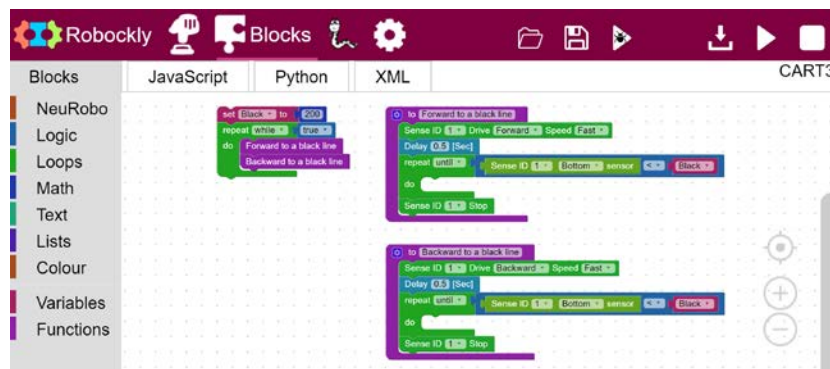
- **Blue (access point)** – When the blue LED is the only light on, the Wi-Fi module is in Access Point mode (described below) and is transmitting a closed wireless network which you can connect devices to.
- **Green (client mode)** – When the green LED is the only light on, the Wi-Fi module is in Client Mode (described below) and allows your wireless devices to control sensors through the website **www.Wi-Fi201.com** as well as browse the internet.
- **Red (transmission)** – The red LED will blink when there is active communication between any connected sensors and the computer or smart device.
- **Orange (USB connection)** – The orange LED will remain lit when the Wi-Fi module is working in USB wire connected mode. We use the USB mode to upgrade the WIFI-203 firmware with a special update software.

1.5 Blockly and Robockly

Blockly is a client-side JavaScript library for creating visual block programming languages and editors. It is a project of Google and is open-source under the Apache 2.0 License.

It typically runs in a web browser, and visually resembles Scratch. Blockly uses visual blocks that link together to make writing code easier, and can generate JavaScript, Python, PHP or Dart code.

Robockly has all Blockly (Google program) instructions, enhanced with SENSE instructions.



1.6 Python

The screen limits visual block programming. Visual block programming is excellent for small programs or for testing functions or program sections.

For programs with many functions and procedures we prefer to use high level language based on text editor.

The Python is very simple and very powerful high-level programming language that works on any computer platform. It is an open-source coding program.

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.

An interpreted language, Python has a design philosophy that emphasizes code readability and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.

The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.

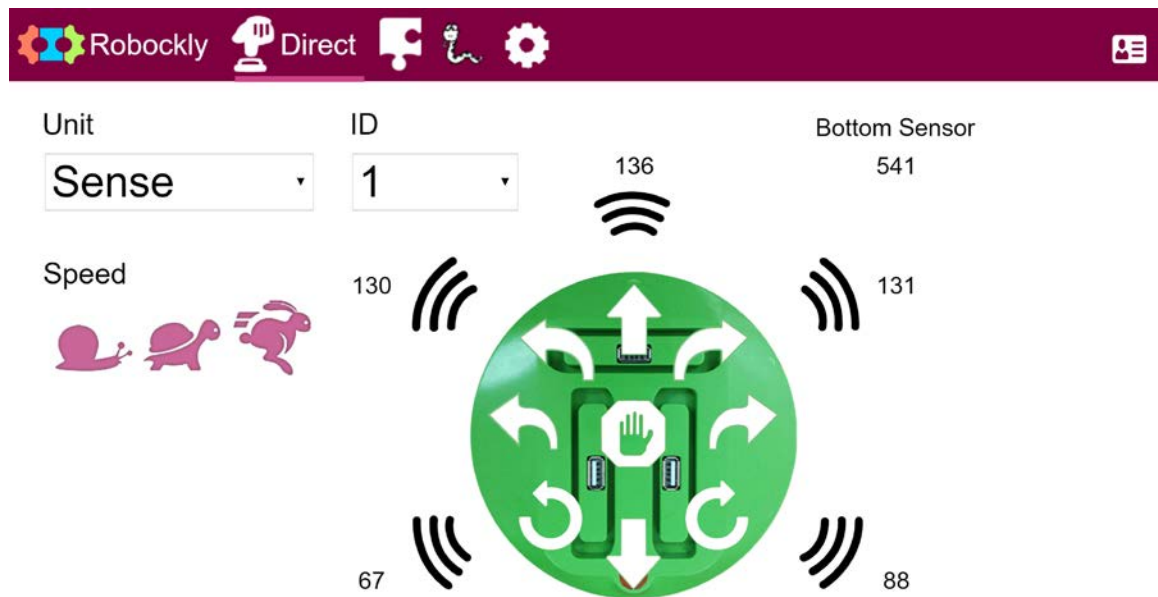
CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, often using English keywords where other languages use punctuation.

Python does not use curly brackets to delimit blocks, and semicolons after statements are optional, in contrast to many other programming languages. Further, Python has fewer syntactic exceptions and special cases than C or Pascal.

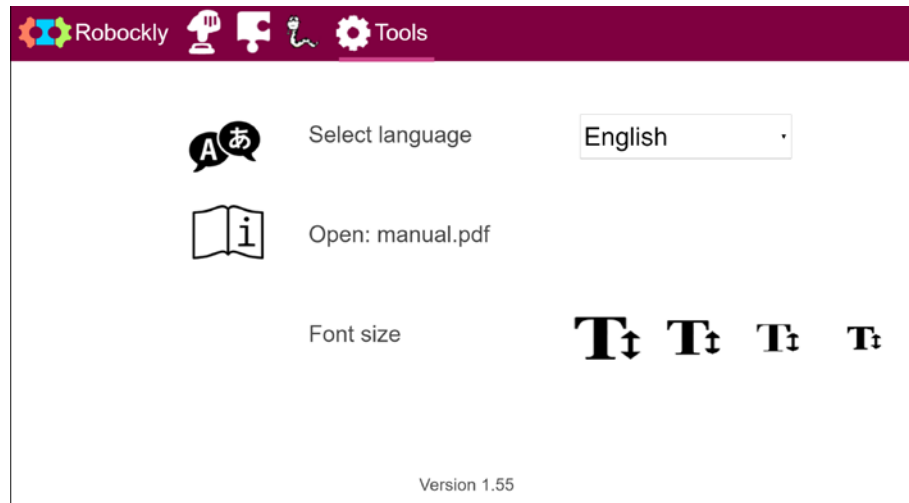
1.7 Starting the WIFI-203

1. Check the number appears on the white label of the back of the WIFI-203 module (i.e., 3233).
2. Plug the WIFI-203 on the Sense robot.
3. Plug battery module on the Sense robot.
4. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
5. Enter your computer Wi-Fi network and select the neulog network (i.e. neulog3233).
6. Go to Chrome browser and surf to **wifi203.com**.
7. The following screen should appear:



This screen is for **Direct mode** (explained in experiment 1.1).

8. Click on the **Tool**  button and the following screen should appear:



9. This screen enables you to change language, and fonts.
10. The screen shows the software version.

It is recommended to compare it with the last version of the download software in **neulog.com** site and to upgrade when it is lower.

Exit is done in two steps.

1. Close the browser window.
2. Disconnect the battery module from the SENSE.

Experiment 1.1 – Direct Mode

Objectives:

- To study the SENSE units and components.
- How to operate the SENSE units in direct mode.

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 battery module

Discussion:

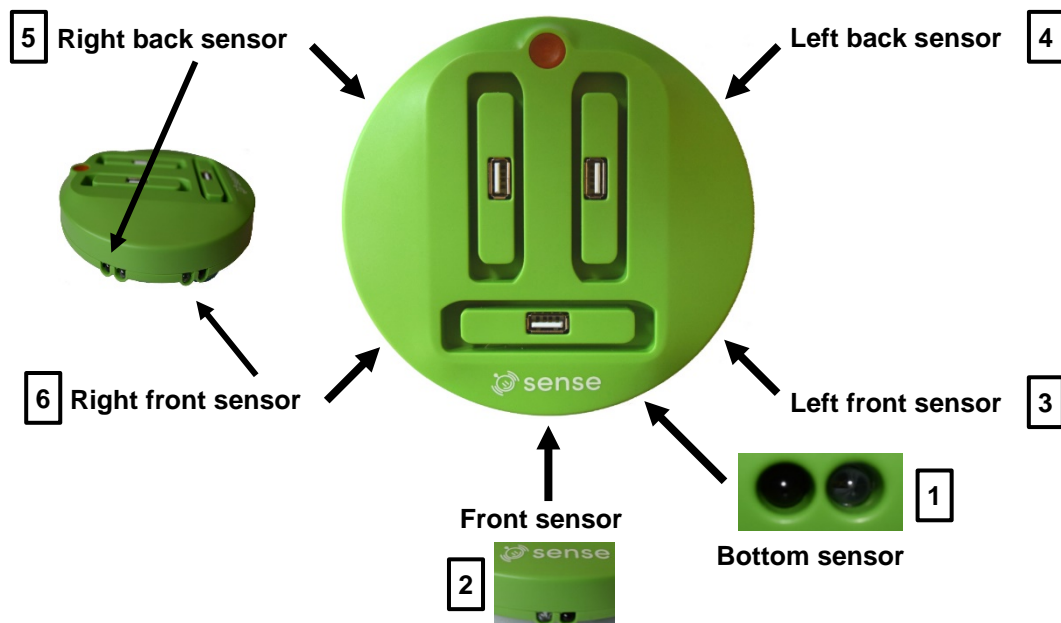
Robockly software enables the user to operate the SENSE controller directly.

We shall learn how to operate the SENSE motors and to read its sensors directly without programming.

Procedure:

1. Observe the SENSE.
2. Hold the SENSE and turn it.
3. The SENSE includes:
 - Two motors with wheels.
 - One pivot wheel.
 - Five range sensors made of an infrared (IR) LED and phototransistor each.
 - A line (black or white) sensor made of an infrared LED and phototransistor too.

Identify them.



4. Check the number appears on the white label of the back of the WIFI-203 module (i.e. 3233)
5. Plug the WIFI-203 on the Sense robot
6. Plug battery module on the Sense robot
7. Wait until the WIFI-203 LED stops blinking and turns to blue constant
8. Enter your computer Wi-Fi network and select the neulog network (i.e. neulog3233)

9. Go to Chrome browser and surf to **wifi203.com**
10. The following **Direct mode** screen should appear:



The direct mode enables you to test the system and the sensors' readings before programming and running the programs. This stage saves a lot of frustration in development.

This **Direct** mode window is for manual control and test of the robot optional units: **Sense, Robo, Robo Ex, Brain arm, Brain servo, Brain motor** or **NeuLog** sensor.

The following screens are **Direct** mode windows of the **Robo, Brain arm** and **NeuLog** sensor.



Brain arm and NeuLog sensor are explained in chapter 3.

The **Direct** screen is changed according to the selected unit. Each unit has its own default ID number. The user can change the module ID number. In this book, we shall refer to the default ID number of the unit as 1.

1.1.1 SENSE movement

The SENSE robot has 9 movement commands:

Forward	both wheels rotate forward
Stop	both wheels stop
Backward	both wheels rotate backward
Left deviate	right wheel rotates forward fast and left wheel rotates forward slower a little
Left turn	right wheel rotates forward fast and left wheel rotates forward very slowly
Left rotate	right wheel rotates forward fast and left wheel rotates backward fast
Right deviate	left wheel rotates forward fast and right wheel rotates forward slower a little
Right turn	left wheel rotates forward fast and right wheel rotates forward very slowly
Right rotate	left wheel rotates forward fast and right wheel rotates backward fast

Each command has an arrow button on the **Direct** screen.

There are three buttons for changing the robot speed:



1. Identify the arrow buttons.
2. Hold the SENSE in your hand.
3. Click on the buttons and observe the SENSE robot reaction.
4. Place the Sense robot on your desk.
5. Again, click on the buttons and observe the SENSE robot reaction.

1.1.2 The SENSE sensors

The SENSE has five wall range sensors on its perimeter and one line sensor on its bottom, having the following names:

- Bottom sensor
- Front sensor
- Right front sensor
- Right back sensor
- Left front sensor
- Left back sensor

Each sensor is composed of an infrared (IR) LED (Light Emitting Diode) and phototransistor (light sensor) directed outward.

When the SENSE controller receives a command to read one of the range sensors, it lights the LED and measure the intensity of the received light.

The range sensors are based on infrared light in order to reduce the effect of the surrounding light.

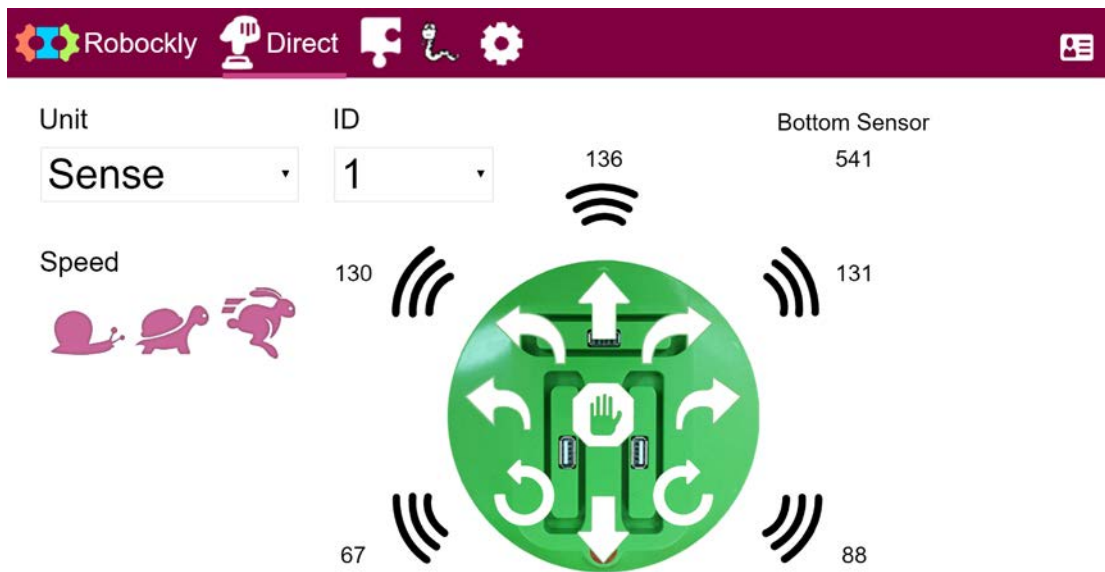
A white surface returns much more light then a black surface. Also colored surfaces return different values. The line sensor on the bottom is based on this effect.

Note:

The range sensors are not calibrated. The read values represent the intensity of the received reflected IR light. For the same distance, you may get different value from each sensor.

Pay attention to the side range sensors. They are all at an angle of 45°. The Tracking a Wall experiment (experiment 1.4) explains the reason for that.

1. Observe the **Direct** screen and the read values around the SENSE picture.



These values are the read values of the SENSE sensors.

2. Place the SENSE on a white surface.

The value of the bottom sensor should be above 500.

3. Place the SENSE on a black surface.

The value of the bottom sensor should go down dramatically.

Note that there may be a big variation in the read value between different black surfaces.

4. Place the SENSE robot at different distances from a wall and observe the effect on each of the five wall sensors.

Note:

The read value of each sensor for the same distance may be different from sensor to sensor.

Experiment 1.2 – First Programs

Objectives:

- Programming with Robockly and Python
- Downloading a program to the controller and running it

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- Battery module

Discussion:

A computer program is composed of chains of instructions according to the programming language instruction set. There are various programming languages, with different instruction sets and multiple types of programming.

We must tell the computer which instruction is the first instruction in the chain. The computer will execute this instruction and will continue on to the following instruction in the chain.

The program may include instructions that direct the computer to other instructions other than the subsequent one. The program may include instructions that move the computer to other chains under condition.

There are many types of programming languages. Each one has its own syntax and its own set of instruction.

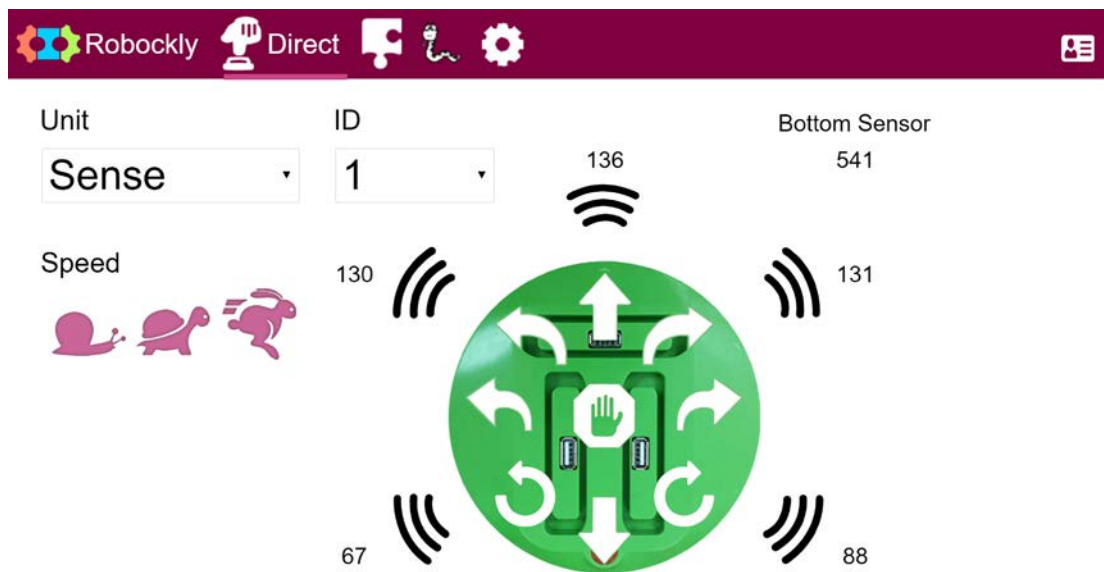
The Robockly is a visual block-programming editor. The Robockly uses blocks that link together to build a program and to concentrate on problem solving, instead of writing code texts in a certain programming language.

The Robockly is very user friendly, very rich and it is easy to create and run robotics programs. It is powerful for robotic programs.

Visual block programming languages are limited for programs with many functions and procedures. The Python is a very simple and powerful high-level programming language that works on any computer platform.

Procedure:

1. Check the number appears on the white label of the back of the WIFI-203 module (i.e., 3233).
2. Plug the WIFI-203 on the Sense robot.
3. Plug the battery module into the Sense robot.
4. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
5. Enter your computer Wi-Fi network and select the neulog network (i.e., neulog3233).
6. Go to Chrome browser and surf to **wifi203.com**.
7. The following **Direct** mode screen should appear:

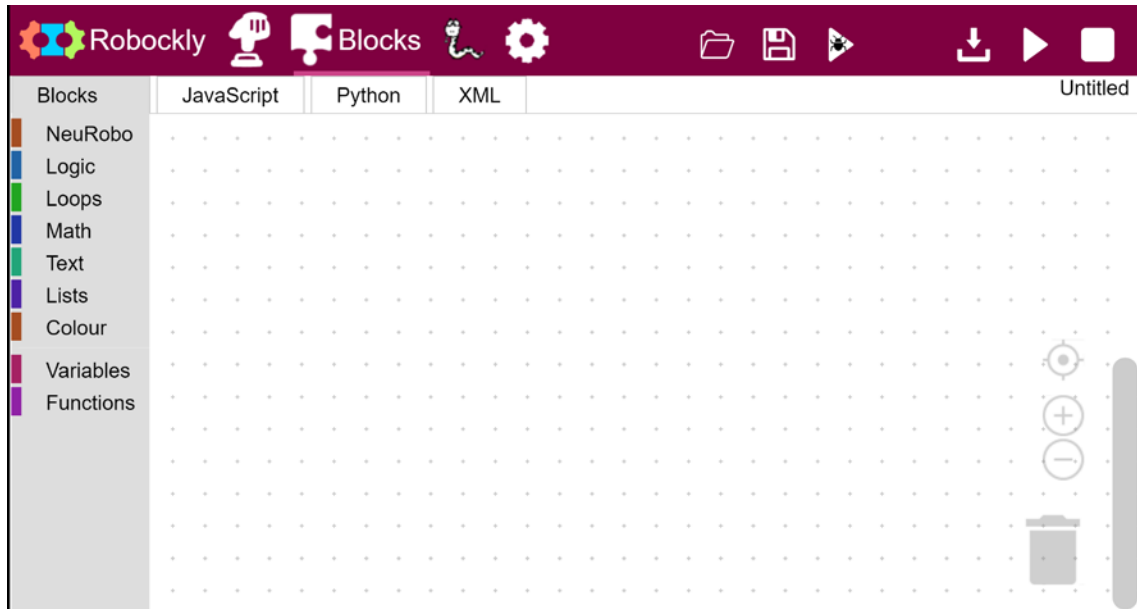


The direct mode enables you to test the system and the sensors' readings before programming and running the programs. This stage saves a lot of frustration in development.

8. Test the SENSE motors as described in experiment 1.1.

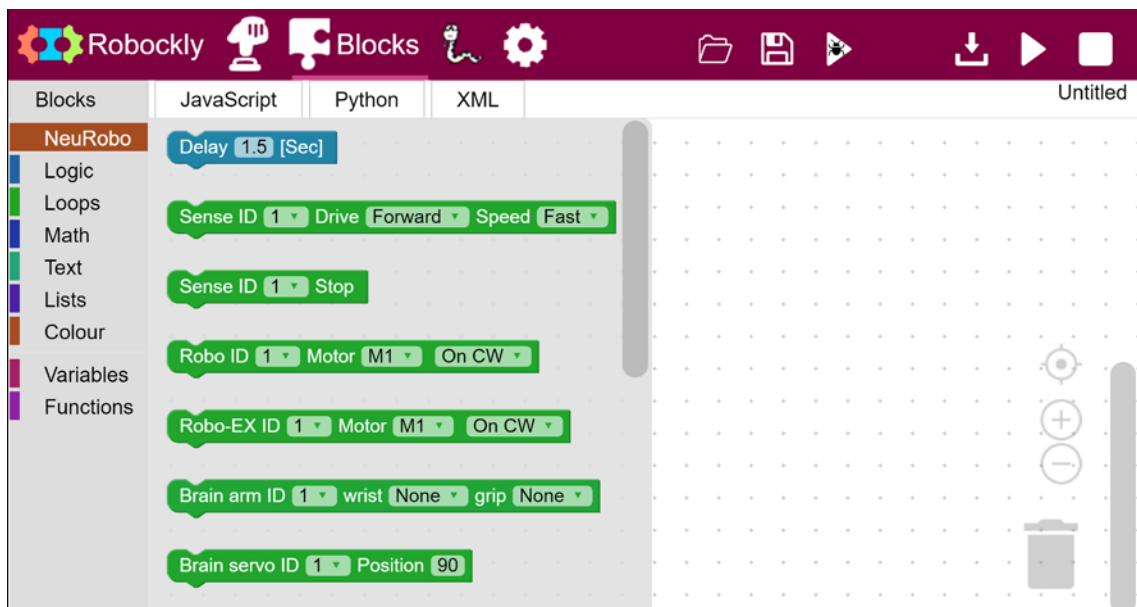
1.2.1 First program – forward

1. Move to **Block**  mode.

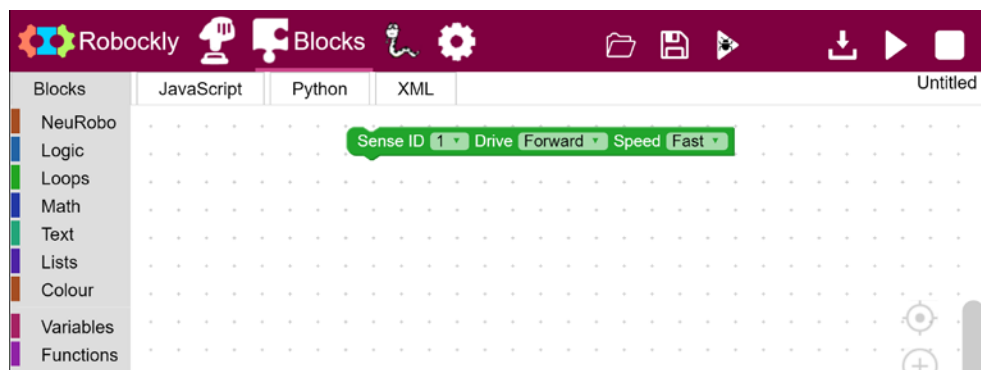


A computer program is composed of chains of instructions.

2. Click on the **NeuRobo** button and the following instruction list will appear:



- Click on the **SENSE Drive** instruction block and drag it to the screen.



- Click on the **NeuRobo** button and select the **Delay** instruction block.

Drag this block and attach it under the **Drive** instruction block.




- Change the delay value to 3 seconds.
- Click on the **NeuRobo** button and select the **Stop** instruction block.

Drag this block and attach it under the **Delay** instruction block.



The **Stop** instruction will be executed 3 seconds after the **Drive Forward** instruction.

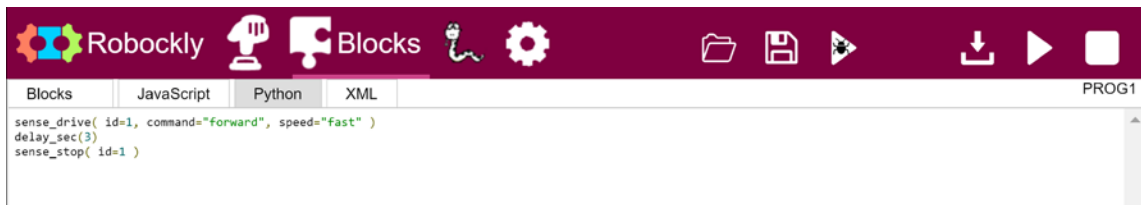
- Click on the **Save**  button and the following keyboard will appear.



Press 'X', type the **PROG1** and click **Done**.

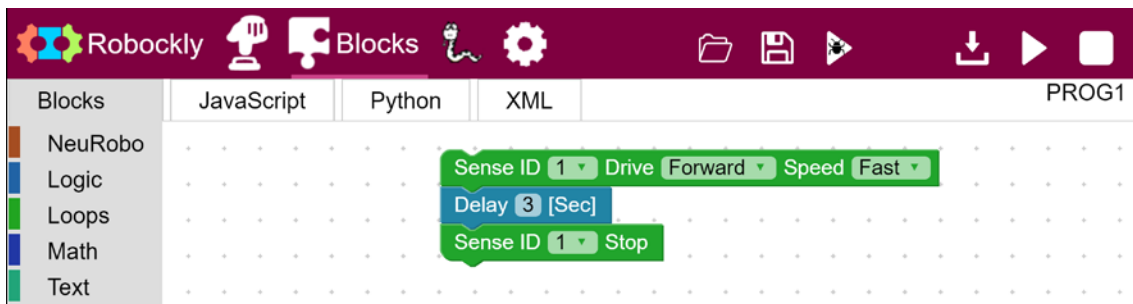
The program will be saved under the name PROG1.


8. Click on the **Python** menu button and observe the following screen.



This screen displays the Python program of the above Robockly program.

9. Return to the **Blocks** screen.



10. The **Debug**  button runs the program while highlighting the executed block.

Click on this button.

The SENSE will move forward for 3 seconds and stops, while the executed block is highlighted on the screen.

Note:

Until now, the programs (Robockly and Python) that we see are in the PC.

At **Debug** run, the PC sends the instructions one after the other to the SENSE while highlighting the executed screen.

1.2.2 Program download

1. We can download the program from the PC into the WIFI-203 memory.

Click on the **Program Download**  button.

This will transfer the PC program into the WIFI-203 flash memory under the name **PROG1**.

Note:

The WIFI-203 is a computer with Linux operating system.


Its flash memory is used as hard disk and the programs are saved as files.


2. Click on the **Run**  button.

The SENSE will move forward for 3 seconds and then stops.

While running, the LED on the WIFI-203 turns to red and blinks.

Note:

It takes about two seconds from clicking on the **Run**  button and the SENSE movement.

You can stop the program by clicking on the **Stop**  button on the right.

3. To run the program in the WIFI-203 memory, we can also use the **Start/Stop** orange pushbutton located on the WIFI-203 panel.

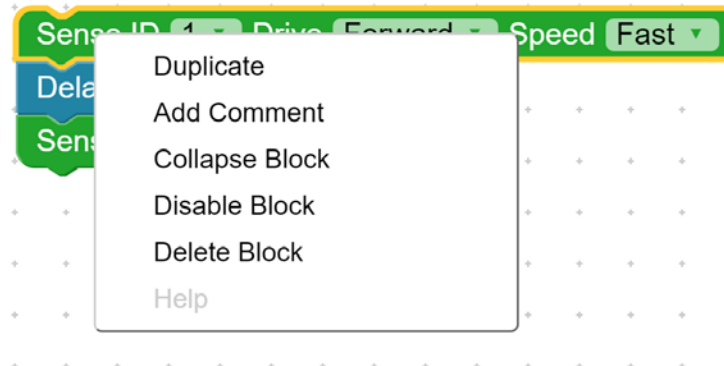
Press the WIFI-203 panel button and you will see the SENSE move forward for 3 seconds and then stop.

Note:

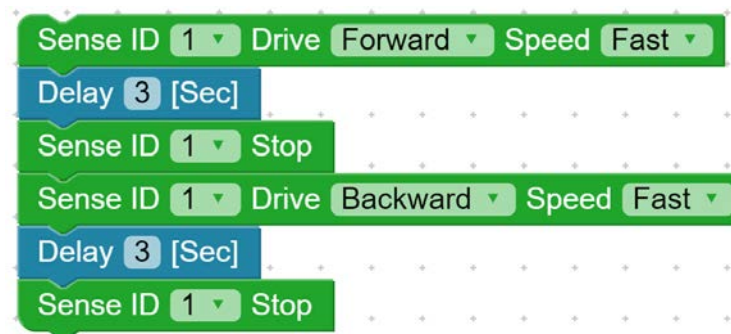
The WIFI-203 LED blinks in red while running.

1.2.3 Forward and backward

1. Right click on the **Drive** instruction block.






2. Select **Duplicate** and a new **Drive** instruction block appear. Attach it to the **Stop** instruction block.
3. Change the commands in the new **Drive** instruction block to **Backward**.
It is better to stop a motor before changing its rotation direction.
4. Duplicate the **Delay** and the **Stop** instruction blocks.
5. Drag the instruction blocks and build the following program:



6. Click on the **Python** menu button and observe the following screen.

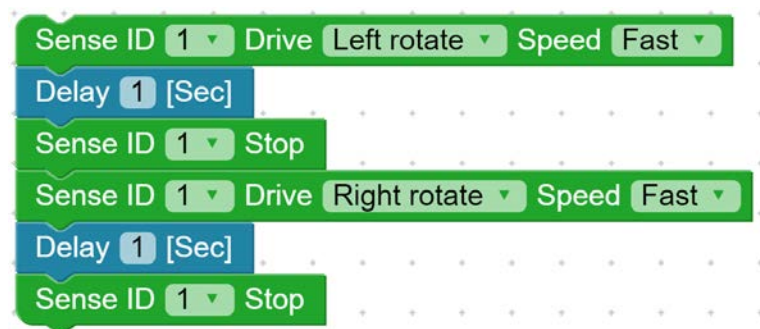


7. Click on the **Save**  button and save the program under the name **PROG2**.
8. Click on the **Program Download**  button.
9. Click on the **Run**  button.

The SENSE moves forward for 3 seconds, moves backward for 3 seconds and then stops.

1.2.4 Turning left and right



1. Change the program to the following one:



```
sense_drive( id=1, command="left_rotate", speed="fast" )
delay_sec(1)
sense_stop( id=1 )
sense_drive( id=1, command="right_rotate", speed="fast" )
delay_sec(1)
sense_stop( id=1 )
```

Pay attention to the **Drive** instructions.

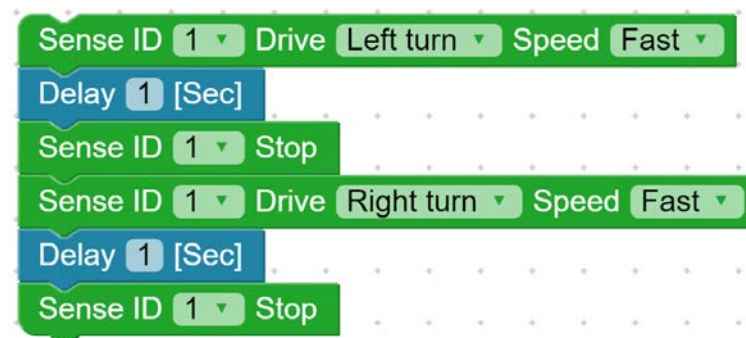
The delays are changed to 1 second.

2. Click on the **Save**  button and save the program under the name **PROG3**.
3. Click on the **Program Download**  button.


4. Click on the **Run**  button.

The SENSE rotates to the left for 1 second, rotates to the right for 1 second, and then stops.

5. Change the delay time for having turns of 90°.
6. Change the program to the following one:



```
sense_drive( id=1, command="left_turn", speed="fast" )
delay_sec(1)
sense_stop( id=1 )
sense_drive( id=1, command="right_turn", speed="fast" )
delay_sec(1)
sense_stop( id=1 )
```

7. Click on the **Save**  button and save the program under the name **PROG4**.

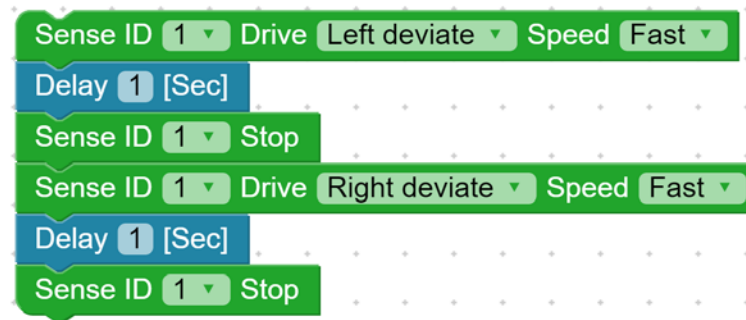
8. Click on the **Program Download**  button.

9. Click on the **Run**  button.


The SENSE turns to the right for 1 second, turns to the left for 1 second, and then stops.

10. Change the delay time for having turns of 90°.

11. Change the program to the following one:



```
sense_drive( id=1, command="left_deviate", speed="fast" )
delay_sec(1)
sense_stop( id=1 )
sense_drive( id=1, command="right_deviate", speed="fast" )
delay_sec(1)
sense_stop( id=1 )
```

12. Click on the **Save**  button and save the program under the name **PROG5**.

13. Click on the **Program Download**  button.

14. Click on the **Run**  button.

The SENSE deviates to the right for 1 second, deviates to the left for 1 second, and then stops.

What is the difference between the behaviors of the three rotating programs?

1.2.5 Challenge exercises – Moving in a square

Task 1: Change the **Drive** instructions in the program to the **Deviate** instructions.

Observe the SENSE movement.

Task 2: Make a program that moves the SENSE in a 30x30 cm square until it returns to its original place.

Use the **Rotate** instructions for rotating.

Task 3: Make a program that moves the SENSE in a 30x30 cm square until it returns to its original place.

Use the **Turn** instructions for rotating.

Experiment 1.3 – Interactive Programs

Objectives:

- Program that reacts to sensors
- Using variables
- Moving the SENSE between lines
- Moving the SENSE between line and a wall

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module

Discussion:

In this experiment, we will move the SENSE between two black lines. The position of the lines limits its motion. The robot changes direction when it finds a black line. This is an example of a system called a Manipulator.

We will learn how to read and react to the Line and the Front Range sensors.

A closed loop system is a control system, which reacts to sensors and switches.

An example for closed loop system is a control system that lights up a lamp when it is dark, and turn it OFF when there is light. This system is automatically adapted to summer time (when the night is short) and to wintertime (when the night is long and starts early).

The program of closed loop system contains decision instructions such as:

'Repeat – until', 'Repeat while', 'If – then'.

The programs in this experiment use the **'Repeat – until'** instruction.

1.3.1 Variables

In the delay instruction, we write a number that determines the length of the delay in seconds.

We call a number that is part of the instruction a **constant**. When we want to change this number, we have to search for the relevant instruction.

In programming, we prefer to use variables instead.

A **variable** is a memory cell with a name. In the instruction we indicate the name of the memory cell.

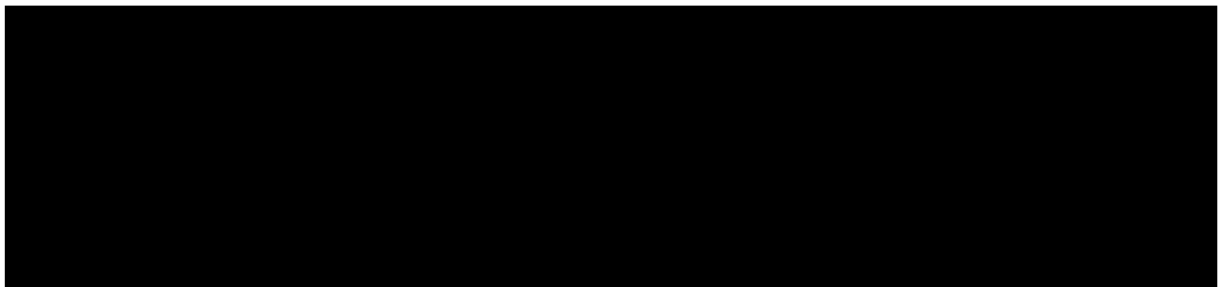
We can set the variable value in a certain place of the program, which saves us the searching for instructions.

Another type of a variable is a variable that relates to one of the SENSE inputs.

This variable gets its value each time it is used or called in the program, as we shall see later.

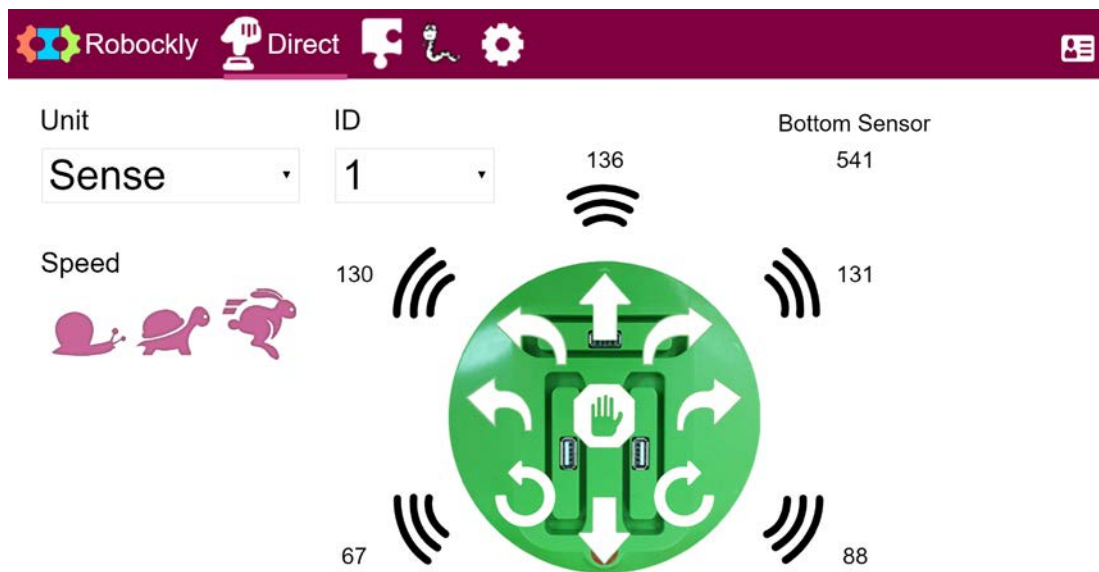
The software includes special instruction for changing variable values according to program conditions.

Before starting the experiment, print two black lines as follows:



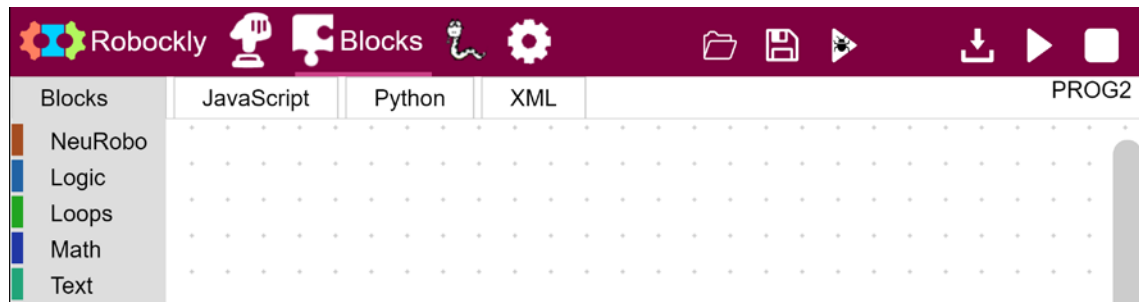
Procedure:

1. Check the number appears on the white label of the back of the WIFI-203 module (i.e., 3233).
2. Plug the WIFI-203 on the Sense robot.
3. Plug the battery module into the Sense robot.
4. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
5. Enter your computer Wi-Fi network and select the neulog network (i.e., neulog3233).
6. Go to Chrome browser and surf to **wifi203.com**.
7. The following **Direct** mode screen should appear:




8. Test the SENSE motors as described in experiment 1.1.
9. Test the SENSE Bottom sensor as described in experiment 1.1.
10. Record the values of the Bottom sensor when the SENSE is on a white surface. We shall call this value White Value.
11. Record the values of the Bottom sensor when the SENSE is on the black line. We shall call this value Black Value.

12. Move to **Block**  mode.




13. Create the following program, which moves the SENSE forward for 3 seconds and stops.



14. Click on the **Save**  button and save the program under the name **CART1**.

15. Click on the **Program Download**  button.

16. Place the SENSE on the table or on the floor and click on the **Run**  button.

The SENSE will move forward for 3 seconds and then stop.

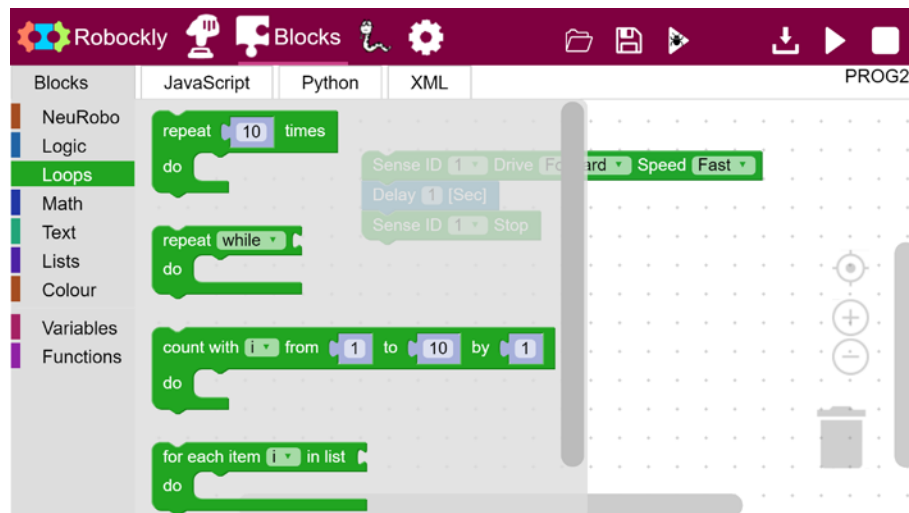
17. Press the WIFI-203 panel button and you will see the robot move forward for 3 seconds and then stop.

The WIFI-203 LED blinks in red while running.

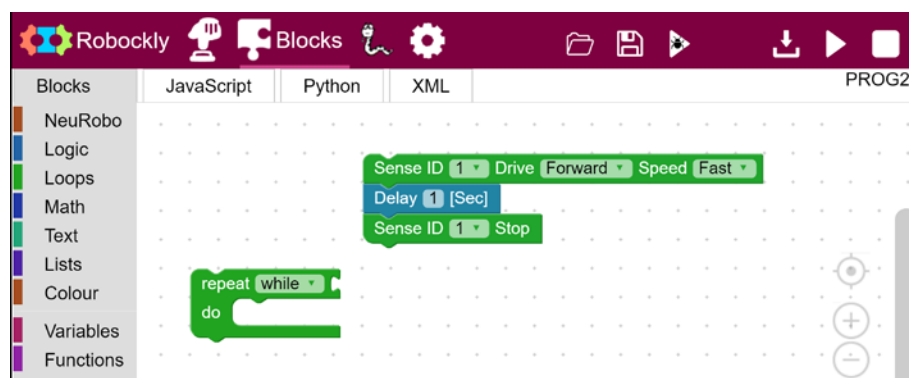
1.3.2 Repeat until

We shall replace the delay instruction with **Repeat until** instruction.

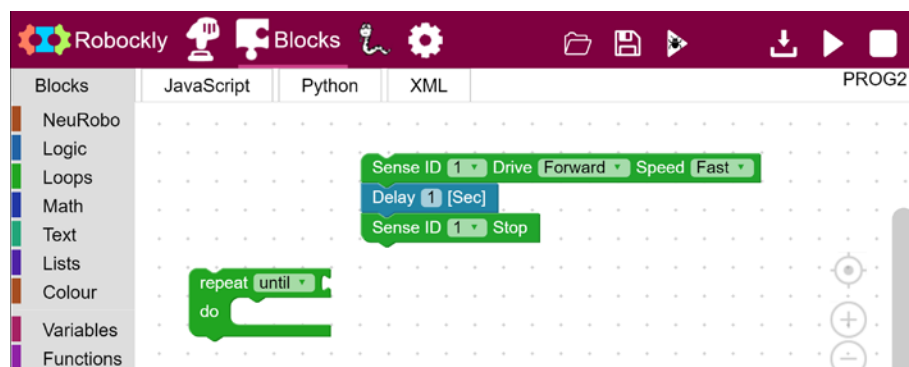
1. Click on the **Loops** button and a list of loop instructions appear:



2. Click on the '**Repeat while**' instruction block.

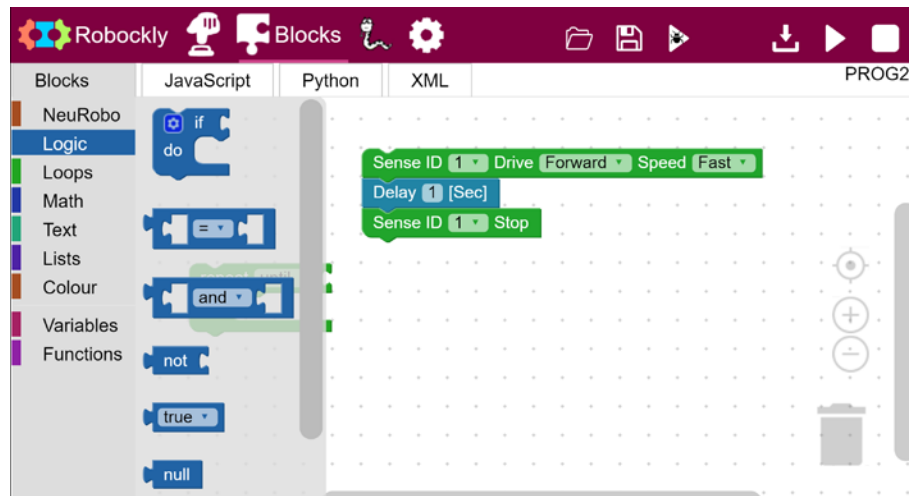


3. Change the **while** in the block to **until**.

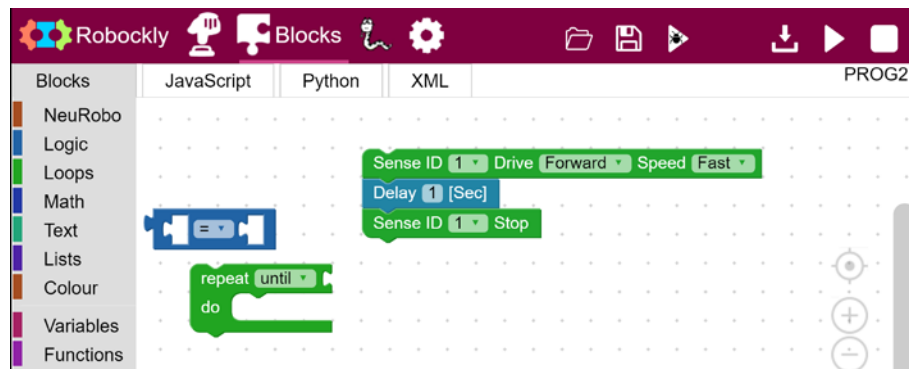


4. This instruction should have a condition statement on the right.

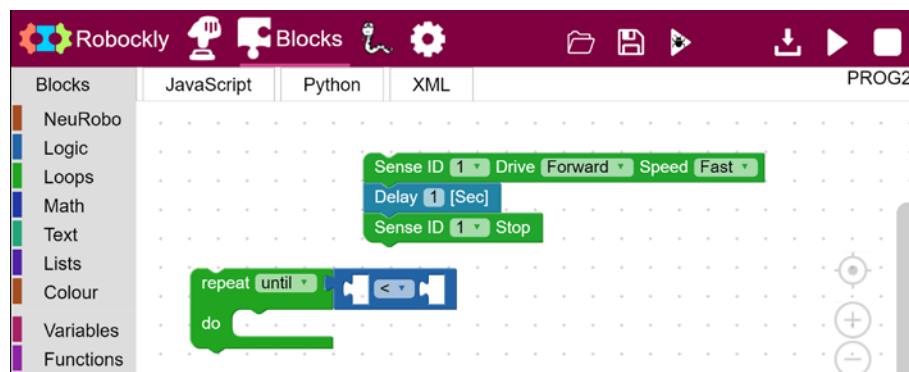
Click on the **Logic** button and a list of logic instructions appear:



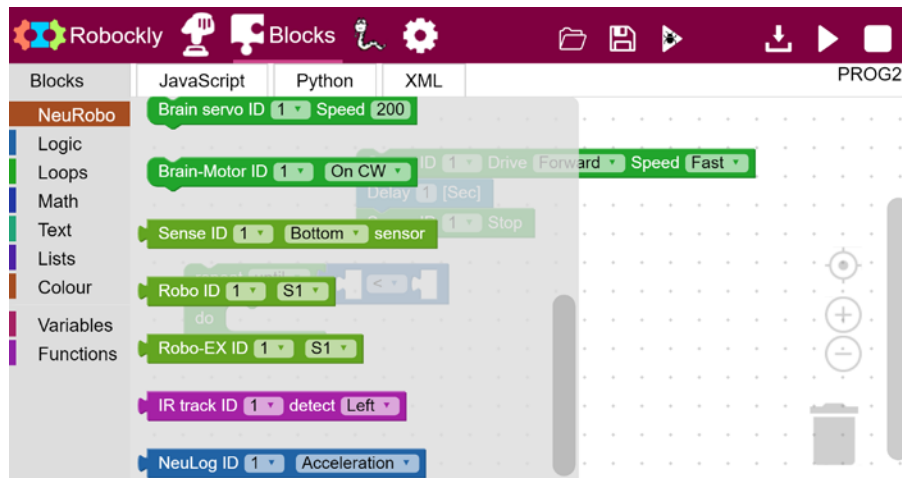
5. Click on the '**Comparison**' instruction block.



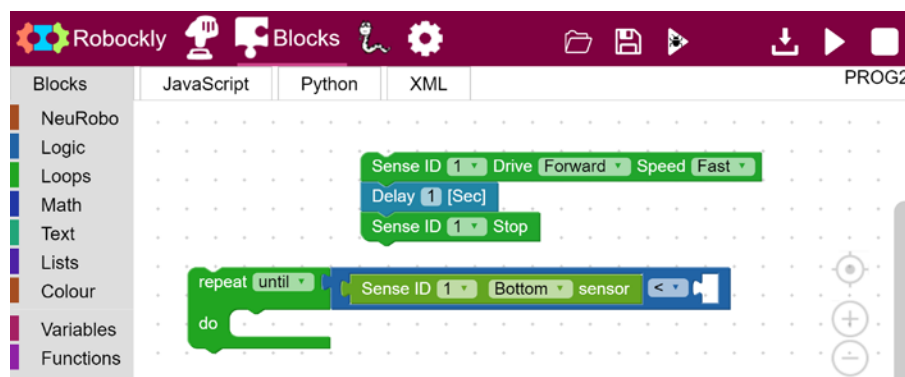
6. Drag the **Comparison** block to the **Repeat until** block and change the comparison sign to < (below).



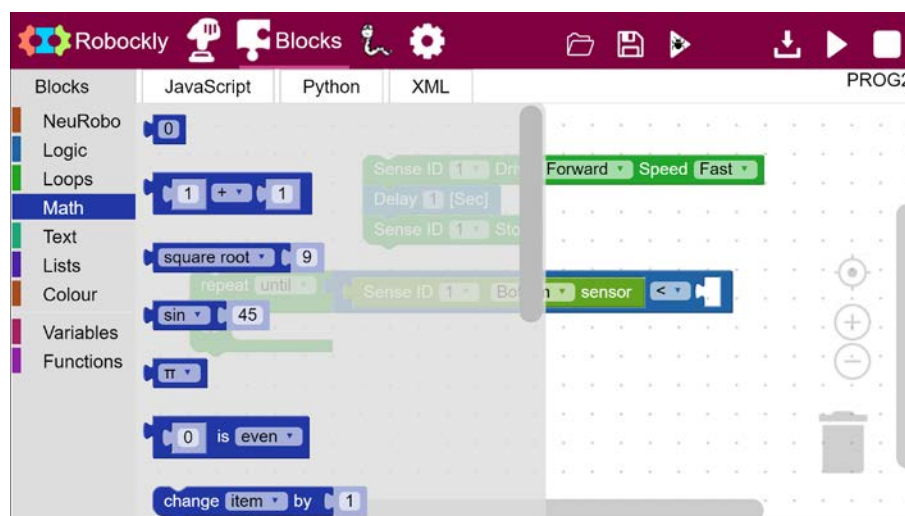
7. Click on the **NeuRobo** button and a list of NeuRobo instructions appear. Scroll down until you have the following:



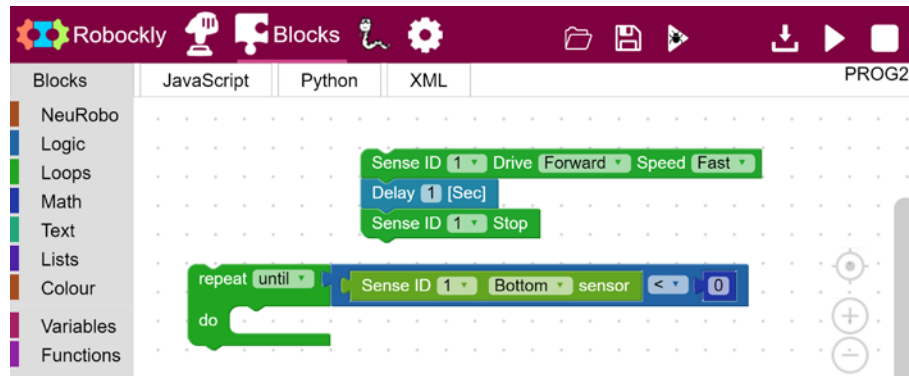
8. Select the **Sense sensor** block and drag it into the **comparison** block.



9. Click on the **Math** button and a list of mathematic instructions appear.



10. Select the **Constant value** block and drag it into the **comparison** block.

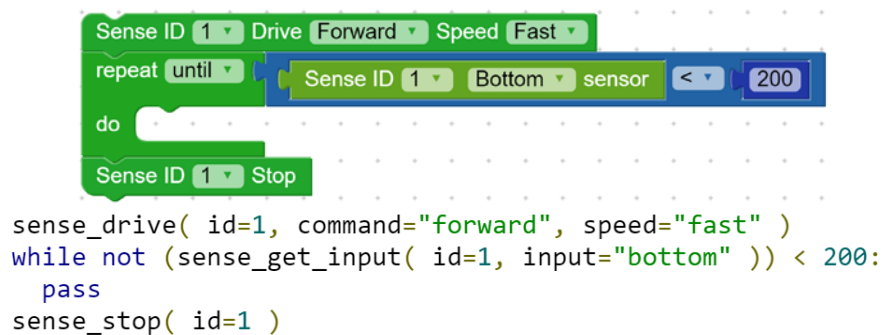


11. Change the number in the instruction to a value that is 50 above the **Black Value** (i.e. 200).

We have now a loop instruction that will wait until the value of the SENSE bottom will be below 200.

While waiting, the program executes all the instructions in the loop block. If there is not any instruction, it just waits.

12. Drug the **Repeat until** instruction below the **Sense** drive instruction and delete the **Delay** instruction block and check that you get the following:



Note:

The **Repeat until** is the opposite condition of the **Repeat while**.

The Python program uses the **while** instruction with **not** for the condition.

13. Place the SENSE on a white surface with a black line on it.
14. Run the program.

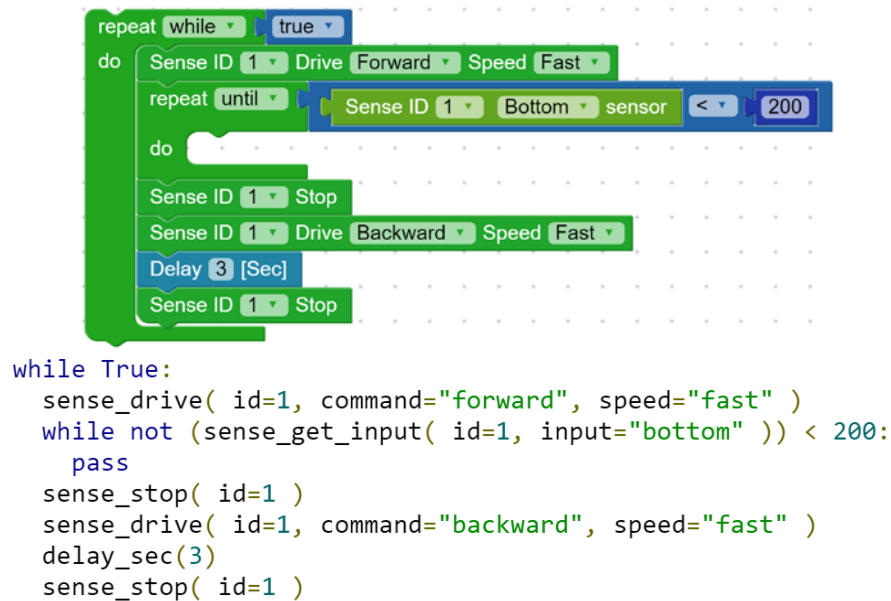
The SENSE should move forward and stop when it reaches the black line.

1.3.3 Endless loop

Most of the control and robotic programs are programs that run in endless loop.

1. Change the program so that the SENSE goes forward and stops when it meets the black line, goes back for 3 seconds, and forward again in endless loop.

We must add an endless loop instruction to the program as follows:




We added the instruction **Repeat while true**.

This instruction should run endlessly because nothing will change this condition.

We took the **true** constant from the **Logic** instruction set.

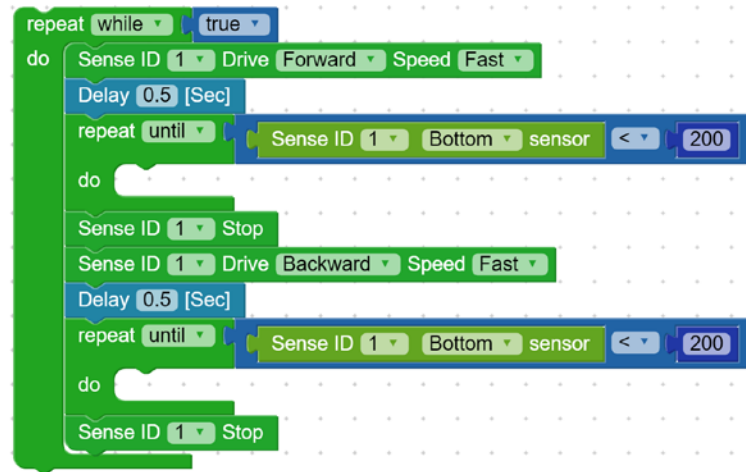
2. Place the SENSE on a white surface with a black line on it.
3. Run the program.

The SENSE should move forward until it reaches the black line, stop, then go backward for 3 seconds and then forward again.

4. Stop the program by pressing the SENSE pushbutton or by clicking the **Stop**  button, if the SENSE is connected to the computer.

1.3.4 Movement between two lines

1. Change the program so the SENSE will move between two lines.



```
while True:
    sense_drive( id=1, command="forward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < 200:
        pass
    sense_stop( id=1 )
    sense_drive( id=1, command="backward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < 200:
        pass
    sense_stop( id=1 )
```

Pay attention to the **Delay** instructions.

They are added in order to be sure that the robot will move from the current black line and wait before moving to the other black line.

2. Save the program under the name **CART2**.
3. Place the SENSE between two black lines and run the program.

The SENSE should run back and forth between the lines.

Increasing the distance between the lines changes the SENSE's travel accordingly.

1.3.5 Challenge exercise – Between a wall and a line (I)

Task 1: Improve the CART2 program so that the SENSE will move between a wall in front and a black line at the back.

Note:

You have to use the Front sensor while moving forward. Take care for the compare sign (> or <).

Experiment 1.4 – Functions and Variables

Objectives:

- Using functions in a program
- Using variables in a program
- Reacting to sensors

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module


Discussion:

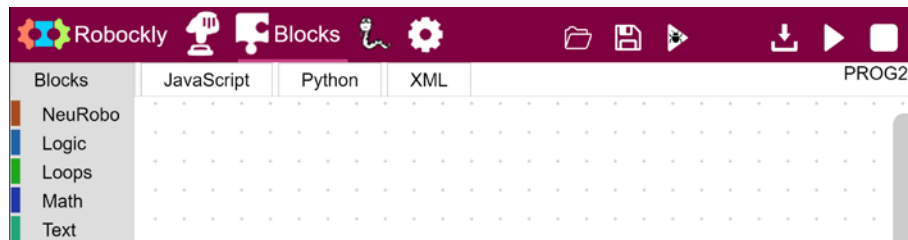
A computer program is composed of chains of instructions.


Instead of having a single chain of instructions, we can divide the program to functions, which are short chains and give a name for each chain.

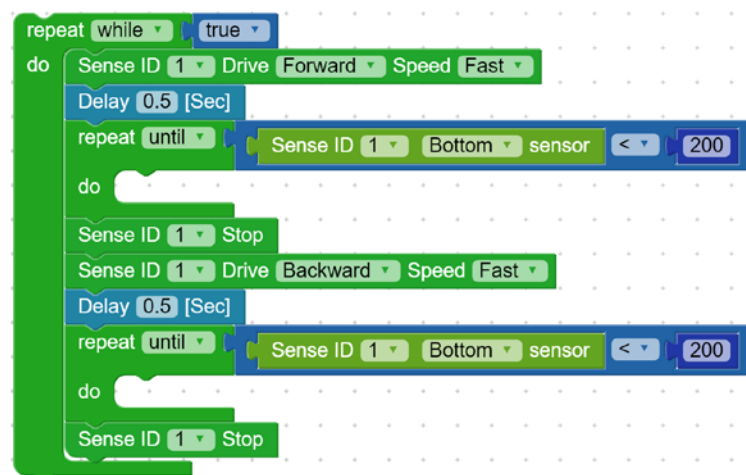
In this experiment, we shall build and use movement functions.

Procedure:

1. Plug the WIFI-203 and BAT-202 on the Sense robot.
2. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
3. Select the neulog Wi-Fi network in your computer (i.e., neulog3233).
4. Browse to **wifi203.com**.
5. Move to **Block**  mode.



6. Click on the **Open**  button and open the program **CART2**.
7. Check that you have the following program:



```
while True:
    sense_drive( id=1, command="forward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < 200:
        pass
    sense_stop( id=1 )
    sense_drive( id=1, command="backward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < 200:
        pass
    sense_stop( id=1 )
```

If not, build this program and save it under the name **CART2**.

8. Place the SENSE between two black lines and run the program.

The SENSE should run back and forth between the lines.

Increasing the distance between the lines changes the SENSE's travel accordingly.

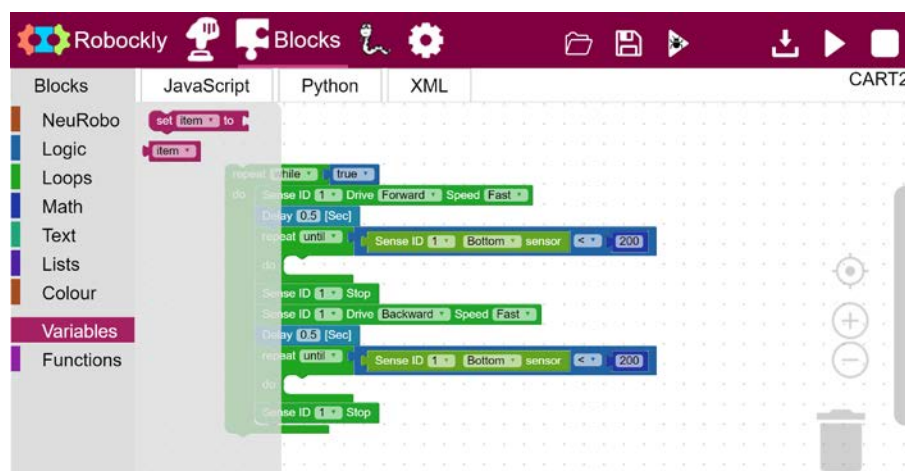
9. Stop the program.
10. Drag the program to the **Trash box** in order to clear the screen.

1.4.1 Variable instead of constant

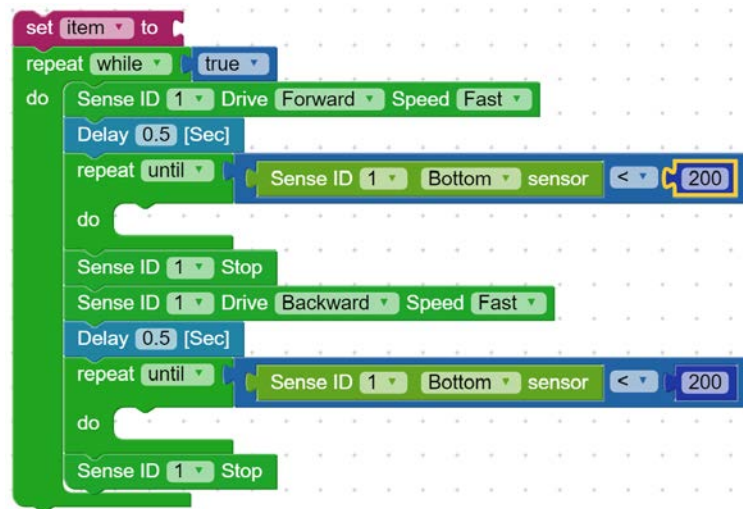
1. In the program, we compare the values of the bottom sensor twice with a number.

This number is a constant. If we want to change it, we have to look for the instructions that use it.

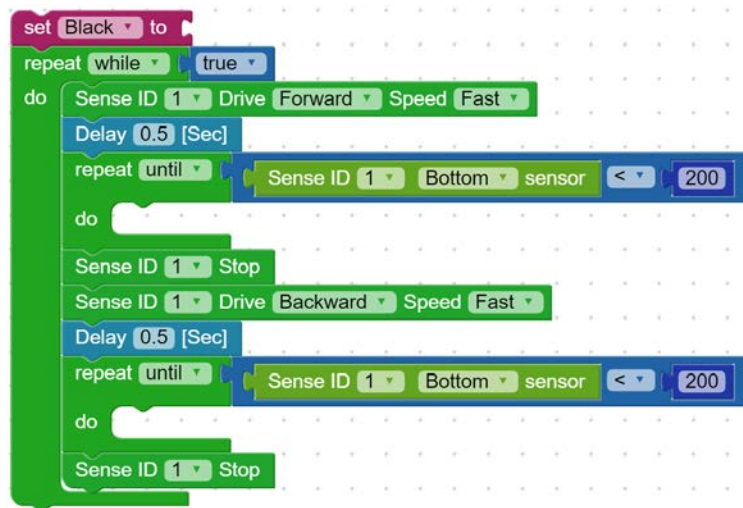
Click on the **Variables** button and the following screen appears:



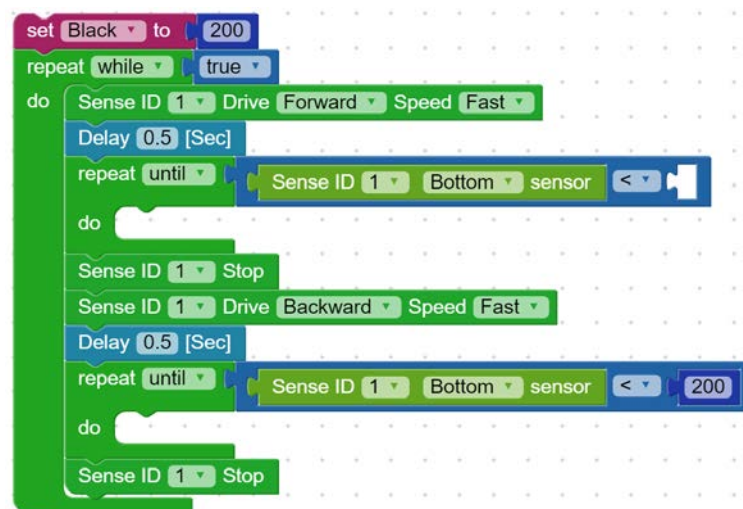
2. Click on the **set** block and drag it as the following:



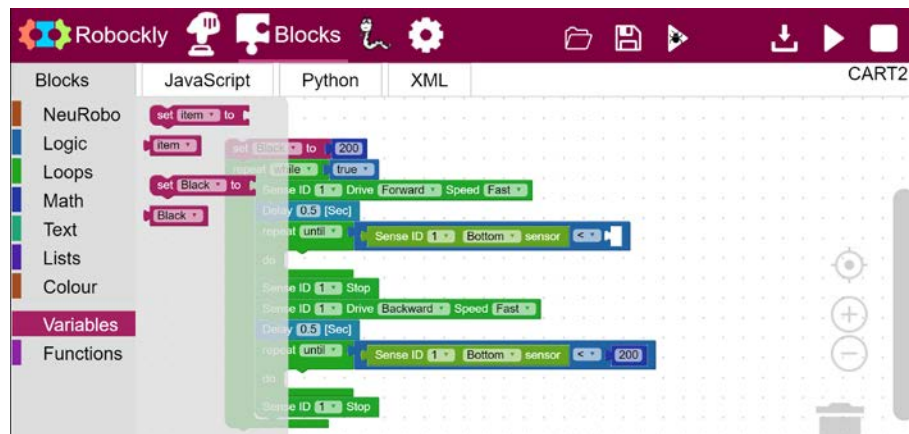
3. Use the drop down arrow and rename the variable to **Black**.



4. Drag the constant 200 to the **set** block.

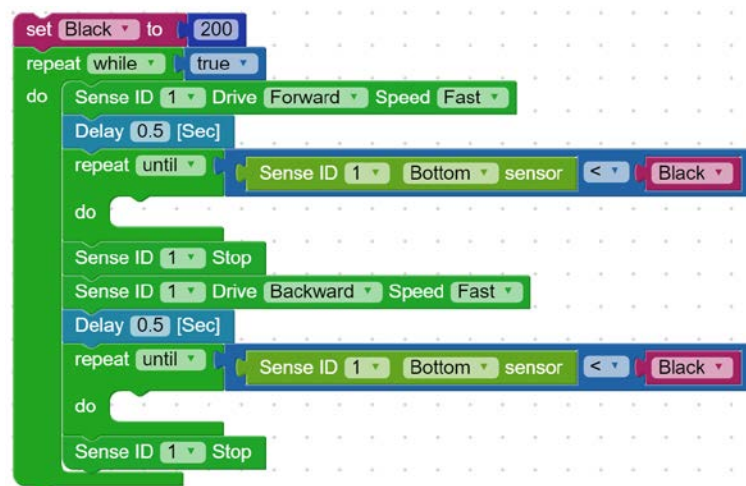


5. Click on the **Variables** button and the following screen appears:



6. The black variables appear in this screen.

Drag the **Black** variable into the program instead of the constants.



Black = None

```
Black = 200
while True:
    sense_drive( id=1, command="forward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < Black:
        pass
    sense_stop( id=1 )
    sense_drive( id=1, command="backward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < Black:
        pass
    sense_stop( id=1 )
```

7. Save the program under the name **CART3**.

8. Place the SENSE between two black lines and run the program.

The SENSE should run back and forth between the lines.

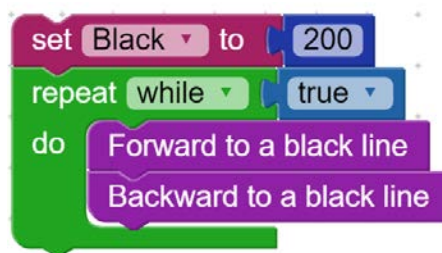
Increasing the distance between the lines changes the SENSE's travel accordingly.

9. Stop the program.

1.4.2 Functions

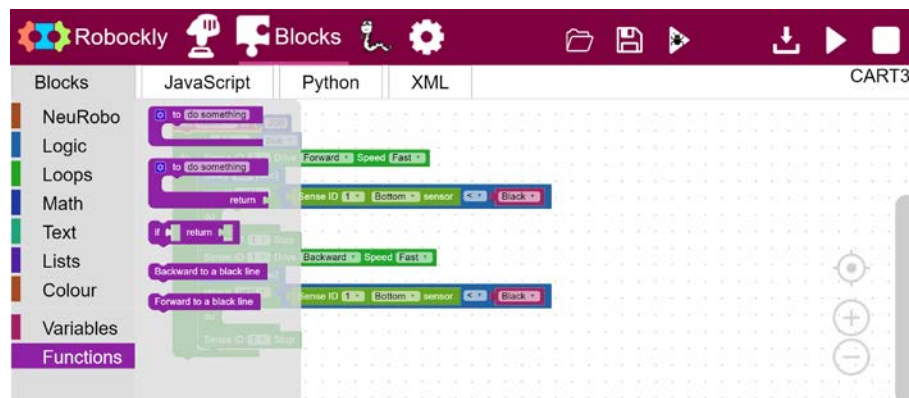
Now, we know exactly what CART3 does and what the purpose of every instruction is. In two weeks from now, we will remember any and it will take time to understand the program.

A main program like the following is much clearer.



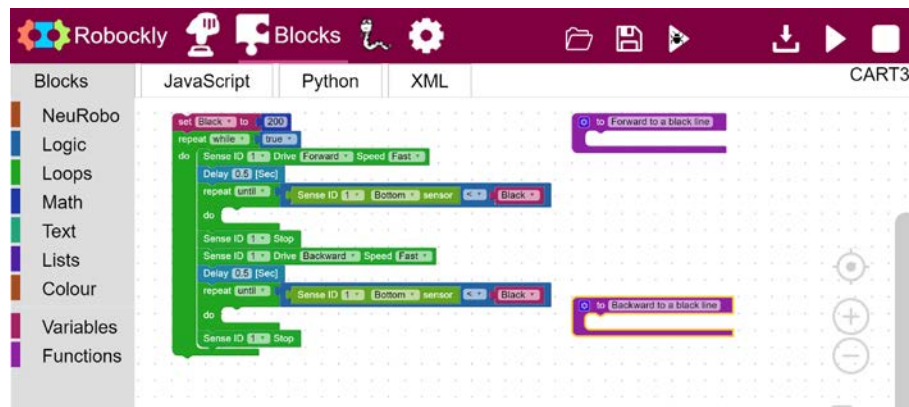
This main program uses two functions with names that explain what they doing.

1. Click on the **Functions** button and the following screen appears:

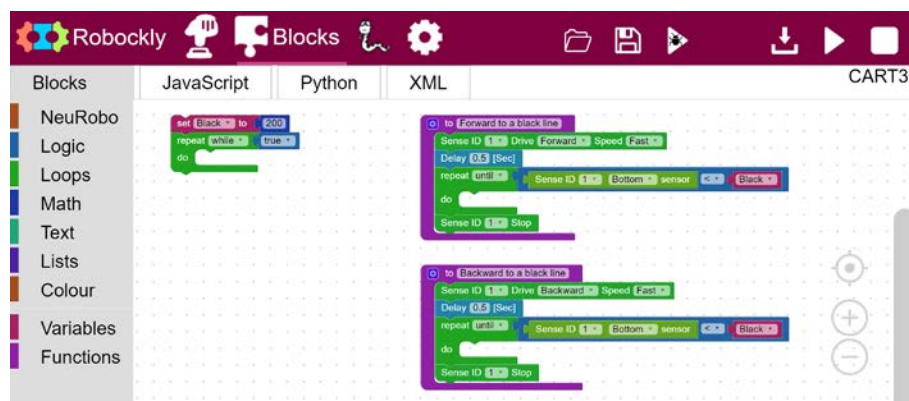


2. Drag the upper function (the one without the return) to the right.
3. Click on its name field and change it to **Forward to a black line**.

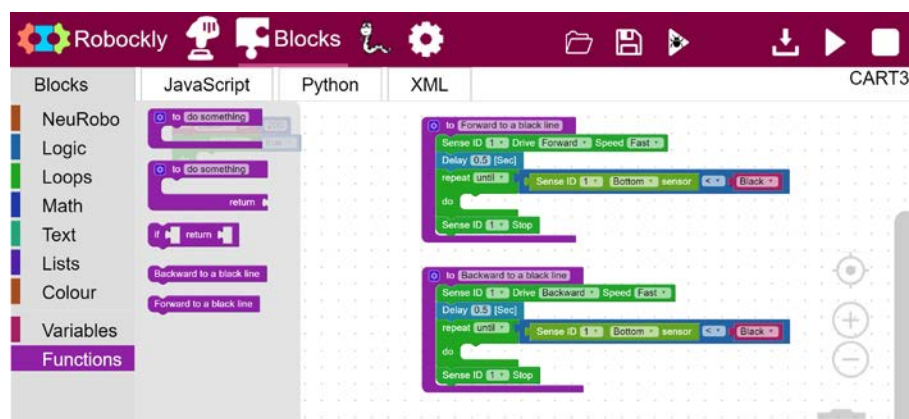
4. Duplicate this function and change its name to **Backward to a black line**.
5. Check that you have the following screen:



6. Drag the last four blocks of the endless loop into the **Backward** function.
7. Drag the first four blocks of the endless loop into the **Forward** function.
8. Check that you have the following screen.

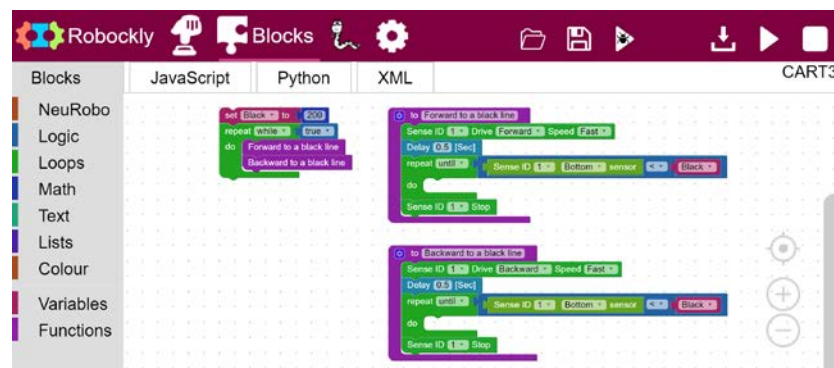


9. Click on the **Functions** button and the following screen appears:



The **Forward** and the **Backward** functions appear in the **functions** list.

10. Drag them into the endless loop as in the following screen.



```
Black = None
```

```
def Forward_to_a_black_line():
    global Black
    sense_drive( id=1, command="forward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < Black:
        pass
    sense_stop( id=1 )

def Backward_to_a_black_line():
    global Black
    sense_drive( id=1, command="backward", speed="fast" )
    delay_sec(0.5)
    while not (sense_get_input( id=1, input="bottom" )) < Black:
        pass
    sense_stop( id=1 )
```

```
Black = 200
while True:
    Forward_to_a_black_line()
    Backward_to_a_black_line()
```

11. Observe the python program.

It starts with the definition of the variable **Black** with the value none.

Next are the definitions of the two functions **Forward** and **Backward**.

The main program comes at the end and calls already defined variable and functions.

12. Save the program under the name **CART4**.
13. Place the SENSE between two black lines and run the program.

The SENSE should run back and forth between the lines.

14. Stop the program.

1.4.3 Challenge exercise – Between a wall and a line (II)

Task 1: Improve the CART4 program so that the SENSE will move between a wall in front and a black line at the back.

Run and check.

1.4.4 Moving along a black line

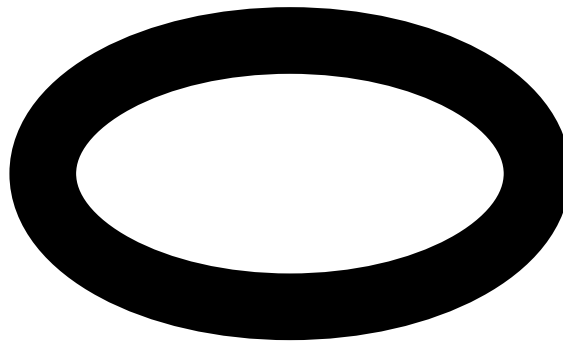
To move the SENSE along a black line we use slow turn procedures of the SENSE.

In slow turns, one wheel rotates and the other wheel stops. This way the SENSE still moves forward while turning.

In the main program, we do the movement according to the following idea:

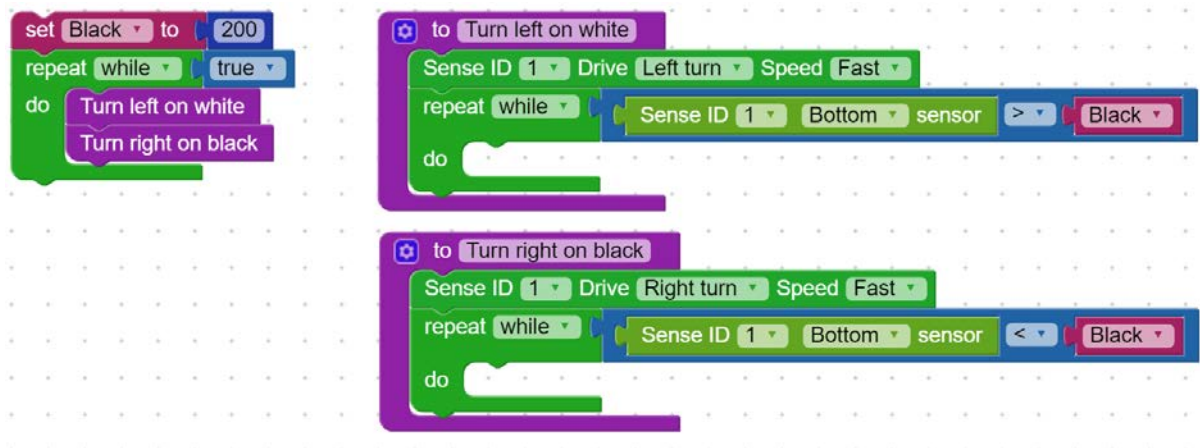
Turning left until the SENSE find a black surface, and then turning right until the SENSE find a white surface.

1. Print on a full page a black line as the following:



The width of the line should be at least 4cm.

2. Change the program and the functions as follows:



```
Black = None
```

```
def Turn_left_on_white():
    global Black
    sense_drive( id=1, command="left_turn", speed="fast" )
    while (sense_get_input( id=1, input="bottom" )) > Black:
        pass
```

```
def Turn_right_on_black():
    global Black
    sense_drive( id=1, command="right_turn", speed="fast" )
    while (sense_get_input( id=1, input="bottom" )) < Black:
        pass
```

```
Black = 200
while True:
    Turn_left_on_white()
    Turn_right_on_black()
```

Note:

Pay attention to the compare signs (< and >).

3. We changed the instructions **Repeat until** to **Repeat while**.

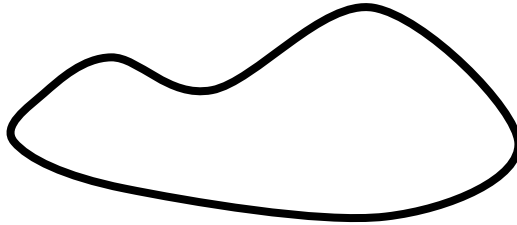
Explain why.

4. Save the program under the name **CART5**.
5. Put the SENSE near the black line.
6. Run and check the SENSE movement.
7. Change the value of the **Black** variable to create smooth movement of the SENSE.

1.4.5 Challenge exercise – Along a complex black line

Task 1: Create different black lines for the SENSE and check its behavior. Improve the programs when needed.

Example of a complex line:



Note:

In order to duplicate a set of instruction, click on the first instruction, move it a little with the mouse, keep pressing the mouse button and click **Ctrl+c**. Now click **Ctrl+v** and this will duplicate the set of instructions.

Experiment 1.5 – Conditions and Decisions

Objectives:

- The IF instruction
- OR condition
- AND condition

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module

Discussion:

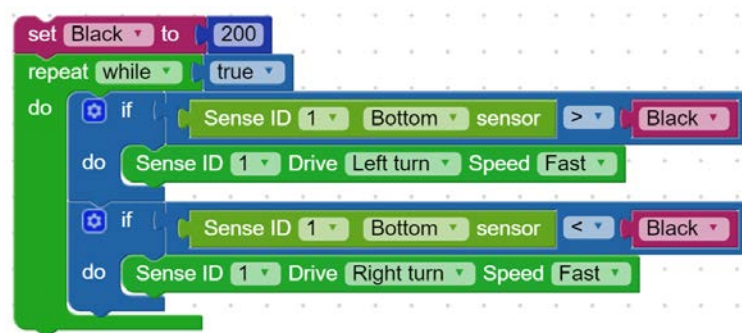
1.5.1 The If instruction

In the previous experiment, we learned about the **Repeat until** instruction. This is one of the condition instructions.

The **If** instruction is the main condition instruction. It is composed of a condition and a procedure to operate when the condition exists.


The condition is checked when the condition instruction is executed. If the condition does not exist, the following instruction is executed.

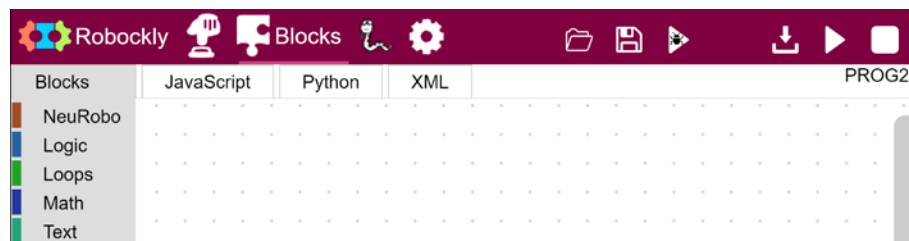
The following program is a main procedure for moving along black line using **If** instructions.




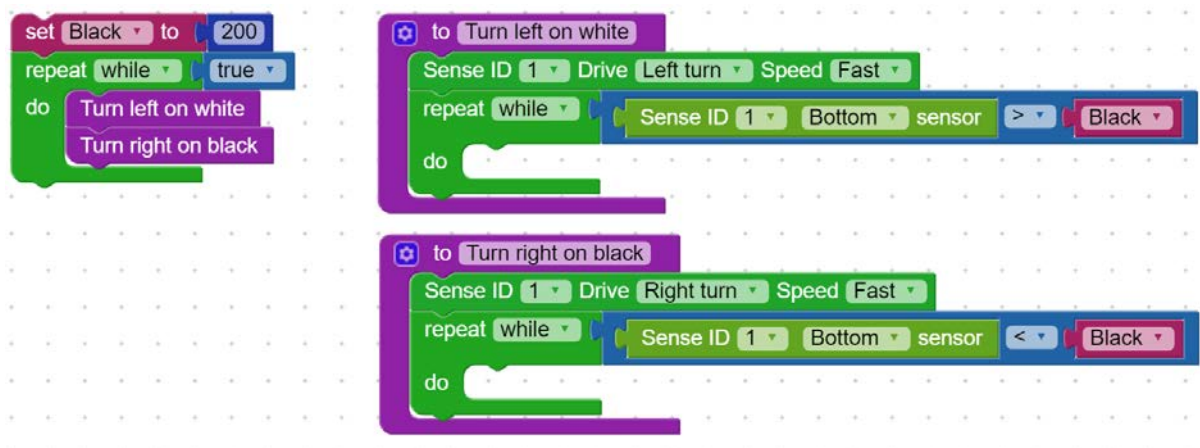
We can create complex conditions, called AND condition and OR condition, explained in the experiment later.

Procedure:

1. Plug the WIFI-203 and BAT-202 on the Sense robot.
2. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
3. Select the neulog Wi-Fi network in your computer (i.e., neulog3233).
4. Browse to **Wi-Fi203.com**.
5. Move to **Block**  mode.

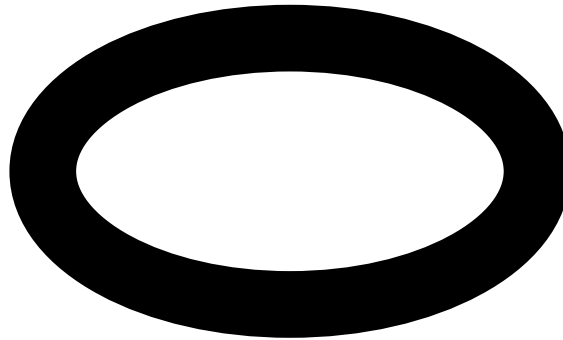


6. Click on the **Open**  button and open the program **CART5**.
7. Check that you have the following program:



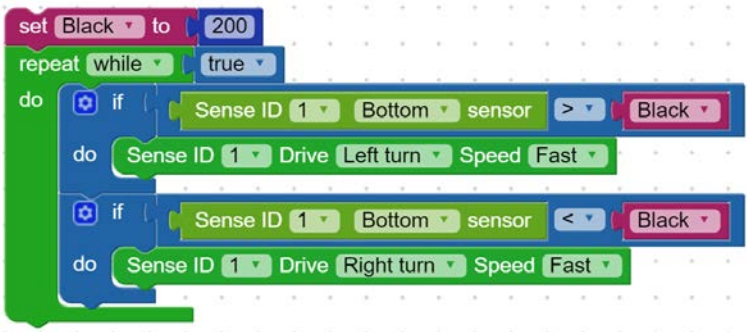
If not, build this program and save it under the name CART5.

8. Place the SENSE on the black line circle and run the program.



The SENSE should go along the black line.

9. Stop the program.
10. Change the program to the following:



```

Black = None

Black = 200
while True:
    if (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
  
```

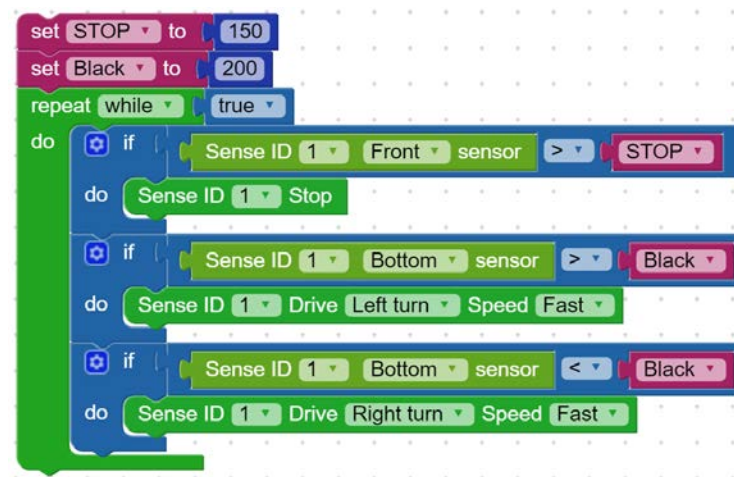
The **If** instruction is in the **Logic** instruction list.

11. Save the program under the name **CART6**.
12. Put the SENSE near the black line.
13. Run and check the SENSE movement.

14. Change the value of the **Black** variable to create smooth movement of the SENSE.

We shall improve the CART6 program so the SENSE stops when you put your hand in front of it.

15. Change the program to the following:



STOP = None
Black = None

```
STOP = 150
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP:
        sense_stop( id=1 )
    if (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
```

We added another variable that relates to the front wall sensor.

In every cycle, the main procedure checks the distance from the wall and calls the **STOP** instruction when the SENSE is close to the wall.

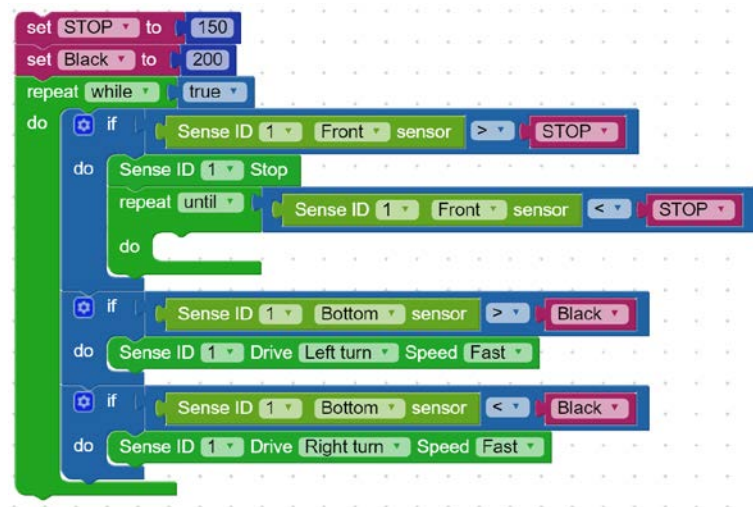
16. Put the SENSE near the black line.
17. Run and check the SENSE movement.
18. Put your hand in front of the SENSE, while it moves.

Instead of stopping, the SENSE just slows down.

Think why.

19. We have to improve the **STOP** decision to wait until you take off your hand.

Change the **STOP** decision to the following:



STOP = None
Black = None

```
STOP = 150
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP:
        sense_stop( id=1 )
        while not (sense_get_input( id=1, input="front" )) < STOP:
            pass
    if (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
```

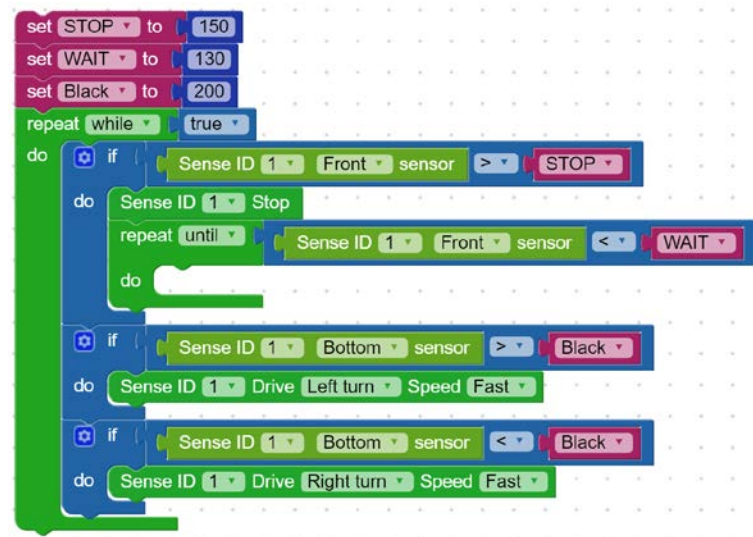
20. Save the program under the name **CART7**.
21. Put the SENSE near the black line.
22. Run and check the SENSE movement.
23. Put your hand in front of the SENSE, while it moves.

Change the values of the variables until the SENSE works well.

1.5.2 OFF and ON with different values

In control systems, we usually prefer that the OFF condition value will be different from the ON condition value. The reason is that we want to avoid having the system "bounce".

1. Change the program and procedures to the following:



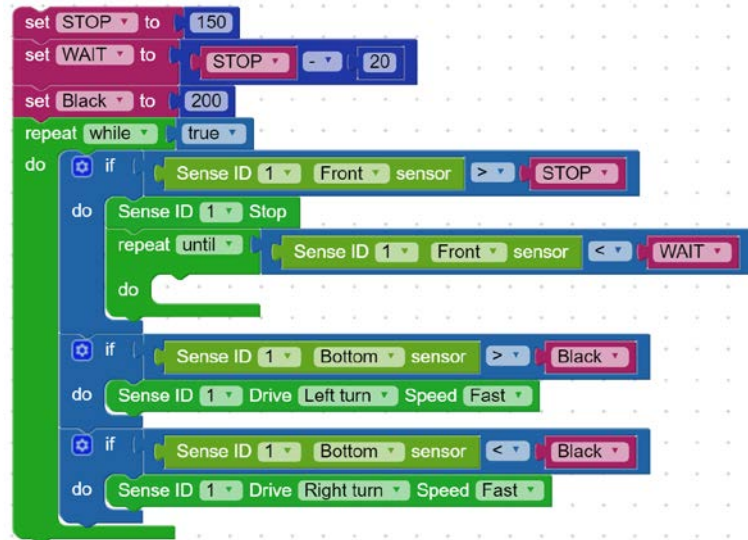
STOP = None
WAIT = None
Black = None

```
STOP = 150
WAIT = 130
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP:
        sense_stop( id=1 )
        while not (sense_get_input( id=1, input="front" )) < WAIT:
            pass
    if (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
```

2. The **STOP** value is higher than the **WAIT** value.
3. Run and test this program.

4. When we change the **STOP** value, we have to change the **WAIT** value.

The following program makes sure that the **WAIT** value will always be lower than the **STOP** value.



```
STOP = None
WAIT = None
Black = None
```

```
STOP = 150
WAIT = STOP - 20
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP:
        sense_stop( id=1 )
        while not (sense_get_input( id=1, input="front" )) < WAIT:
            pass
    if (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
```

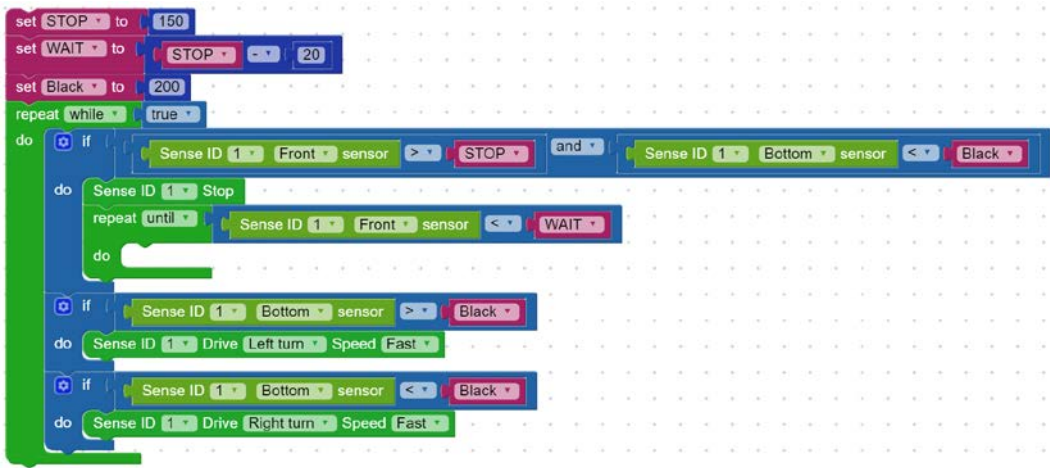
5. Run and test this program.

1.5.3 AND condition

We would like to stop the SENSE only when it is close to the wall **and** only when it is on the black line.

To achieve that we need the **AND** condition.

1. Change the program to the following one:



```

STOP = None
WAIT = None
Black = None

STOP = 150
WAIT = STOP - 20
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP and (sense_get_input( id=1, input="bottom" )) < Black:
        sense_stop( id=1 )
        while not (sense_get_input( id=1, input="front" )) < WAIT:
            pass
        if (sense_get_input( id=1, input="bottom" )) > Black:
            sense_drive( id=1, command="left_turn", speed="fast" )
        if (sense_get_input( id=1, input="bottom" )) < Black:
            sense_drive( id=1, command="right_turn", speed="fast" )

```

2. Analyze the program.
3. Save the program under the name **CART8**.
4. Put the SENSE near the black line.
5. Run and check the SENSE movement.
6. Put your hand in front of the SENSE, while it moves.

Check the SENSE behavior.

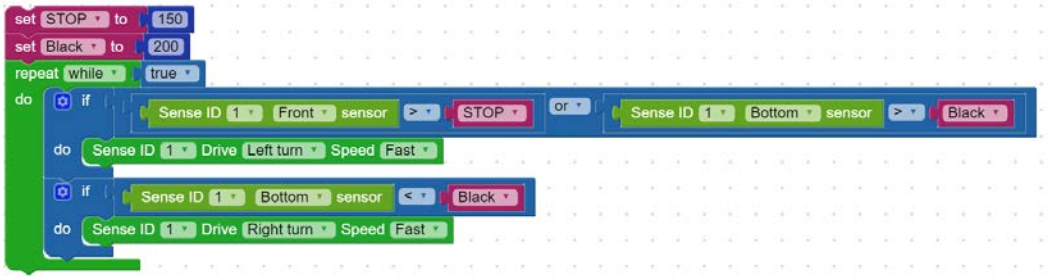
1.5.4 OR condition

Instead of stopping near the wall, we would like that the SENSE will turn around and continue on the black line to the other direction.

The SENSE will turn to the left when it is on a white surface or when it is close to the wall or both.

To achieve that we need the **OR** condition.

1. Change the program to the following one:



The block diagram shows a Scratch-style script. It starts with two 'set' blocks: 'set STOP to 150' and 'set Black to 200'. This is followed by a 'repeat while' loop with 'true' as the condition. Inside the loop, there is an 'if' block with an 'or' condition. The first part of the 'or' condition is 'Sense ID 1 Front sensor > STOP', and the second part is 'Sense ID 1 Bottom sensor > Black'. If this condition is true, the 'do' block contains 'Sense ID 1 Drive Left turn Speed Fast'. After this, there is another 'if' block with the condition 'Sense ID 1 Bottom sensor < Black'. If true, its 'do' block is 'Sense ID 1 Drive Right turn Speed Fast'.

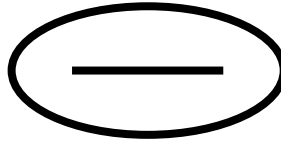
```

STOP = 150
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP or (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
  
```

2. Analyze the program and the procedures.
3. Save the program under the name **CART9**.
4. Put the SENSE near the black line.
5. Put a block on the black line.
6. Run and check the SENSE movement.

1.5.5 Challenge exercise – Along two lines

Task 1: Prepare two black lines as follows:



Put an obstacle on the inner line and let the SENSE move along this line.

When it meets the obstacle, it moves to the outer line and goes along it.

1.5.6 Movement along a wall

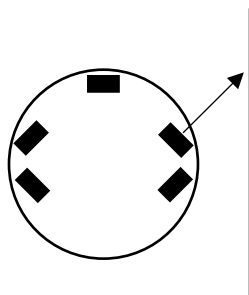
To move the SENSE along a wall, we use the same algorithm of moving the SENSE along a black line. We use the slow turn procedures.

In the main program, we use the **If** instruction to make the movement according to the following algorithm:

- Turn left when the SENSE is too close to the wall.
- Turn right when the SENSE is far from the wall.

To go along a wall on the right, we use the front side range sensor.

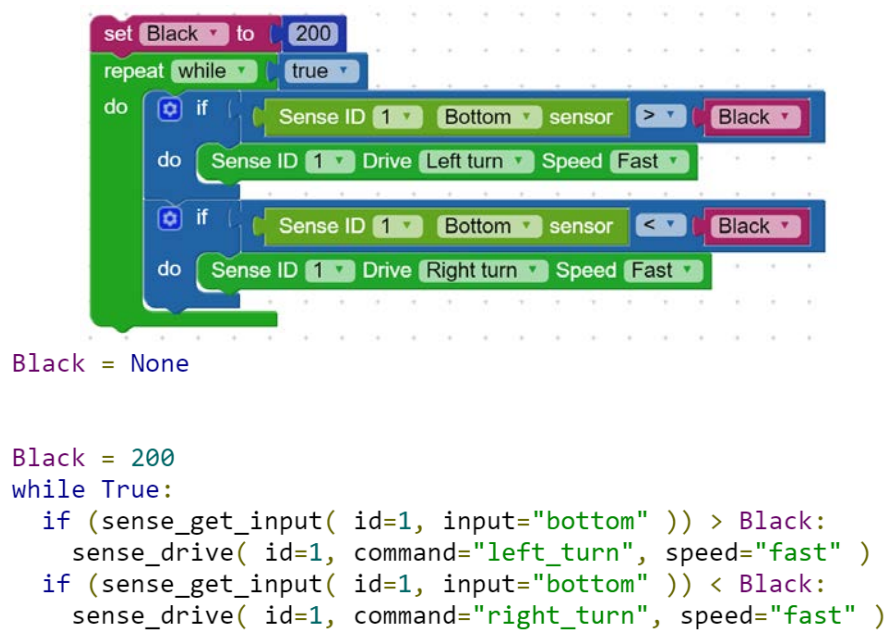
The side range sensors are installed in 45° to the SENSE base.



When the SENSE turns to the right, the measured distance is smaller than when it turns to the left.

Think what will happen if the range sensor is parallel to the wall.

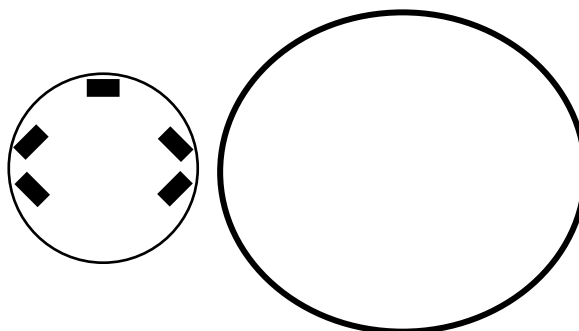
1. Take a round box (cylinder) to be used as the first exercise wall.
2. **CART6** is the basic program for movement along a black line.



3. Change the main procedure of **CART6** to use the **Right Front sensor** instead of the **Bottom sensor**.

Change the variable name from **Black** to **Wall** and its value for distance of 4cm from the wall.

4. Save the program under the name **WALL1**.
5. Put the SENSE near the round box and run the program.



Check that the SENSE moves around the box.

1.5.7 Challenge exercises – Along walls

Task 1: Improve the **WALL1** program so the SENSE goes forward when it does not sense a wall on its right side.

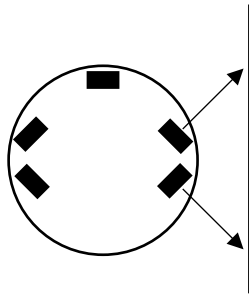
The SENSE stops when it meets a wall, turns to the left and starts moving along this wall.

Save this program under the name **WALL2**.

Task 2: Improve the **WALL2** program so the SENSE goes around a square box.

Put some obstacles in the way and improve the program so it will bypass them.

Task 3: Improve the **WALL2** program using two range sensors.



Challenge 1.6 – Counting

Draw block lines on a white paper.

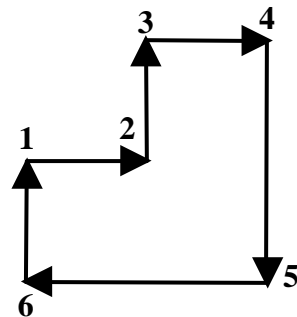


Create a program that moves the robot through the block lines and make it stop on the fourth line.

Use variables to count the lines.

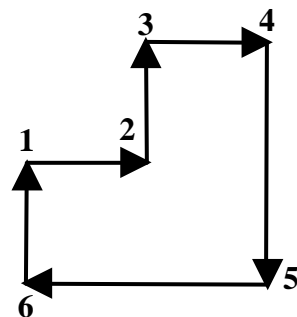
Challenge 1.7 – Automatic movement

Create a program that moves the robot according to the following figure:



Challenge 1.8 – Loops

Use loop commands to make the robot do the following cycles 3 times.



Challenge 1.9 – Loops and procedures

Convert each turn and forward movements into a procedure so the main program will have only the loop and run procedure instructions.

The program should do the same as the program in challenge 1.8.

Challenge 1.10 – "Don't touch me" robot

Create a "Do not touch me" program.

The robot should move away when you bring your hand close to one of its range sensors.

Challenge 1.11 – Robots in a convoy

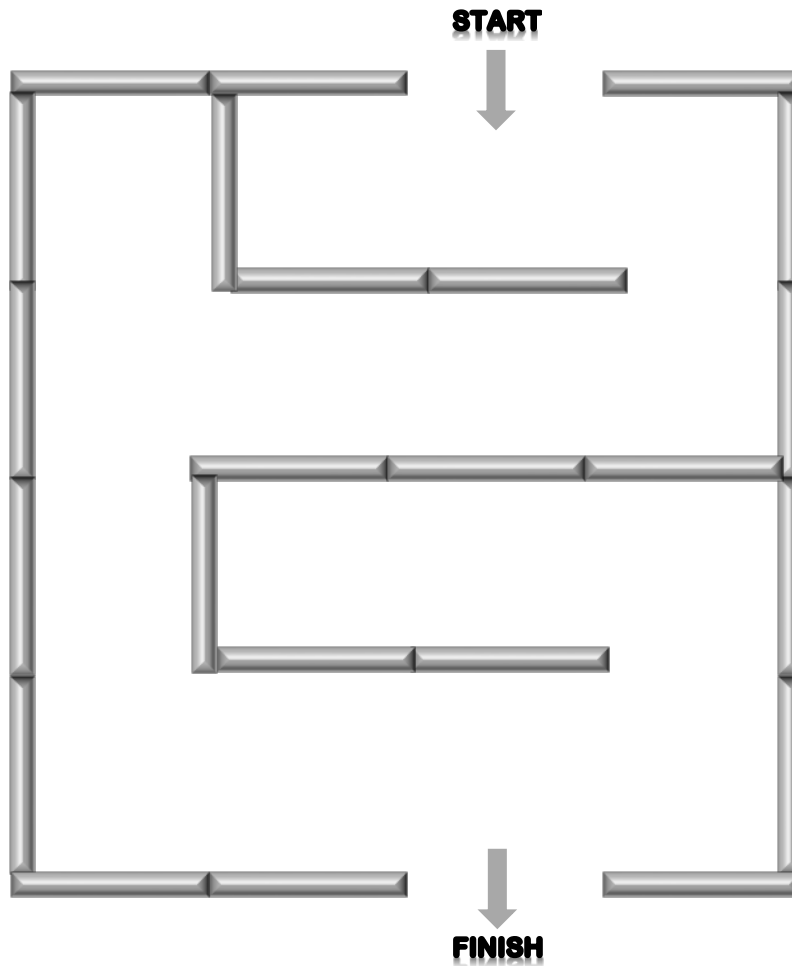
Put two robots on a black line.

The first robot should move along the black line and stop every 10 seconds.

The second robot should move along the black line and stop when it is close to the first robot.

Challenge 1.12 – Movement in a labyrinth

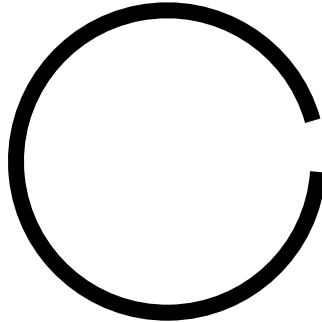
Build a labyrinth as follows:



Create a program that moves the robot from the **START** point to the **FINISH** point without touching the walls.

Challenge 1.13 – Exiting a circle

Draw a wide black line as follows:



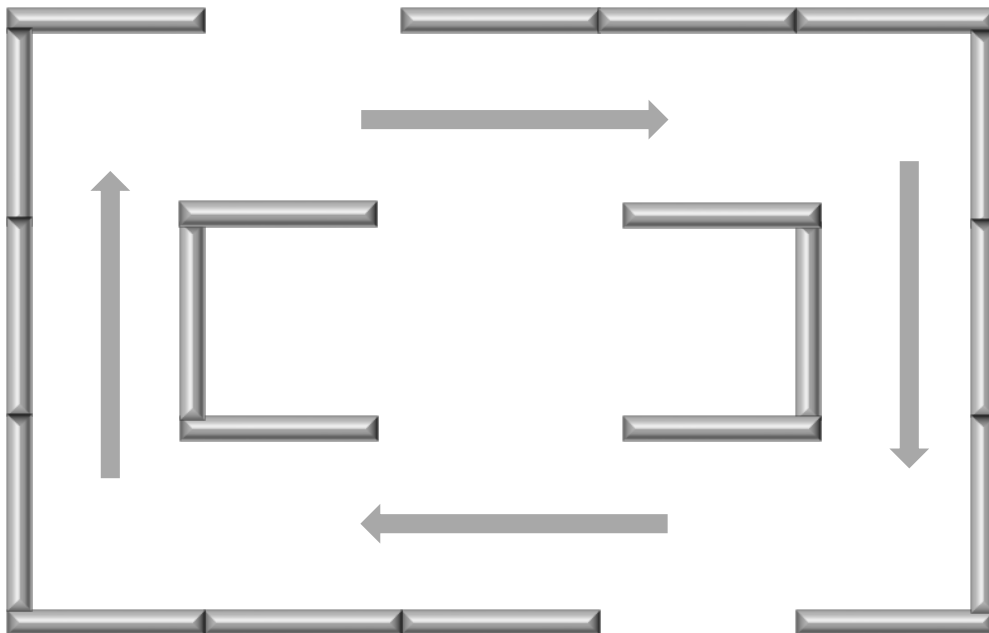
Put the robot inside the circle.

The robot should not cross the black line or move along the black line.

Create a program that makes the robot exit the circle according to the above rules.

Challenge 1.14 – Moving along corridors

Build the following corridor model:



The corridor's and the doors' widths are about 20cm.

The robot should move in the corridor without getting out through the doors.

Create a program that answers this challenge.

Experiment 1.15 – Editing Python program

Objectives:

- Editing Python programs

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module

Discussion:

1.15.1 Python built-in library

The Python is very simple and very powerful high-level programming language that works on any computer platform. It is an open-source coding program.

The Python interpreter has a number of functions built into it that are always available. They are listed here in alphabetical order.

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

The built-in functions are just part of all other options available in Python.

Python is simple and intuitive coding program. The following is CART5 program that uses the SNSE Python instructions.

```
Black = None

def Turn_left_on_white():
    global Black
    sense_drive( id=1, command="left_turn", speed="fast" )
    while (sense_get_input( id=1, input="bottom" )) > Black:
        pass

def Turn_right_on_black():
    global Black
    sense_drive( id=1, command="right_turn", speed="fast" )
    while (sense_get_input( id=1, input="bottom" )) < Black:
        pass


Black = 300
while True:
    Turn_left_on_white()
    Turn_right_on_black()
```

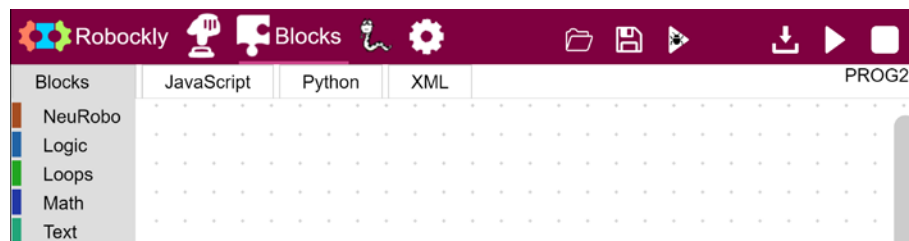
One idiom that trips up many new Python developers is indentation. Python uses indentation (4 spaces) to logically organize code into sections called code blocks.


A code block starts with an indent and ends with a un-indent. Incorrect indentation will generate an error in Python preventing your code from executing.

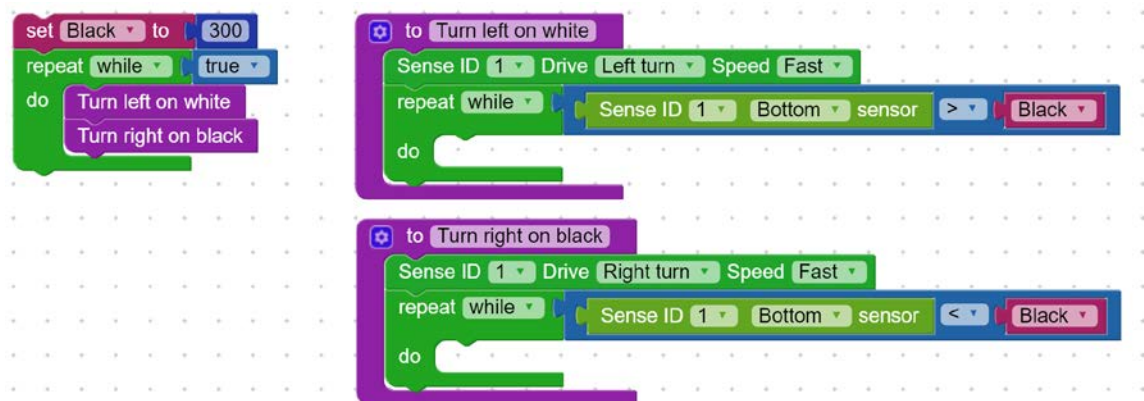
You should also use spaces instead of tabs when indenting.

Procedure:

1. Plug the WIFI-203 and BAT-202 on the Sense robot.
2. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
3. Select the neulog Wi-Fi network in your computer (i.e., neulog3233).
4. Browse to **wifi203.com**.
5. Move to **Block**  mode.



6. Click on the **Open**  button and open the program **CART5**.
7. Check that you have the following program:



If not, build this program and save it under the name CART5.



8. Observe the Python program of CART5 and compare the lines with the **Robockly** program

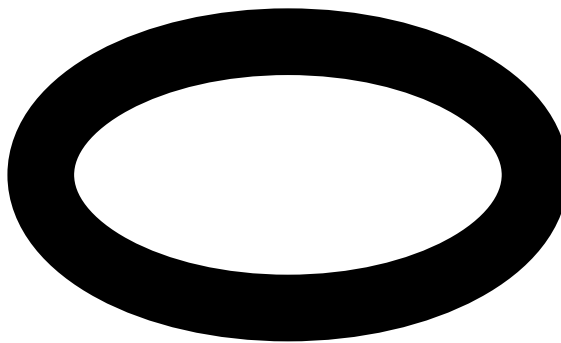
```
Black = None

def Turn_left_on_white():
    global Black
    sense_drive( id=1, command="left_turn", speed="fast" )
    while (sense_get_input( id=1, input="bottom" )) > Black:
        pass

def Turn_right_on_black():
    global Black
    sense_drive( id=1, command="right_turn", speed="fast" )
    while (sense_get_input( id=1, input="bottom" )) < Black:
        pass


Black = 300
while True:
    Turn_left_on_white()
    Turn_right_on_black()
```

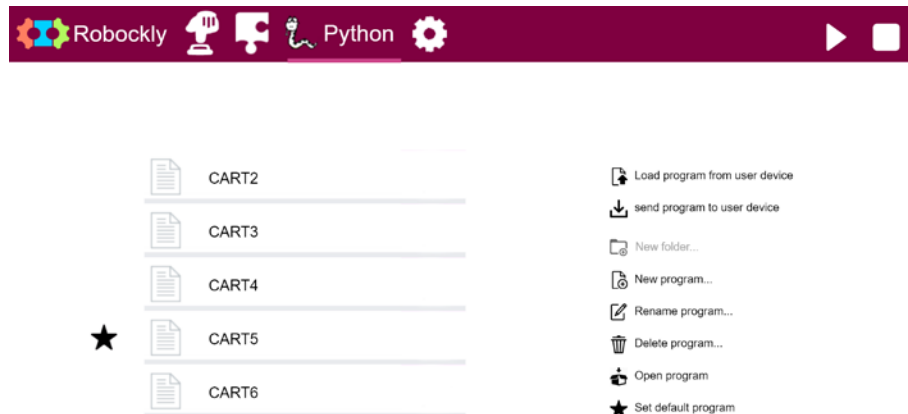
9. Return to **Blocks** screen.
10. Place the SENSE on the black line circle.
11. Download  the program and run  it.



The SENSE should go along the black line.

12. Stop  the program.

13. Click on the **Python**  button and you will see the following screen:



The list of the programs that were downloaded to WIFI-203 flash memory appears on the left.

The file with the '**Star**' is the selected one – the default program. On the right are the operations that can be operated on these files.

Clicking on a file name highlights it but does not make it the default.

14. To change the default program, highlight another file name and click on the '**Set default program**' button on the right.
15. After editing a Python program, it cannot be converted back to Robockly. This is why it is better to change its name before editing it.

Change the name of the program by renaming it to CART52.

16. Make the **CART52** the default program and click on for highlighting it.



17. Click on the 'Open program' button and you will get the following screen:

 The left sidebar shows a file explorer with 'Turn_left_on_white' and 'Turn_right_on_black' files. The right sidebar has a close button (X)."/>

The program starts with the Python version.

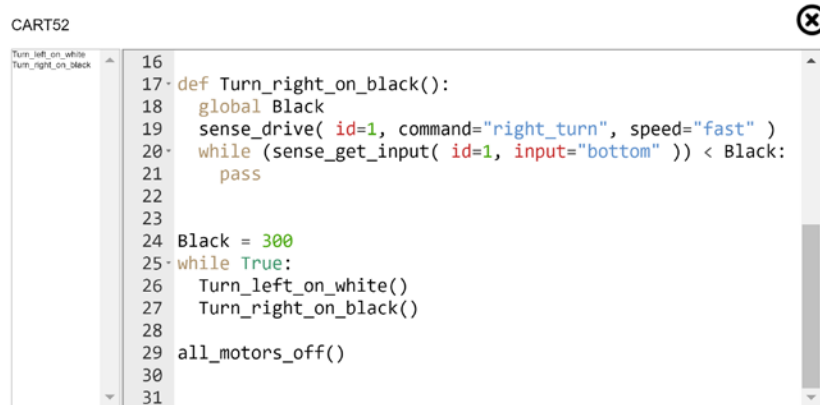
Lines 6 and 7 tell that the program imports functions and variables from the **robockly_lib** file the **init_robo()** library.

The program starts with declaration of the variable **Black** with the default value **none**. This variable is **global** as declared in the functions, which means that it can be called any place in the program.

After that are the definitions of the functions.

The main program is at the end.

18. Scroll down and view the program.



```

16 def Turn_right_on_black():
17     global Black
18     sense_drive( id=1, command="right_turn", speed="fast" )
19     while (sense_get_input( id=1, input="bottom" )) < Black:
20         pass
21
22
23
24 Black = 300
25 while True:
26     Turn_left_on_white()
27     Turn_right_on_black()
28
29 all_motors_off()
30
31

```

19. The instruction '**all_motors_off()**' is inserted by **Robockly** automatically.

20. On the left, there are the names of the program functions.

Clicking on one of them will highlight the first line of this function.

This help to navigate in the program.

Try that.

21. We shall add the '**Check_stop**' function with **STOP** variable.

Scroll up and add the STOP variable.



```

1 #!/usr/bin/python2.7
2 #
3 # SES Robockly program
4 #
5
6 from robockly_lib import *
7 init_robo()
8
9 STOP = None
10 Black = None
11
12 def Turn_left_on_white():
13     global Black
14     sense_drive( id=1, command="left_turn", speed="fast" )
15     while (sense_get_input( id=1, input="bottom" )) > Black:
16         pass



```

22. Add the following **def Check_stop** function as in the screen below.

```

24 def Check_stop():
25     global STOP
26     if (sense_get_input( id=1, input="front" )) > STOP:
27         sense_stop( id=1 )
28     while (sense_get_input( id=1, input="front" )) > STOP:
29         pass
30
31
32 STOP = 300
33 Black = 200
34 while True:
35     Check_stop()
36     Turn_left_on_white()
37     Turn_right_on_black()
38
39 all_motors_off()

```

23. Note that the **Check_stop** name appears on the left list.
24. Add the **STOP = 300**.
25. Add the **Check_stop()** call in the main program.
26. Save the program by clicking on the **Download**  button.
27. You can run and stop the program while you are on this screen using the buttons .
28. Put the SENSE near the black line.
29. Run and check the SENSE movement.
30. Put your hand in front of the SENSE, while it moves.

Change the values of the variables until the SENSE works well.

31. You can return to the file management screen and run it from there too when CART52 is the default program.

The last downloaded program automatically becomes the default one.

Note:

CART52 cannot be transferred back to **Robockly**.

Chapter 2 – Brain Units

2.1 Brain units

Some of the input units can have their own "brain". The NeuLog sensors are such brain units. They send to the control unit, upon request, processed data such as: temperature (°C or °F), light intensity in Lux, distance in meters, etc.

The output units can also be brain units. For example, units that control the motor speed and direction, lamp intensity, servo motor angle, etc.

These brain units are connected in a chain to the main control unit, which communicates with them through messages.

Every brain unit has an ID number. Every message from the control unit starts with ID number. Only the brain unit with this ID number interprets the message and executes it.

This system construction is the way modern systems are built, and has important advantages:

1. It creates a system with much less wires. The wires go from one module to another and not from all modules to the control unit.
2. This kind of system can easily be changed and expanded, and does not depend on the control units number of inputs and outputs.

The experiments in this chapter use the following brain units:

- Neulog sound sensor (NUL-212)
- Neulog motion sensor (NUL-213)
- Neulog light sensor (NUL-204)
- Neulog magnetic sensor (NUL-214)
- Brain tracking unit (SNS-101)
- Brain gripper arm (SNS-167)

If you do not have them, you can read about them and move to chapter 3.

Chapter 3 experiments are with The SENSE robot and battery module.

2.2 NeuLog sensors as brain units



NeuLog sensors (Neuron Logger Sensors) are also brain units. Each sensor includes a tiny computer, which samples, processes and stores the sampled data. Each probe connected to the sensor is pre-calibrated in the factory and no further calibration is required.

The data provided by the sensor is processed digital data. The sensor includes different measurement ranges. Changing the measuring range or type of processing is done simply on the computer screen with NeuLog software.

The sensors are plugged to each other with almost no limitation on the composition and number of sensors in the chain.

NeuLog has over 50 different sensors. Some sensors perform as two to three sensors.

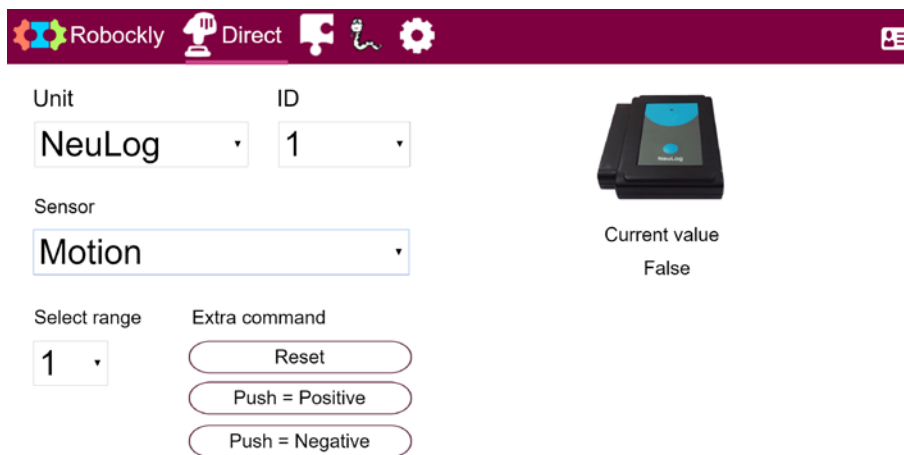
The SENSE has three sockets for NeuLog sensors.

2.3 Changing Brain unit ID

As said before, every brain unit has ID number. The ID number enables us to up to nine brain units of the same kind. We just have to take care that each one of them will have different ID number.

In **Direct** mode screen, we have a special icon for changing the ID number of the unit.

The following screen is the **Direct** screen of the motion sensor.



Robockly Direct

Unit: NeuLog ID: 1


Sensor: Motion

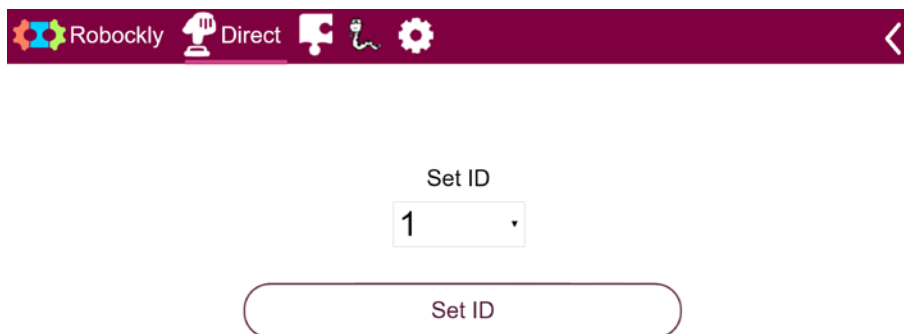
Select range: 1

Extra command:

- Reset
- Push = Positive
- Push = Negative

Current value: False

Clicking on the  icon on the right, will show the following screen:



Robockly Direct

Set ID

1

Set ID

In order to change the ID number of the unit, we have to connect only one unit to the PC, to set the required ID number in the Set ID field and to click on the **Set ID** button.

Experiment 2.1 – Sound Sensor

Objectives:

- The sound sensor
- Operating the SENSE by sound

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module
- NUL-212 NeuLog sound sensor

Discussion:

The sound sensor uses an internal microphone and special amplifier. Sound waves enter through the hole in the top of the sensor's plastic body so you should point that directly towards the sound source for best readings.

The sound sensor has two modes (ranges) of operation:


1. **Arbitrary analog units (Arb)** – An arbitrary unit indicates a number according to signal shape. At this mode, the sound is sampled and reconstructed as a signal.
2. **Decibel (dB)** – A unit of measure to show the intensity (loudness of sound). Please note that this is a logarithmic unit.

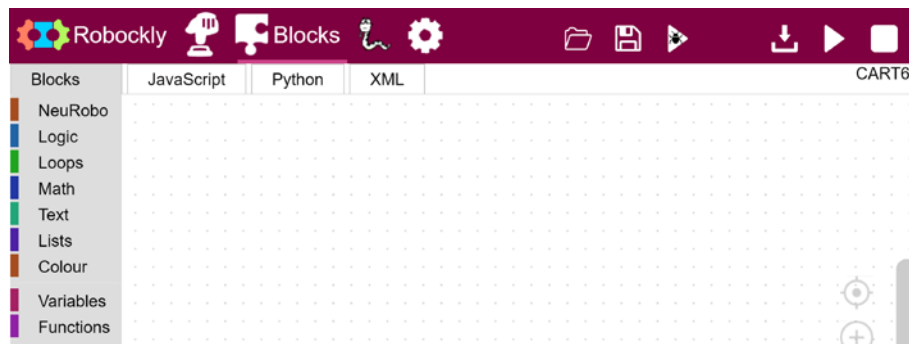
At this mode, the wave is sampled and the average intensity (calculated by the sensor controller) is converted into dB value. 40 dB represents silence.

In this experiment, we shall use it at dB mode and we assume its ID is 1 as the default ID.

Selecting the range should be done by the NeuLog software.

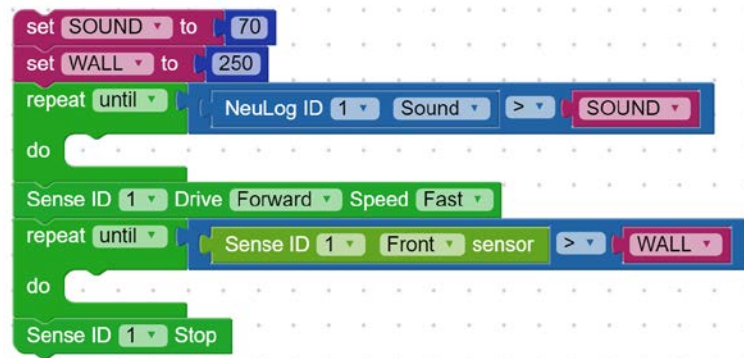
Procedure:

1. Plug the NeuLog sound sensor into one of the SENSE sockets.
2. Plug the WIFI-203 and BAT-202 on the Sense robot.
3. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
4. Select the neulog Wi-Fi network in your computer (i.e., neulog3233).
5. Browse to **wifi203.com**.
6. Move to **Block**  mode.



Create a program that waits for a sound level of 70 dB, moves the SENSE to a wall, and then stops.

7. Build the following program and its main procedure:



```
SOUND = None
WALL = None
```

```
SOUND = 70
WALL = 250
while not (neulog_get_input( type="sound", id=1 )) > SOUND:
    pass
sense_drive( id=1, command="forward", speed="fast" )
while not (sense_get_input( id=1, input="front" )) > WALL:
    pass
sense_stop( id=1 )
```

8. Observe the program and make sure that you understand all of its instructions.
9. Place the SENSE in front of a wall and run the program.

The SENSE should not move.

10. Clap your hand or make high sound.

The SENSE should move towards the wall and stop.

2.1.1 Challenge exercise – Wait for sound

Task 1: Improve the program so:

- The SENSE will wait for a sound above 70 dB, then moves forward until it meets a wall and then stops.
- It will wait again for the sound, moves backward until it reaches a black line and then stops.
- Returns to the beginning.

Experiment 2.2 – Motion Sensor

Objectives:

- The motion sensor as distance sensor
- Moving the robot according to the motion sensor

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module
- NUL-213 NeuLog motion sensor

Discussion:

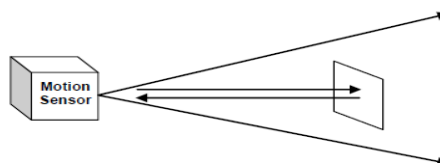
The motion sensor uses an ultrasonic transducer to both transmit an ultrasonic wave, and to measure its echo return. Objects in the range of 0.15 to 10 meters can accurately be measured to give distance, velocity, and acceleration readings using this method.

The motion sensor can collect data using the following measuring units:

- **Meters (m)** – The SI (International System of Units) distance unit
- **Meters/second (m/s)** – The SI velocity unit, which measures the distance traveled over time.
- **Meters/second² (m/s²)** – The SI acceleration unit, which measures the change in velocity over time.

The motion sensor has two working ranges – one between 0.2 and 10.0 meters and one between 0.15 to 2 meters.

Ultrasonic waves are emitted from the sensor and spread out in a cone pattern at about 15° around the point of reference.



The ultrasonic transducer is a device that can convert pulse train to transmitted ultrasonic pulses. These pulses can sense and convert back to electronic pulse train by another similar ultrasonic transducer, or by itself.

The ultrasonic transducer is based on ceramic crystal, which is cut in a certain way and is placed between two metal plates. The crystal is characterized by the piezoelectric effect. Electrical field changes between the plates create mechanical vibrations in the crystal.

The crystal has a resonance frequency. The mechanical vibrations and electrical reactions depend on this resonance frequency.

Supplying pulses to the crystal of the ultrasonic transducer (in a rate according to its frequency) causes it to vibrate and to transmit these pulses as an acoustic sound. This sound cannot be heard because it is above the hearing frequency range (usually it is at 40KHz).

The acoustic sound can be converted back to electronic pulses by another ultrasonic transducer or by the transmitter when it stops transmitting. The acoustic pulses vibrate this transducer and these vibrations are turned into voltage pulses.

The speed of the ultrasonic wave is about 300 m/s because it is a sound wave.

For distance measurement, a burst of the transducer frequency wave is sent and the system measures the time between the sending and the receiving.

$$S = 300 \cdot t$$


Velocity is calculated by the difference between two successive distances divided by the time between the samples (according to the sampling rate).

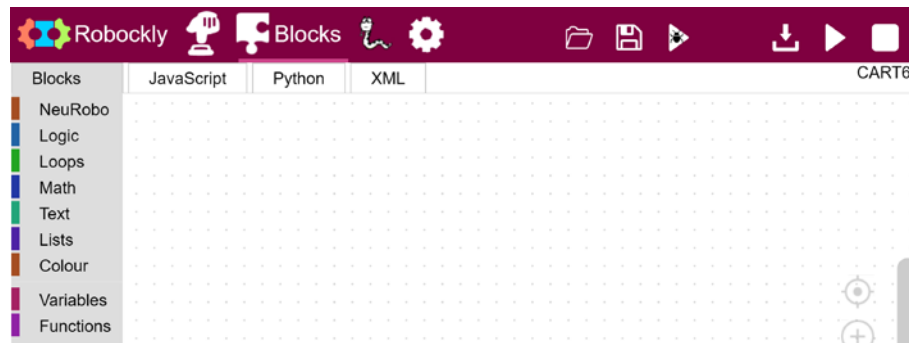
Acceleration is calculated the difference between two successive velocities divided by the time between the samples (according to the sampling rate).

The motion sensor uses a very sophisticated method that enables it to measure long distance range with a low power of pulses.

In this experiment, we shall use it at distance range and we assume its ID is 1 as the default ID. Selecting the range should be done with the NeuLog software.

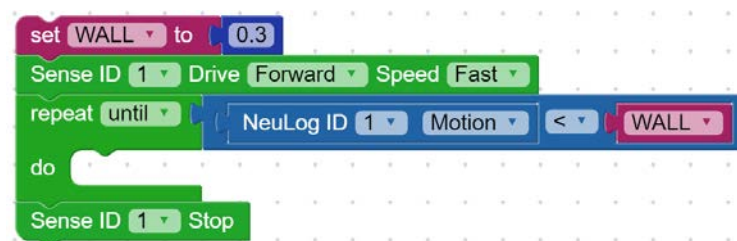
Procedure:

1. Plug the NeuLog motion sensor into one of the SENSE sockets with its transducer directly to the front of the SENSE.
2. Plug the WIFI-203 and BAT-202 on the Sense robot.
3. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
4. Select the neuLog Wi-Fi network in your computer (i.e., neuLog3233).
5. Browse to **wifi203.com**.
6. Move to **Block**  mode.



Create a program that moves the SENSE forward to a wall and stops 30 cm away from it.

7. Build the following program and its main procedure:



WALL = None

```
WALL = 0.3
sense_drive( id=1, command="forward", speed="fast" )
while not (neulog_get_input( type="motion", id=1 )) < WALL:
    pass
sense_stop( id=1 )
```


8. Observe the program and make sure that you understand all of its instructions.
9. Place the SENSE in front of a wall and run the program.

The SENSE should move to the wall and stop 30 cm away from it.

3.2.1 Challenge exercise – Moving in a distance range

Description: Going forward towards a wall, stop 30cm before the wall, then go backward and stop at 50cm from the wall and return.

Task 1: Improve the program so the SENSE will:

- move towards the wall,
- stop 30 cm in front of it,
- wait for 2 seconds,
- go backwards until a distance of 60 cm,
- stop for 2 second,
- return to the beginning.

Experiment 2.3 – Brain Tracking Unit

Objectives:

- The brain tracking unit
- Moving to an IR (infrared) transmitter
- Following an IR transmitter

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module
- SNS-101 Brain tracking unit
- SNS-160 IR transmitter

Discussion:

2.3.1 IR Transmitter

The infra-red transmitter (SNS-160) can be plugged into any of the SENSE sockets or in the backup battery socket to be followed by the brain tracking unit.

Infrared light is transmitted from a heat source. We cannot see the IR light. The frequency of this light is a little below the red light and this is why we call it infra (before) red.

The surrounding light does not affect this light much.



2.3.2 Brain tracking unit

The brain unit, in a rigid plastic case, can be plugged into one of the SENSE sockets.

The brain tracking unit has two IR (infra-red) sensors that enables it to track the IR transmitter.

The two IR sensors are at the same line with an opaque partition between them.



When IR light falls on both of them, it means that the SENSE is in front of the IR light source.


When the SENSE is at angle to the light source, the IR light will fall only on one of the IR sensors.

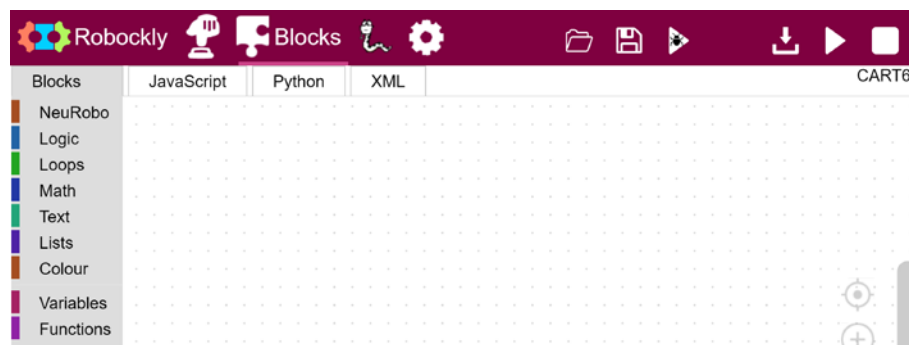
The third IR sensor measures the environment IR light. The brain unit controller uses this measurement to eliminate the environment light.

The brain unit output is a binary number that describes the detection status of an IR transmitter. This number is converted to detection results as the following:

- None (00) – No IR transmitter light
- Right (01) – IR transmitter light on the right
- Left (10) – IR transmitter light on the left
- Front (11) – IR transmitter light at front

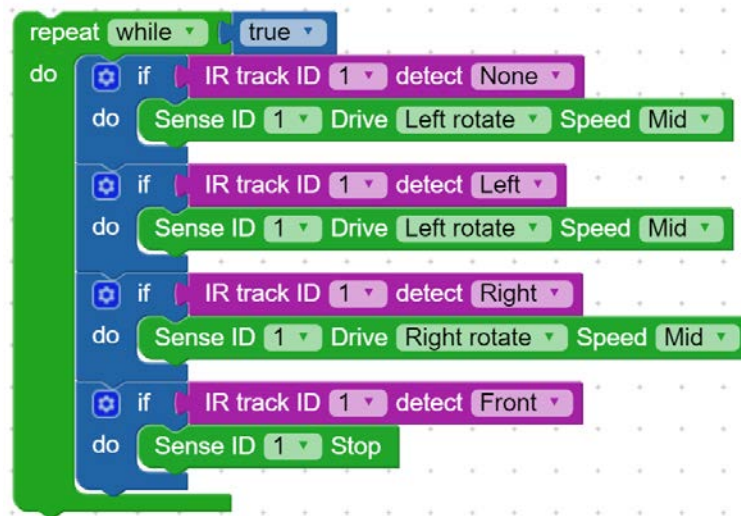
Procedure:

1. Plug the brain tracking unit into the front socket of the SENSE.
2. Plug the IR transmitter into a backup battery. Put the battery backup on a 3cm high surface.
3. Plug the WIFI-203 and BAT-202 on the Sense robot.
4. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
5. Select the neulog Wi-Fi network in your computer (i.e., neulog3233).
6. Browse to **wifi203.com**.
7. Move to **Block**  mode.



Create a program that rotates the SENSE to the left until it "sees" the IR transmitter and then tracks it without moving (just rotates).

8. Build the following program and its main procedure:



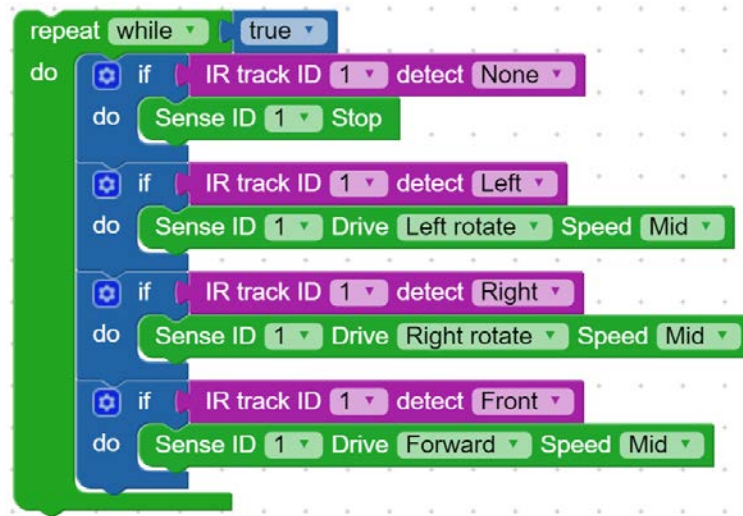
```
while True:
    if ir_track_get_input( id=1, pos="none" ):
        sense_drive( id=1, command="left_rotate", speed="mid" )
    if ir_track_get_input( id=1, pos="left" ):
        sense_drive( id=1, command="left_rotate", speed="mid" )
    if ir_track_get_input( id=1, pos="right" ):
        sense_drive( id=1, command="right_rotate", speed="mid" )
    if ir_track_get_input( id=1, pos="both" ):
        sense_stop( id=1 )
```

9. Observe the program and make sure that you understand all of its instructions.
10. Download the program.
11. Place the SENSE on the floor and run the program.

The SENSE should turn to the left until it 'sees' the infrared beam.

12. Plug the IR transmitter into the battery module, move it slowly and check that the SENSE tracks it.

13. Change the program to the following:



```
while True:
    if ir_track_get_input( id=1, pos="none" ):
        sense_stop( id=1 )
    if ir_track_get_input( id=1, pos="left" ):
        sense_drive( id=1, command="left_rotate", speed="mid" )
    if ir_track_get_input( id=1, pos="right" ):
        sense_drive( id=1, command="right_rotate", speed="mid" )
    if ir_track_get_input( id=1, pos="both" ):
        sense_drive( id=1, command="forward", speed="mid" )
```

14. This program should move the SENSE towards the IR transmitter. The SENSE waits when it does not detect the IR light. Check that.

2.3.3 Challenge exercise – Tracking a robot with IR transmitter

Task 1: Improve the above program and procedures so the SENSE will stop in front of the IR transmitter.

Put the IR transmitter on a box or another SENSE that can be detected by the front sensor.

Experiment 2.4 – Brain Gripper Arm

Objectives:

- The brain gripper arm.
- Moving an object from one place to another
- Drawing pictures with the brain gripper arm

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module
- SNS-167 Brain gripper arm
- A wooden rod
- A marker

Discussion:

2.4.1 Brain gripper arm

The brain gripper arm has two servo motors.

One servo motor moves the gripper up and down.

The second servo motor opens and closes the gripper.

A servo motor is a motor with feedback. The feedback can be voltage according to the motor speed or the shaft angle, electrical pulses according to the motor shaft rotation and direction, and more.

Each servo motor of the gripper arm has transmission gear and potentiometer. The potentiometer consists of a variable resistor that create variable voltage according to the servomotor shaft angle.

The brain controller of the gripper arm gets the required angle of the shaft. It turns the motor CW (Clock Wise) or CCW (Counter Clock Wise) until the potentiometer voltage suits this angle.

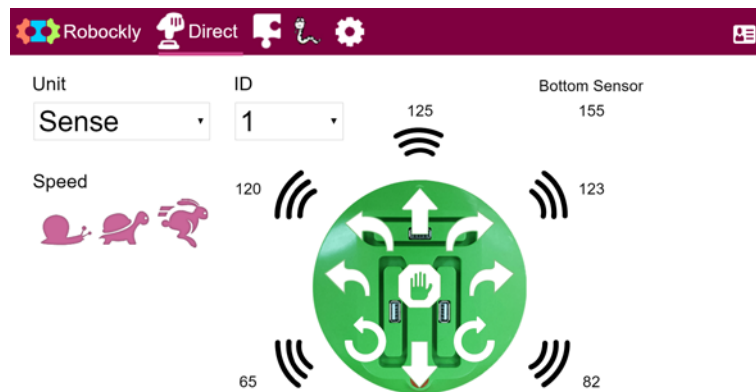
It checks the shaft angle all the time. If it changes mechanically, the controller will turn the motor ON to return the shaft to the right position.



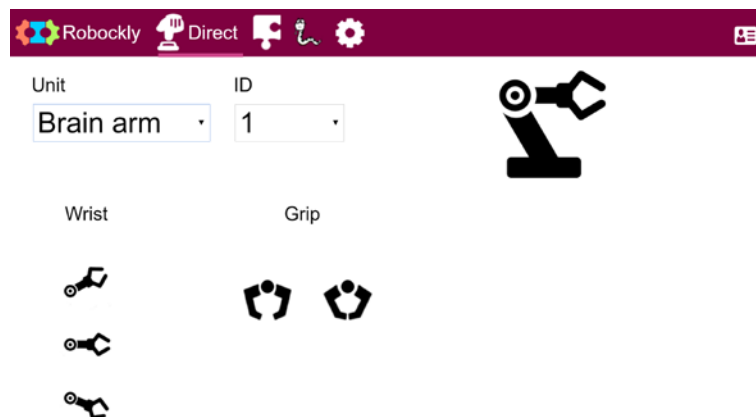
Procedure:

1. Plug the brain gripper arm into the front socket of the SENSE.
2. Plug the WIFI-203 and BAT-202 on the Sense robot.
3. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
4. Select the neulog Wi-Fi network in your computer (i.e., neulog3233).
5. Browse to **wifi203.com**.

The Direct screen appears:



6. Change the selected Unit to **Brain arm**.



7. Play with the buttons on the screen.

Move the gripper to up, mid and down positions.

Open and close the gripper.

8. Move the gripper to mid position.

Open the gripper.

Put the wooden rod between the gripper fingers.

Close the gripper.

Raise the gripper.

The close function closes the gripper for two seconds, measures the angle of the gripper servo motor and holds it in this angle.

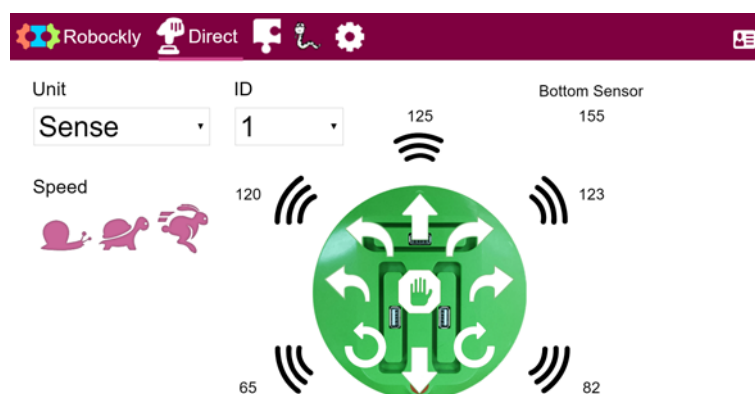
This is why the gripper can hold objects with different thickness.

9. Move the gripper to mid position.

Open the gripper.

Put the wooden rod between the gripper fingers.

10. Change the selected Unit to **Sense**.

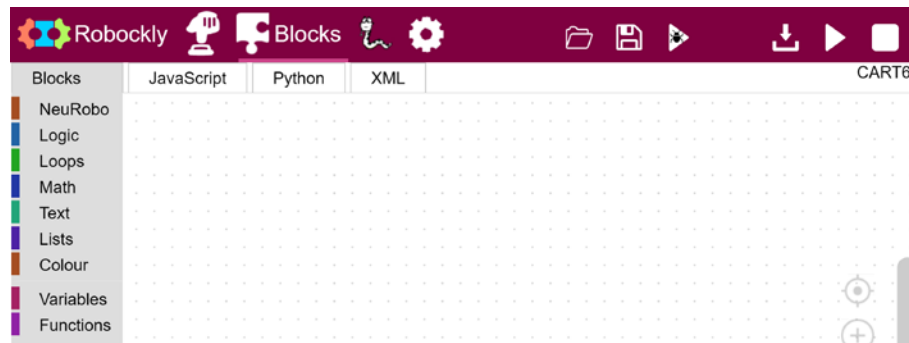


11. Record the value of the front sensor.

For the following example, we shall use the number 200.

12. Put the wooden rod 20 cm in front of the SENSE.

13. Move to **Block**  mode.



14. Create a program that does the following:

- (a) Opens the gripper.
- (b) Raises the arm.
- (c) Moves the Sense forward and stops when the wooden rod between the gripper fingers.
- (d) Lowers the arm to mid position.
- (e) Closes the gripper.
- (f) Raises the arm.
- (g) Moves forward for 2 seconds.
- (h) Lowers the arm to mid position.
- (i) Opens the gripper.
- (j) Moves backward for 2 seconds.

15. Build the following program and its main procedure:



ROD = None

```

ROD = 0.3
brain_arm_set_position( id=1, wrist="none", grip="open" )
brain_arm_set_position( id=1, wrist="up", grip="open" )
sense_drive( id=1, command="forward", speed="slow" )
while not (sense_get_input( id=1, input="front" )) > ROD:
    pass
sense_stop( id=1 )
delay_sec(1)
brain_arm_set_position( id=1, wrist="mid", grip="open" )
delay_sec(1)
brain_arm_set_position( id=1, wrist="mid", grip="close" )
delay_sec(1)
brain_arm_set_position( id=1, wrist="up", grip="close" )
delay_sec(1)
sense_drive( id=1, command="forward", speed="slow" )
delay_sec(2)
sense_stop( id=1 )
delay_sec(1)
brain_arm_set_position( id=1, wrist="mid", grip="close" )
delay_sec(1)
brain_arm_set_position( id=1, wrist="mid", grip="open" )
delay_sec(1)
sense_drive( id=1, command="backward", speed="slow" )
delay_sec(2)
sense_stop( id=1 )

```

16. Observe the program and make sure that you understand all of its instructions.
17. Download the program.
18. Disconnect the SENSE from the computer.
19. Press the SENSE **Run/Stop** button.
20. Check that the SENSE does its mission.
21. Change the program to run in endless loop.
22. Download the program, run and check the SENSE behavior

2.4.2 Challenge exercises – The SENSE with gripper arm

- Task 1: Change the last program to use functions instead of chain of instructions. Put the delays in the functions.
- Task 2: Change the program to make the SENSE to rotate in about 90° with the raised wooden rod before moving forward with it.
- Task 3: Plug Sound sensor to the SENSE and make it wait for hand clapping before picking up the wooden rod.
- Task 4: Make the gripper hold a marker manually.

Place the SENSE on wide white paper attached to the ground or to the desk.

Build some drawing programs.

Experiment 2.5 – Robot and Science experiment

Objectives:

- The Neulog light sensor
- Running an experiment while moving
- Using the SENSE as a USB module with Neulog software

Equipment required:

- Computer
- SENSE autonomous
- WIFI-203 Wireless coding unit
- BAT-202 Battery module
- NUL-204 Neulog light sensor
- NUL-213 Neulog motion sensor
- A flashlight

Discussion:

2.5.1 The Neulog light sensor

The Light Sensor can be used for any science experiment where light intensity measurements are required, such as Chemistry, Physics, Biology, Environmental Science, etc.

This sensor can be used to take light measurements in low, medium and high light intensity environments, such as in classrooms and in open sunlight. The sensor can be used to measure both fast light changes like those produced by light bulbs connected to an AC supply, as well as the light intensity of a bulb or near steady levels outside on a sunny day.

The measurement unit for all three data collection ranges (low, medium, high) is the lux.

Lux (lx, or lux): The SI unit of light intensity.

The light sensor includes a photodiode, which reacts with photons to release free electrons (photoelectrons). The amount of light striking the sensor is directly proportional to the voltage generated by the photoelectrons released. The sensor measures the general voltage released and thus calculates the light intensity.

If the light readout is very low, try changing the sensor's mode to a higher sensitivity. This is done by selecting the “Module setup” button on the light sensor module box in the NeuLog application.

The Neulog light sensor is able to adjust to 3 different sensitivity settings for ambient light because of its ability to change the internal hardware amplifier gain through the application.

Changing from illumination mode into signal mode is done automatically by the firmware according to the sampling rate.

2.5.2 Light intensity vs distance

In this experiment, we shall move the light sensor against a flashlight and a wall. We shall measure the light intensity and the distance with a distance sensor.

The **Robockly** has instructions to run an experiment and to stop it.

The Neulog sensors are logger sensors. They sample and save the measurements in their flash memory.

2.5.3 The SENSE as USB module

The SENSE can act as a USB module for the NeuLog application software.

With this software we can view graphically the experiment results after running the Robockly program.

We can, of course, save and use all Neulog software functions.


Procedure:

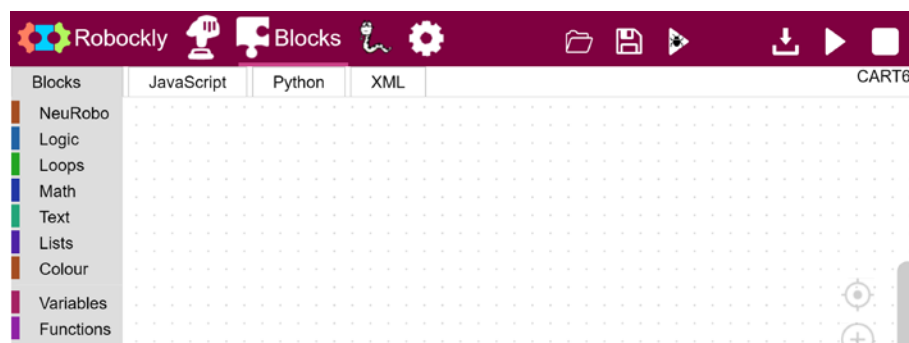
1. Plug the light sensor NUL-204 and the motion sensor units into the socket of the SENSE, as in the following picture.



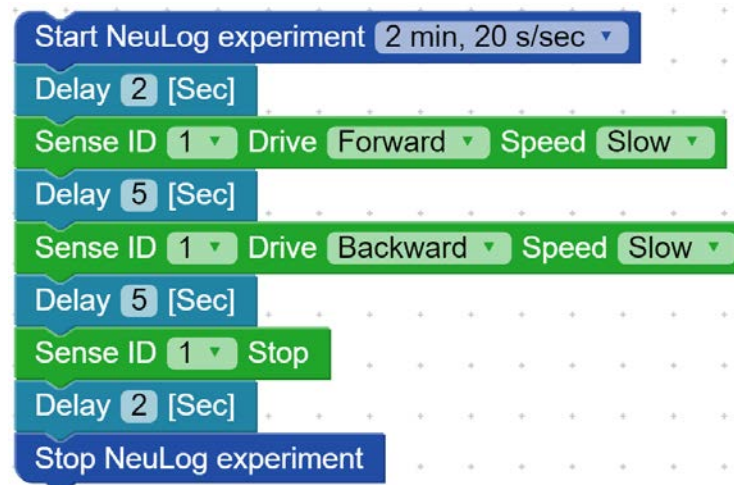
2. Plug the battery to the second socket.
3. Put a flashlight near the wall as in the following picture.



4. Plug the WIFI-203 and BAT-202 on the Sense robot.
5. Wait until the WIFI-203 LED stops blinking and turns to blue constant.
6. Select the neulog Wi-Fi network in your computer (i.e. neulog3233).
7. Browse to **wifi203.com**.
8. Move to **Block**  mode.



9. Build the following program:



```

neulog_run_experiment(7)
delay_sec(2)
sense_drive( id=1, command="forward", speed="slow" )
delay_sec(5)
sense_drive( id=1, command="backward", speed="slow" )
delay_sec(5)
sense_stop( id=1 )
delay_sec(2)
neulog_stop_experiment()

```

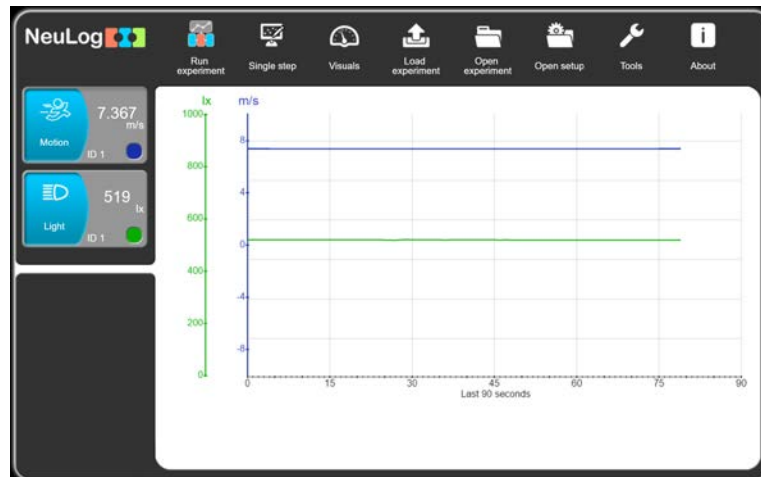
10. Observe the program and make sure that you understand all of its instructions.
11. Download the program.
12. Disconnect the SENSE from the PC and place it on the floor against the flashlight as in the following picture.



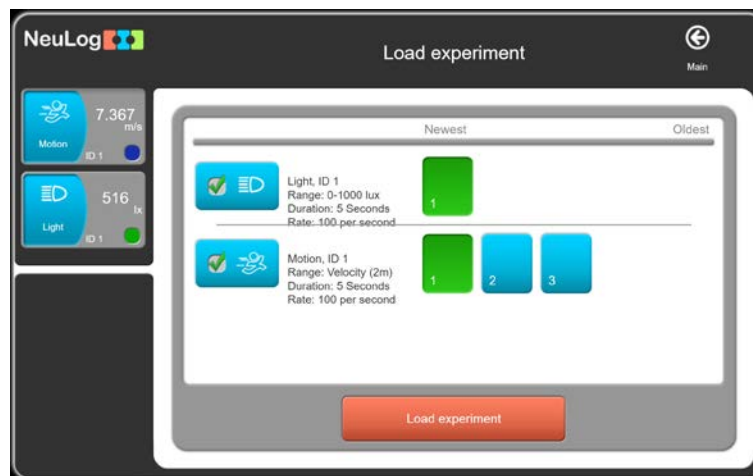
13. Run the program.

The SENSE should wait for 2 seconds, go forward for 5 seconds, then go backward for 5 seconds and stop.

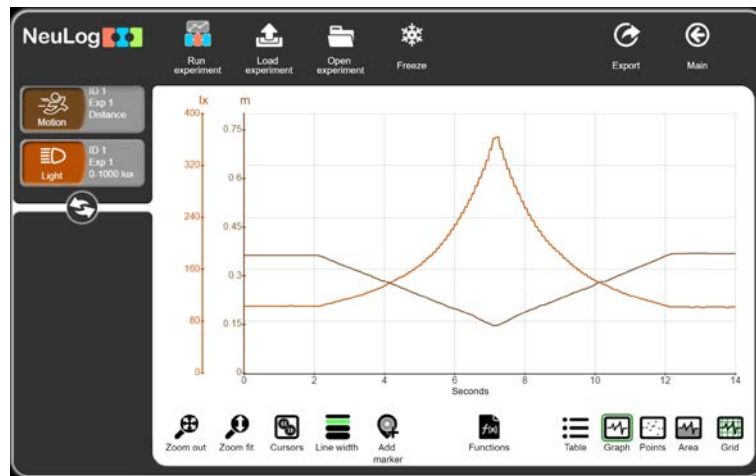
14. Connect the SENSE to the PC.
15. Exit the Robockly program.
16. Run the NeuLog software and wait for the following screen.



17. Click on the **Load experiment** button for the following screen.



18. Click on the **Load experiment** orange button for loading the experiment results from the sensors.



19. You can see on the graph the two seconds delays at the beginning and at the end, when the distance and the light are constant.
20. When the SENSE moves forward, the distance drops linearly and the light intensity increases in parabola shape.
21. When the SENSE moves backward, the distance increases linearly and the light intensity decreases in parabola shape.

2.5.4 Challenge exercise – Magnetic fields vs distance

- Task 1: Repeat the above experiment with distance sensor and magnetic fields sensor against a wall and a strong magnet.

Chapter 3 – Autonomous vehicle challenges

3.1 Autonomous vehicles

We are in the generation of autonomous vehicles, machine learning and artificial intelligence. This is the world of machines making decisions. The decisions are according to the software and programming behind. This is just the beginning.

We can understand this world and the occurring changes by trying to develop programs similar to autonomous car.

The SENSE is a tool for such challenge exercises.

This chapter introduces several of the challenge autonomous exercises. The idea is to let the user to think about algorithms and solutions to solve these challenges.

3.2 Programming tips

This chapter is built as a challenge exercise to solve with no guiding.

The Robockly and the Python are rich and powerful coding programs.

They have so any functions and options.

Start your solutions with the Robockly, but observe the Python program too.

When the screen of the Robockly becomes too loaded, move to Python.

Try to work more with functions and not with long chains of instructions.

There are multiple solutions. Try to find the most efficient way.

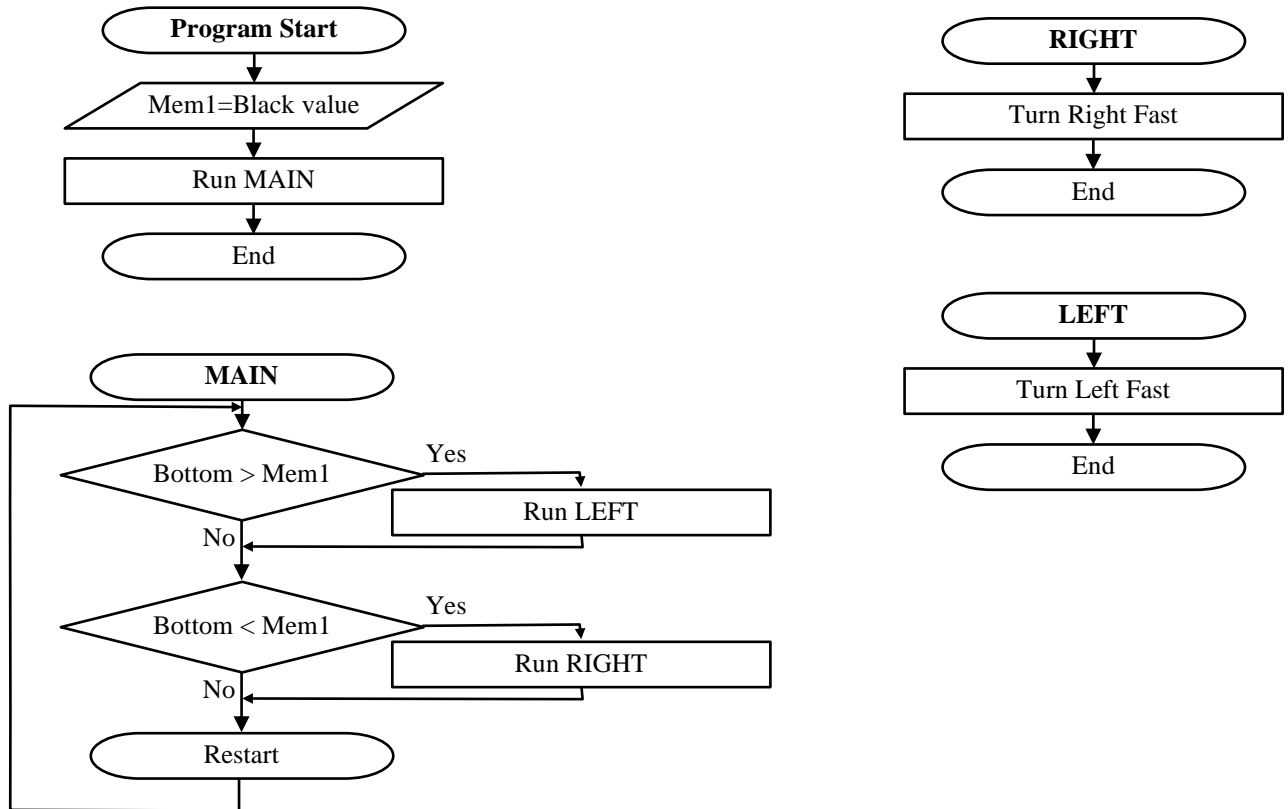
Do not be afraid to fail and to try repeatedly.

Good Luck!!

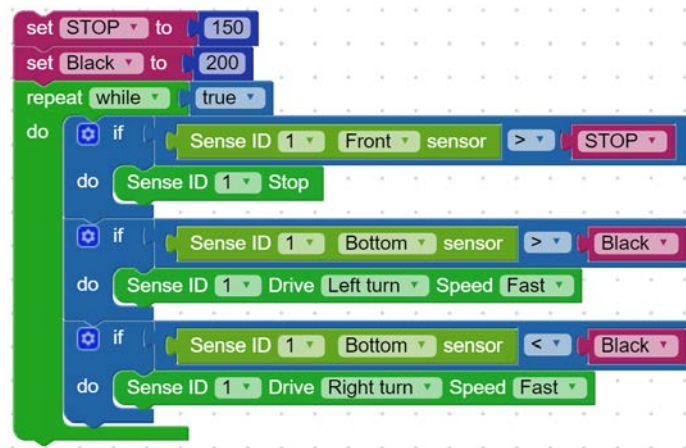
Challenge 3.1 – Along black lines

3.1.1 Left and right along a black line

The following is the flowchart of a simple movement along a black line program.



The following is Robockly program for the above flowchart.



STOP = None
Black = None

```
STOP = 150
Black = 200
while True:
    if (sense_get_input( id=1, input="front" )) > STOP:
        sense_stop( id=1 )
    if (sense_get_input( id=1, input="bottom" )) > Black:
        sense_drive( id=1, command="left_turn", speed="fast" )
    if (sense_get_input( id=1, input="bottom" )) < Black:
        sense_drive( id=1, command="right_turn", speed="fast" )
```

The robot moves by swinging on the edge of the black line.

We use the Robockly **Direct** mode to find the **Black** value for the memory1 definition.

Place the robot on the black line, read the bottom sensor value and set it in the program. This value may be different from one robot to another.

Download, run and check to robot movement.

3.1.2 Smooth movement along a black line

In order to get a smoother movement, we can replace one of the turn instructions with a deviate instruction. This depends on the robot movement direction.

When the robot moves counter clockwise, we shall replace the **Right turn** command with **Right deviate**.

When the robot moves clockwise, we shall replace the **Left turn** command with **Left deviate**.

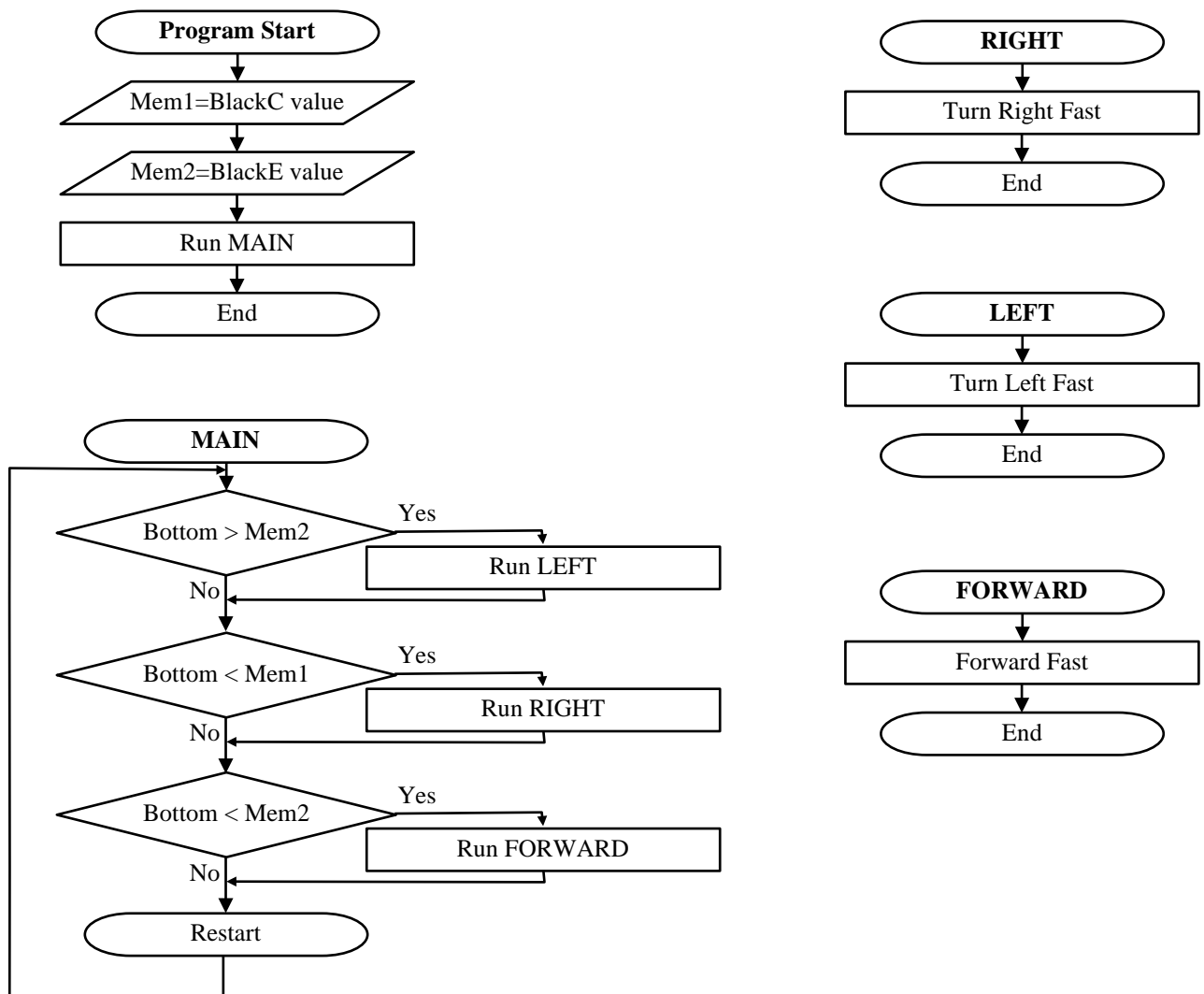
Change the program accordingly, download, run and check to robot movement.

3.1.3 Adding Forward movement

Check at **Direct** mode, the bottom sensor value when it is above the center of the black line and when it is closer to the edge of the line.

We shall call the read value at the center of the black line **BlackC** and the black value close to the edge **BlackE**.

The following program drives the robot forward when it is on the edge part of the black line.



Analyze the flowchart.

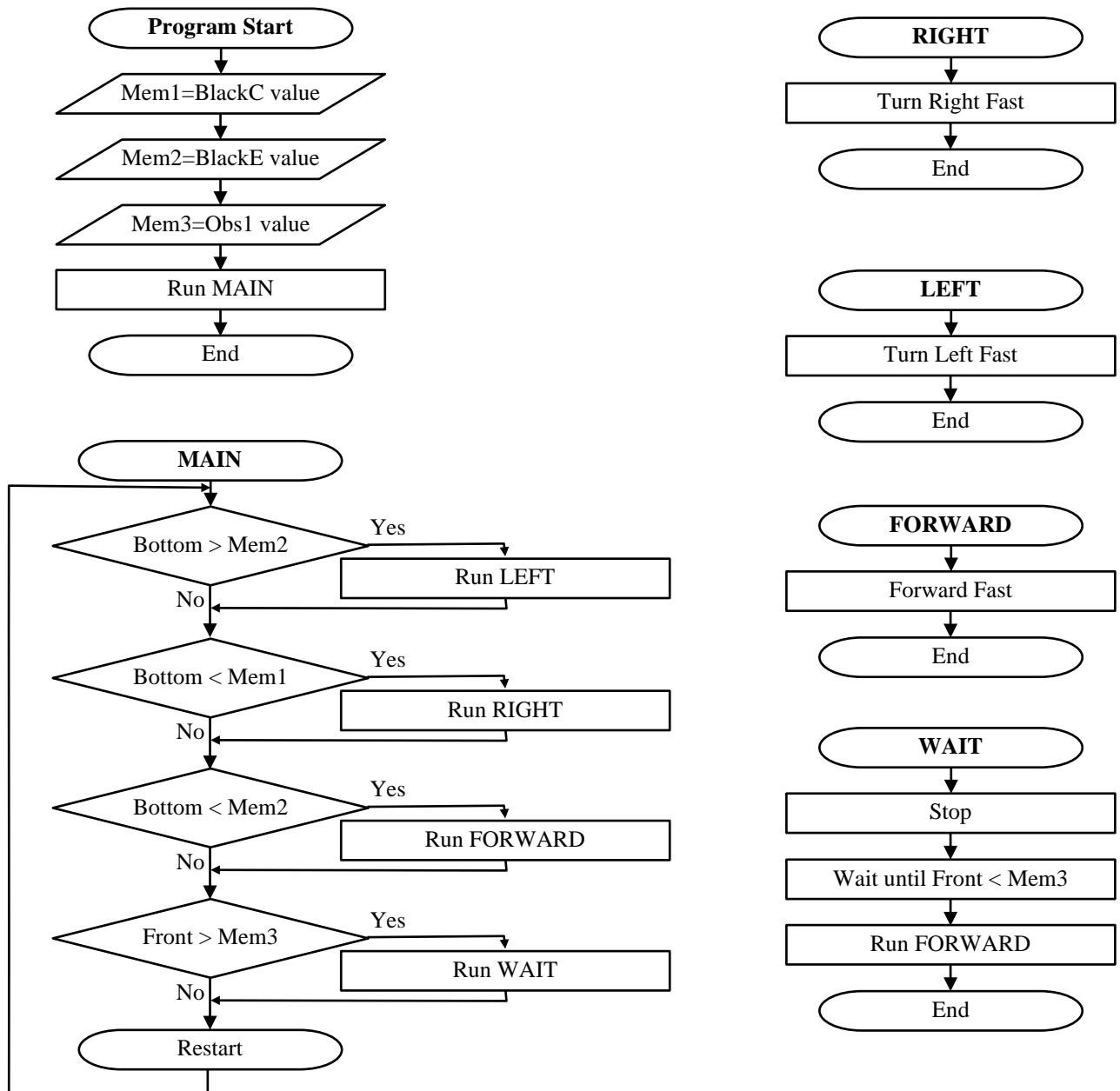
Change the program accordingly, download, run and check to robot movement.

3.1.4 Along a black line with a stop in front of an obstacle

Improve the previous program to stop in front of an obstacle until the obstacle is removed.

Put your hand in front of the SENSE and check at **Direct** mode, the front sensor value. We shall call the read value Obs1.

The following program drives the robot forward when it is on the edge part of the black line.

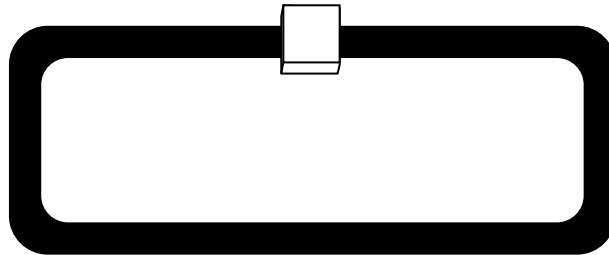


Analyze the flowchart. Change the program accordingly, download, run and check to robot movement.

Challenge 3.2 – AGV – Automatic Guided Vehicle

An AGV is a vehicle or a cart that moves along guidelines. It is very popular in manufacturing places for transporting raw materials or sub-assembly systems from one station to another.

Create the following line.



Put a small box on it as in the picture.

Write a program that moves the SENSE along the line and stops in front of the box for 5 seconds, turns around, moves on the other direction and vice versa.

The SENSE goes on the outer edge.

For this task we have two movements – clockwise and counter clockwise.

The main program should know what the current movement is. To determine that, we use what we call a flag. The main program operates the required procedure according to the value of a certain variable.

The value of this variable is changed when changing direction is needed.

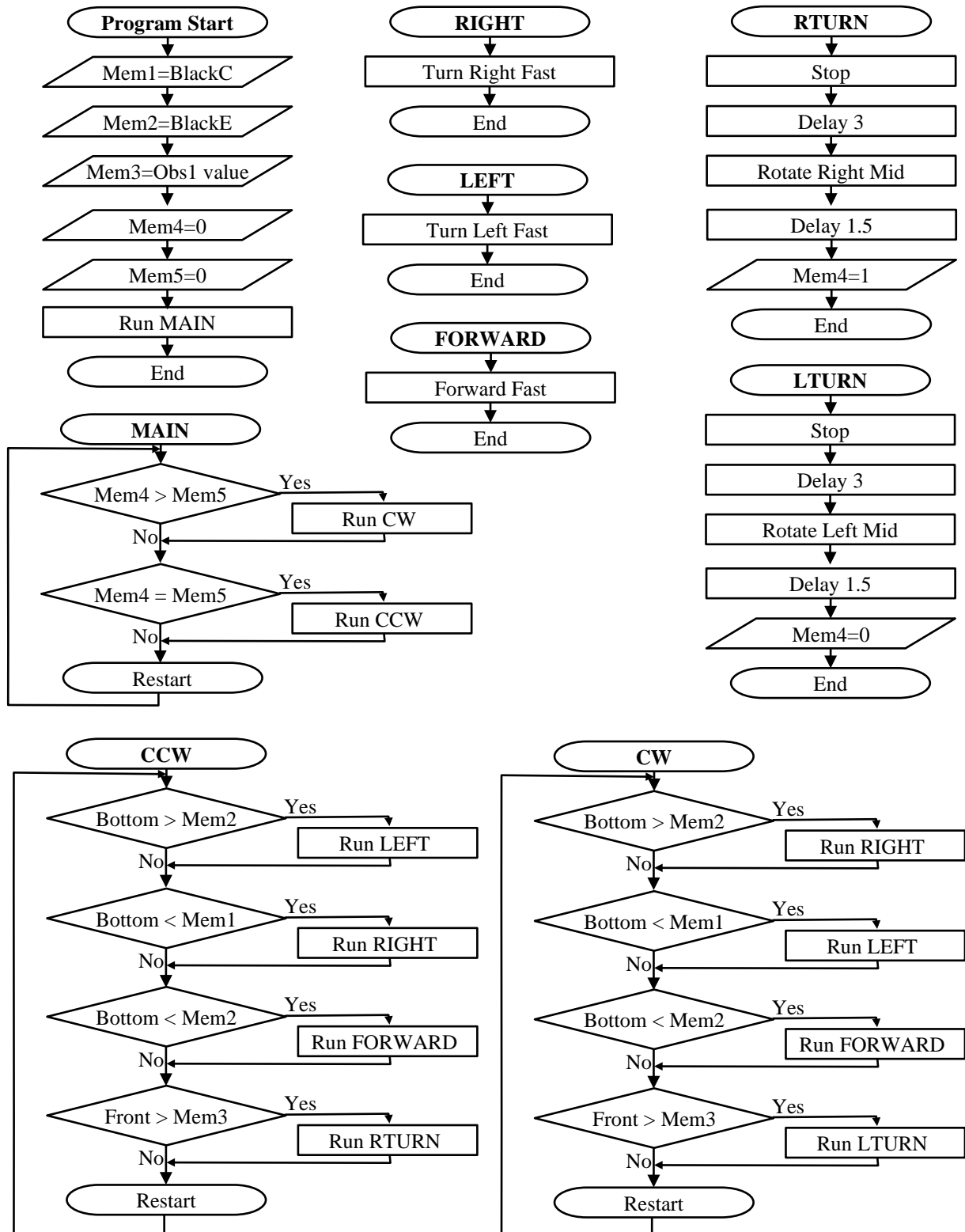
Analyze the following flowchart. Memory 4 is the flag variable.

Build the program accordingly, download, run and check to robot movement.

Every robot behaves a little different.

Adapt the program to your robot sensors and behavior.

Take care to stop in front of the box in a distance that enables the robot to rotate.

AGV Program flowchart

Challenge 3.3 – AGV between stations

Create the following line with the boxes.



Write a program that moves the SENSE from one station to another along the lines in this order: 1-2-1-2-...

The Sense stops at each station and moves to the next station when you put your hand close to the right back sensor.

Hint:

The previous AGV program should answer the movement of the robot.

Challenge 3.4 – Along a building block

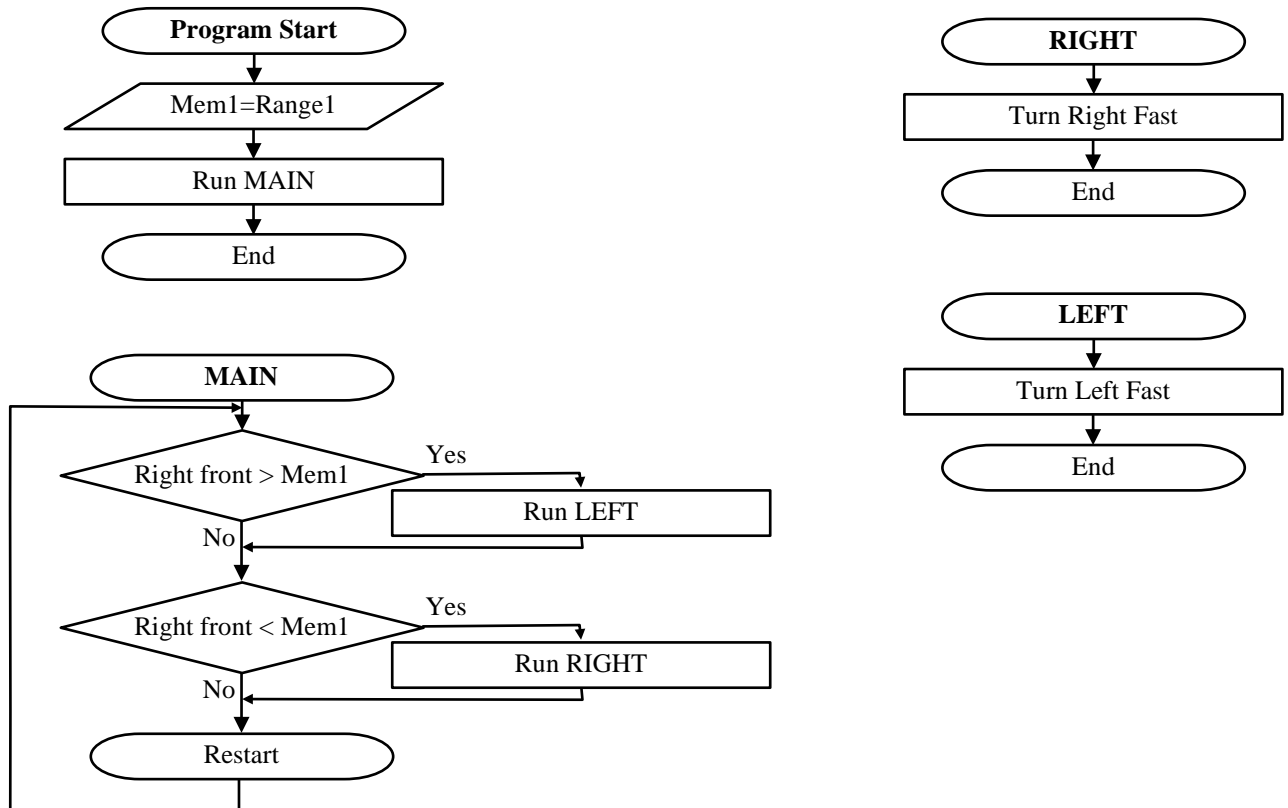
The following exercises deal with movement methods along walls and around a building block.

Use at least 40 X 40 cm box as a simulation of the building block.



3.4.1 Left and right along walls

The following is the flowchart of a simple movement along walls program.



The robot moves by swinging along a wall on its right side.

We use the Robockly **Direct** mode to find the **Range1** value for the memory1 definition.

Place the robot near the box on its right side, read the right front sensor value and set it in the program. This value may be different from one robot to another.

Download, run and check the robot's movement.

3.4.2 Smooth movement along a black line

In order to get a smoother movement, we can replace one of the turn instructions with a deviate instruction. This depends on the robot movement direction.

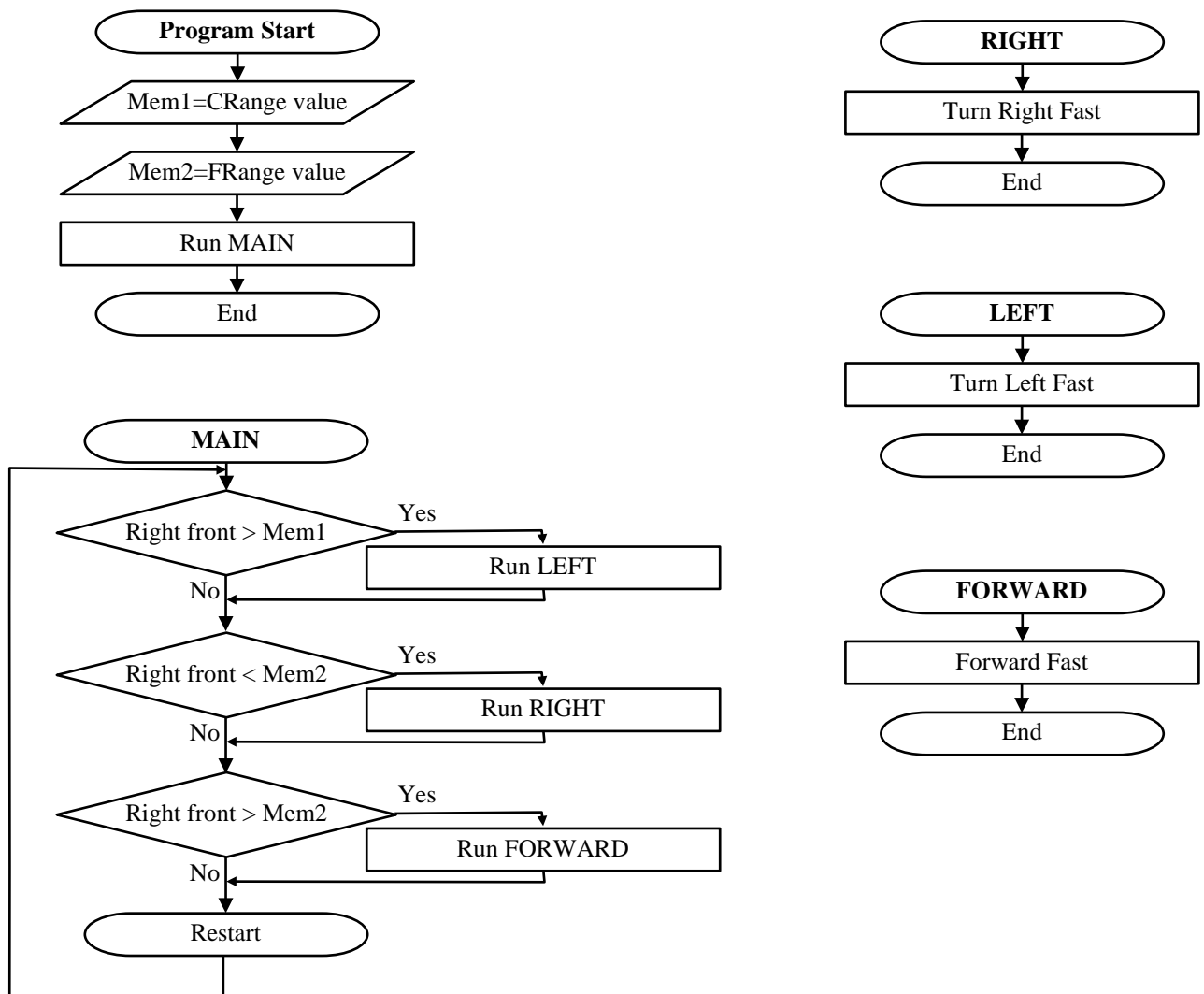
With the above program, the robot moves clockwise, we shall replace the **Left turn** command with **Left deviate**.

Change the program accordingly, download, run and check to robot movement.

3.4.3 Adding Forward movement

We can use two range values – **CRange** (close range) and **FRange** (far range).

The following program drives the robot forward when it is between CRange and FRange.



Analyze the flowchart.

We have to remember that the right front value increase when the robot come closer to the wall.

Determine the **CRange** value as the **Range1** value of the previous program.

Determine the **FRange** value as **CRange** – 10.

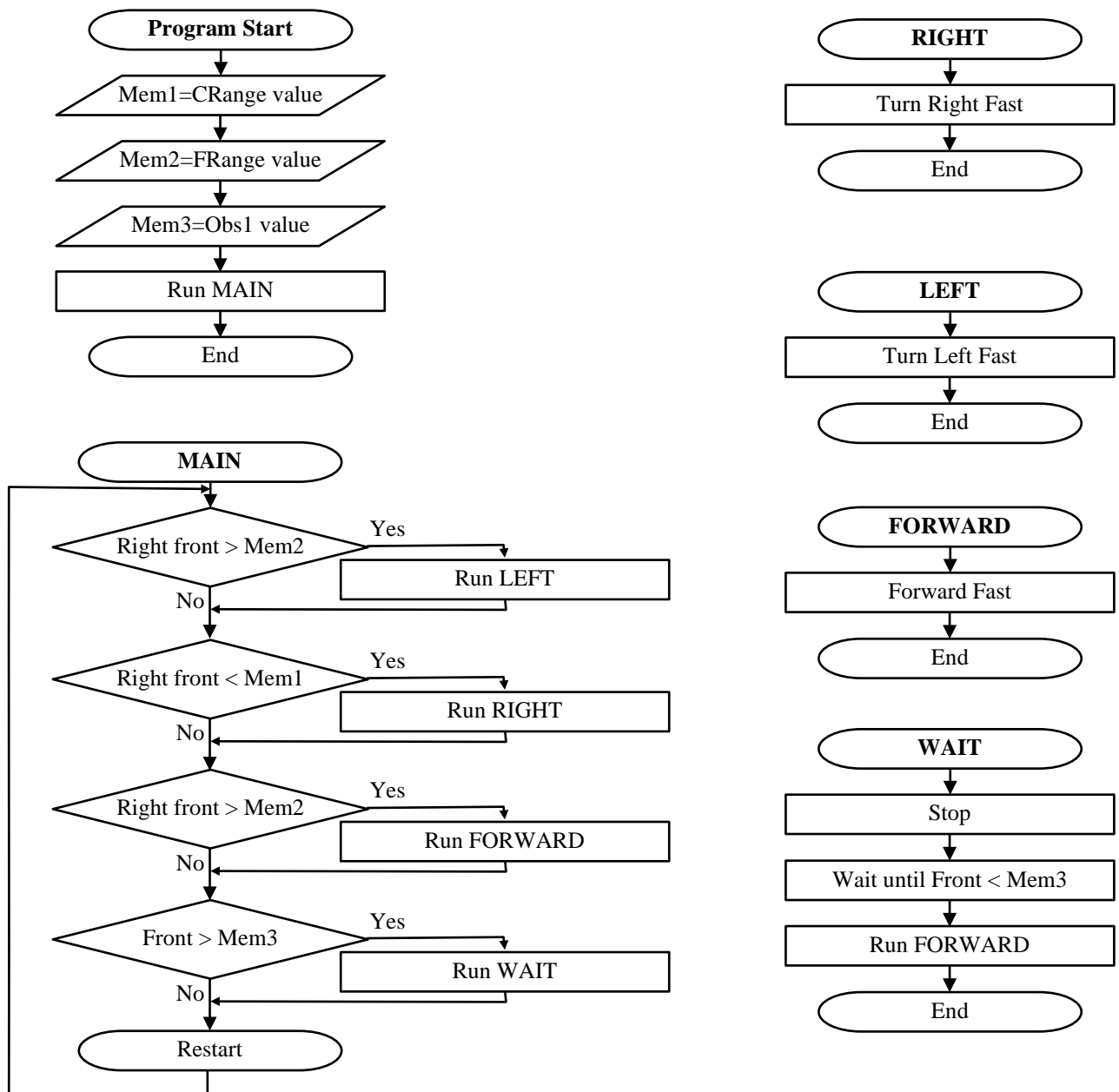
Change the program accordingly, download, run and check to robot movement.

3.4.4 Along a wall with a stop in front of an obstacle

Improve the previous program to stop in front of an obstacle until the obstacle is removed.

Put a small box in front of the SENSE and check at **Direct** mode, the front sensor value. We shall call the read value Obs1.

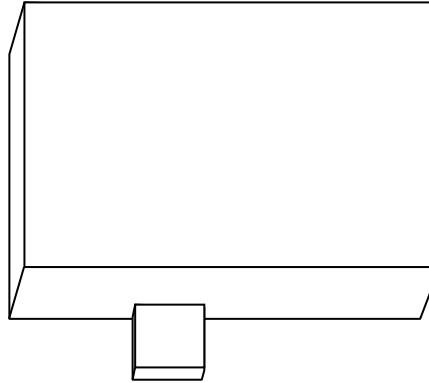
The following program drives the robot forward when it is on the edge part of the black line.



Analyze the flowchart. Change the program accordingly, download, run and check the robot's movement.

Challenge 3.5 – Along a building block and bypass cars

Put an obstacle, as described in the following picture.



Write a program, as in challenge 3.4.4, with **SENSE** bypassing the car. The **SENSE** should return to the right only after passing the car.

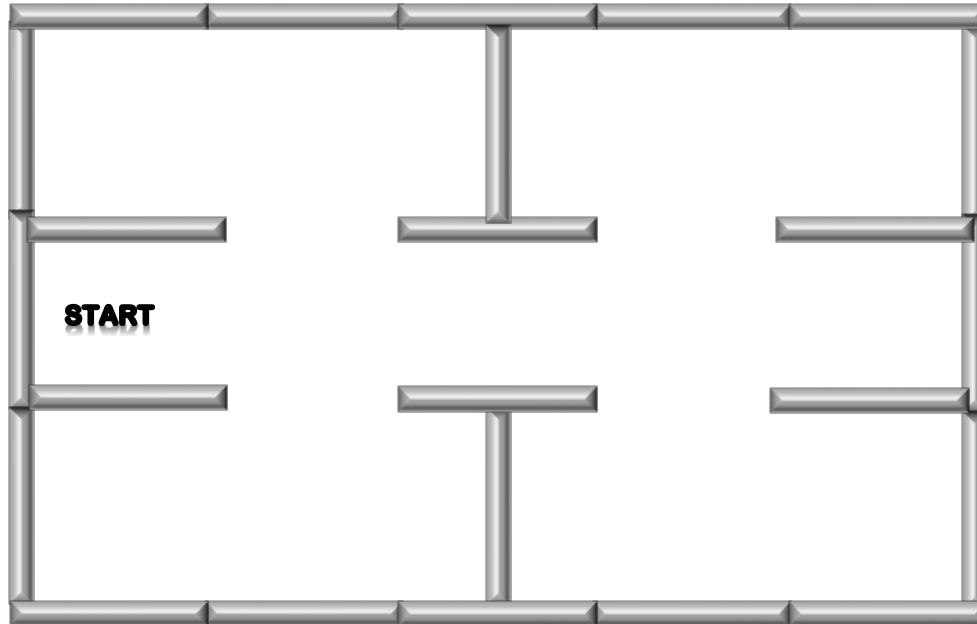
Hint:

Replace the **WAIT** procedure with **TURN** procedure.

The **TURN** procedure rotate to the left until the front sensor is below the **Obs1** value.

Challenge 3.6 – Autonomous museum guard

Build a model of a museum with rooms and corridors as follows:



Create a program that moves the robot along the walls through the museum rooms. The starting point is at the START position.

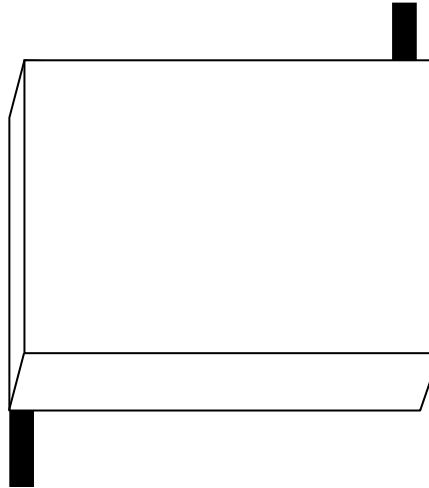
Hint:

The program of challenge 3.5 can serve as a solution for this task.

You have to adapt the memory values to the model.

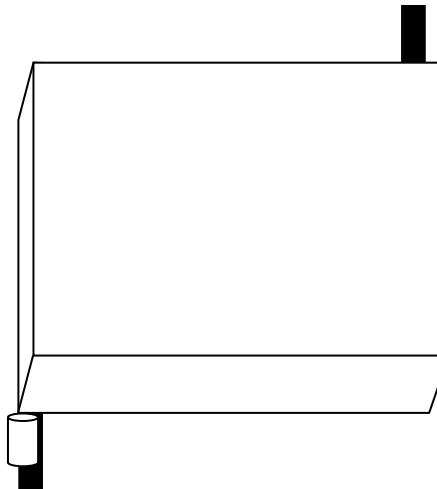
Challenge 3.7 – Along a building block with stop sign

Put black lines at the corners, as described in the following picture.



Write a program, as in challenge 3.4, with SENSE stop at the black line for 3 seconds.

Challenge 3.8 – Along a building block with stop for pedestrian



Write a program, as in challenge 3.4, with SENSE stop at the black line for 3 seconds.

It does not move on if an obstacle is in front of it.

The rod simulates a pedestrian.

Challenge 3.9 – Building block guard



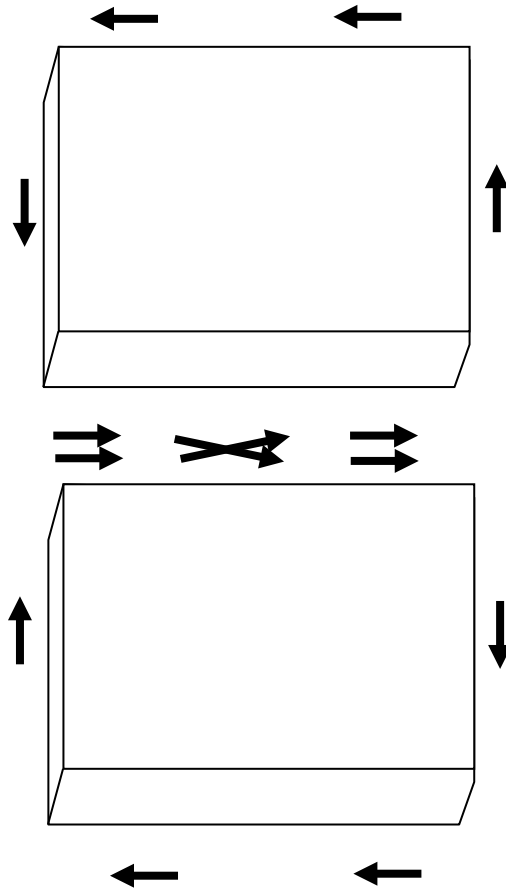
Write a program, as in challenge 3.4, with SENSE stop for 10 seconds after each round.

The program has to count the corner turns.

Hint:

- The SENSE has two range sensors on each side.
- The movement along the wall is according to the front side range sensor.
- When the robot is in a corner, the back side sensor moves away from the wall.
- The program should check the value of the back side sensor.
- When the value is low (the sensor is far from the wall), the robot should call a turn procedure with increasing the corner number.
- After counting four corners, the robot should stop for 10 seconds and then starts again.

Challenge 3.10 – Two buildings guard

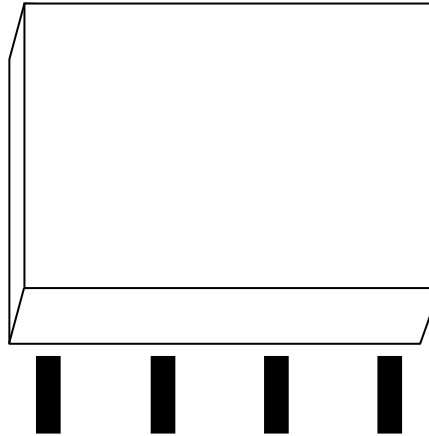


Write a program for the SENSE to go around the two buildings as in the above picture. The robot starts at the road between the two blocks.

The program has to count the corner turns and to change from moving counterclockwise around one building to clockwise around the other building.

Challenge 3.11 – Taxi driver

Put black lines along the building, as described in the following picture.

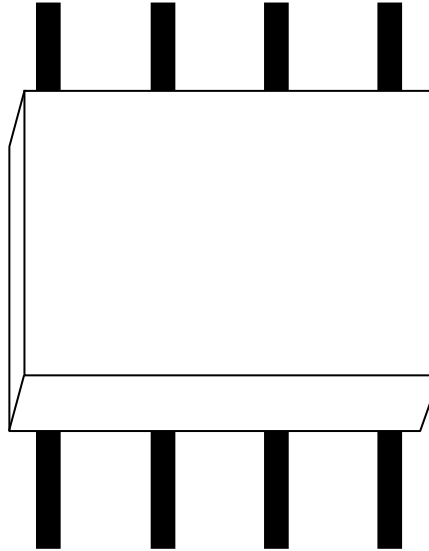


Write a program that moves the SENSE along the building and stops it on the third black line.

Start with the SENSE on the other side of the building.

Challenge 3.12 – Taxi driver with passenger

Put black lines along the building, as described in the following picture.



Write a program that moves the SENSE along the building and stops on the third black line for 5 seconds.

After that, the robot continues to the other side and stops on the second black line.

Challenge 3.13 – Home vacuum cleaner robot

Build a model of a room as follows:



Create a program that moves the robot along the walls in different distances from the walls.

At the first round, the robot will move closer to the walls and at the second round, the robot will move 8cm from the walls.

