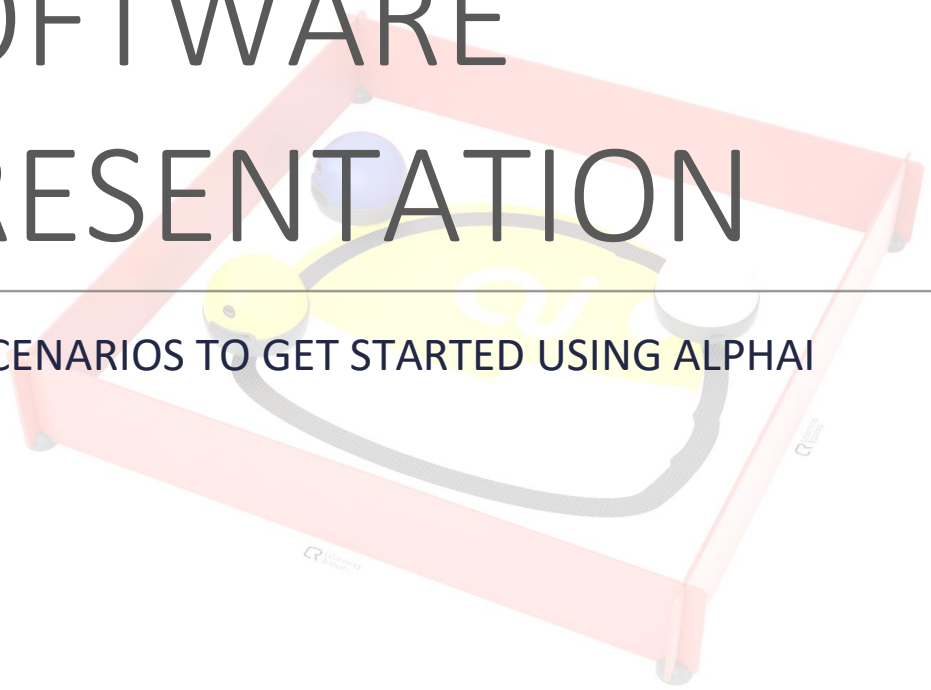
alphai, Education to Artificial Intelligence

SOFTWARE PRESENTATION

AND SCENARIOS TO GET STARTED USING ALPHAI



LEARNING ROBOT | CONTACT@LEARNINGROBOTS.AI

INTRODUCTION

AlphAi aims to teach the basis of **Artificial Intelligence** by handling a learning robot and teaching it an intelligent behavior.

What is Artificial Intelligence? We will not discuss this matter in depth in the introduction, but you will find more on that later in this document and in our online resources. Let's just say AlphAi and its interface will show a *new way of programming*, where instead of giving instructions, **machine learning algorithms** are being used.

This is the reason why the graphical user interface does not resemble known development interfaces like Scratch. What appears in the center of the window is an **artificial neural network** that will be parameterized so it will have the right **learning**. When the learning is done, the neural network will be able to correctly interpret its inputs, the measure from the robot's sensors, to give as an output the right **predictions** of the best action to perform.

This document starts with a **general presentation** of the functions of the software. Some of those descriptions may seem complex? Don't panic! The upcoming **scenarios** in the second part will suggest an experimentation trail using the robot that will introduce each notion one after the other.

TABLE OF CONTENT

SOFTWARE INSTALLATION GUIDE

WINDOWS

Download the latest version at the address: https://learningrobots.ai/downloads/software

Follow the instructions from the installation wizard.

A cmd.exe window opens and proceed to the software and its environment (Miniconda 3 and PuTTY) installation.

LINUX & MAC

There is no installer for Linux and Mac yet (Mac version is on its way), hence you shall this few lines in the terminal.

Install Python: we recommend you install Anaconda distribution

https://www.anaconda.com/products/individual#download-section

Ouvrez un terminal et tapez les commandes suivantes :

Open a terminal, and write those lines:

```
conda create -n alphai python=3.7.6
conda activate alphai
conda install pyqt pyqtgraph vispy param scipy pillow
conda install pyserial imageio
conda install pytorch torchvision cpuonly -c pytorch
pip install gym pyperclip pyserial
```

Go to https://learningrobots.ai/downloads/software/ and download the latest zip version of the AlphAi code source. Extract the zip file where you prefer.

Using Mac, be careful, extracting the zip file by clicking on it may result in an unexpected result. In that case, mark down the path to the file containing AlphAI-xxx.zip. You can get it in the Finder by right-clicking the folder, press the Control key et select “copy […] as a path” then open a terminal, go to that directory using the `cd` command and enter the command `unzip AlphAI-xxx.zip`

To launch the software, mark down the path to the folder and in a terminal, enter the command:

```
cd path/to/AlphAi
conda activate alphai
python Main_server.pyc
```

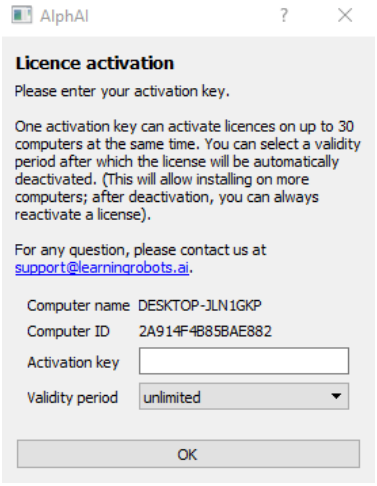
LICENSE ACTIVATION

For your first use of the software, an activation key is required.

Fill the “Activation key” field using the key that has been provided with the robot (XXX-XXX-XXX)

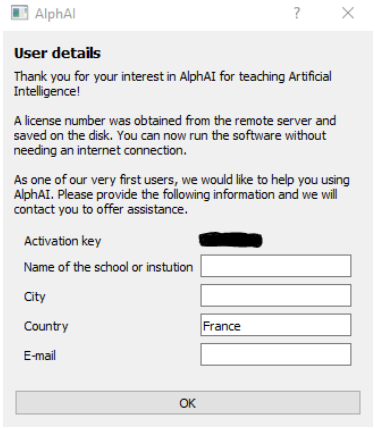
Each activation key allows up to 30 simultaneous use of the software.

For a temporary use (for instance on a student’s computer), you can indicate a validity period, upon which the license will be revoked from this computer and will be available of use for another one.



The following dialog box asks you to provide some basics information.

This step is not mandatory, but recommended so we can provide assistance in need and information about future updates to come.



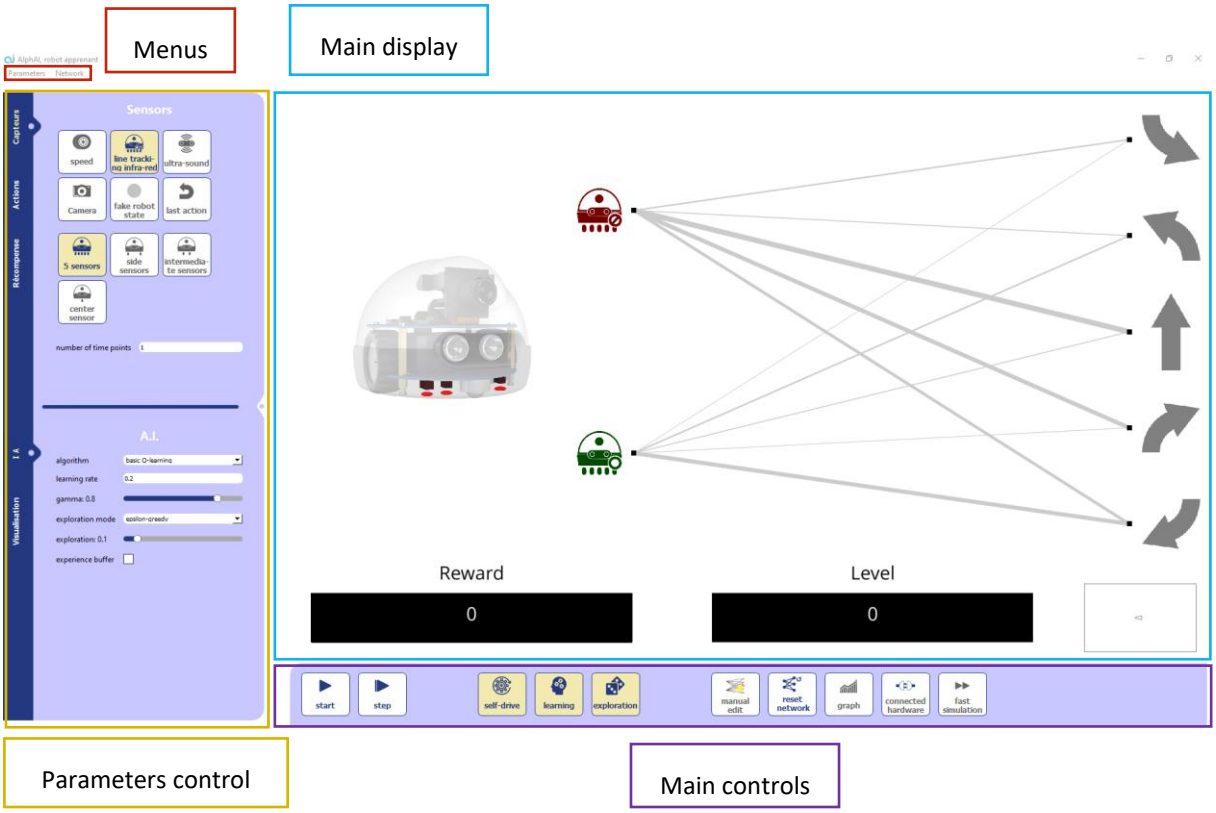
SOFTWARE GENERAL PRESENTATION

The software has two main components:

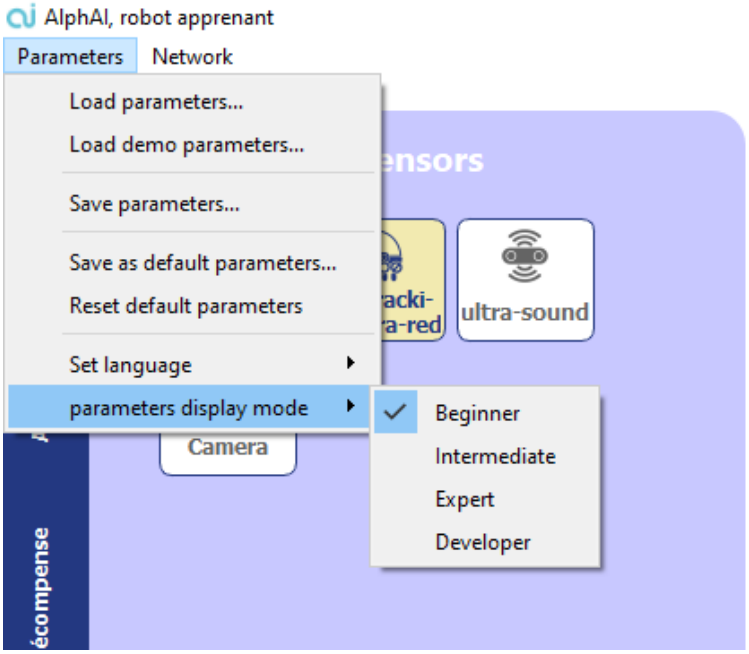
The main **display screen** show the state of the robot, of the learning in progress, of the neural network.

The control interfaces get fractioned into:

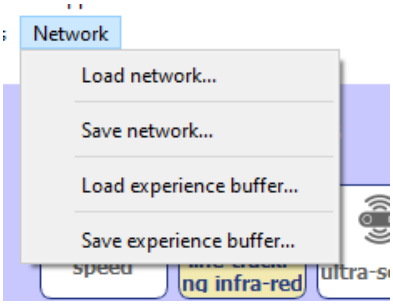
- The main control bar in the lower side
- Parameters control on the left-hand side
- *Menus*.



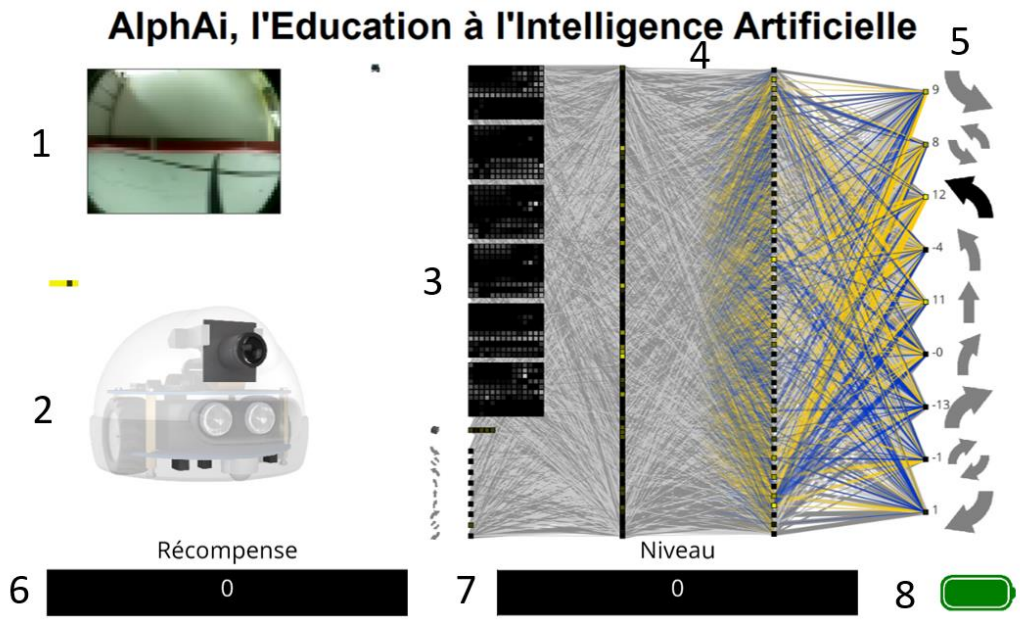
The **Parameters** tab allows you to load predefined parameters or to save current parameters. It also allows to display parameters in beginner, intermediate or expert mode. This will change present options.



The Network tab allows to change the state of the network and of the experience memory or saving the current one.



MAIN DISPLAY



The main screen shows the status of the robot and its training.

1. Display of the robot's camera.

2. Representation of the robot to visualize the sensors that are activated.

3. The activated sensors of the robot. These are the inputs of the neural network.

4. The neural network. You can see its evolution in real time (yellow/blue: transmission of positive/negative activity; green/red: reinforcement/decrease of connections during learning). The thickness of the lines is proportional to the weight of the connection.

5. The possible actions for the robot. These are the outputs of the network. You can click directly on them to force the robot to do an action.

6. Reward: reward that the robot receives for each action during a reinforcement learning.

7. Level : average of the rewards over the last 2 minutes.

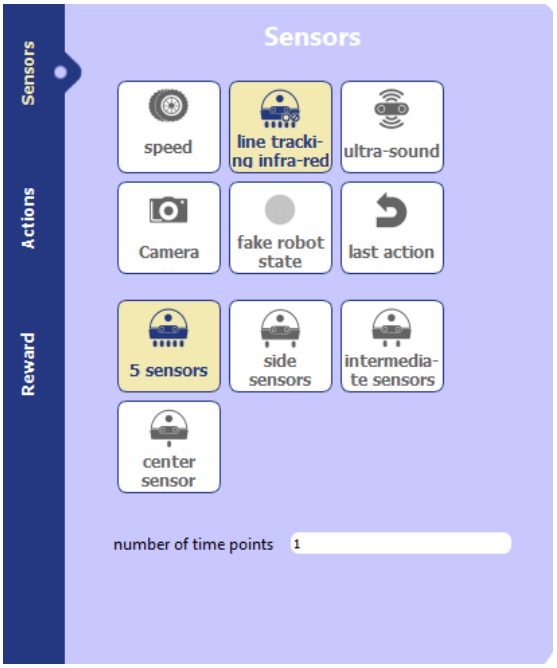
Battery level of the robot. When the robot is not connected, a representation of the simulated robot is displayed instead. The arena is represented by a rectangle, the robot by a triangle. (The front side is the smaller side).

MAIN CONTROL BAR

Please note: All controls display a little help when you hover the mouse over them for a few seconds.

- start / stop Start/stop the robot
- step Start the robot step by step (the robot will only take one step each time you press the button)
- self-drive Self-drive (when it is deactivated, the robot only performs the actions you enter)
- learning Learning (allow the robot to change its neural network to learn).
- exploration Exploration (allow the robot to make a random decision from time to time)
- manual edit Edit the neural network manually
- reset network Reset the neural network
- graph Display the progress graph of the robot (not available in beginner setting)
- connected hardware / connected hardware Connect to the robot
- fast simulation Accelerate the simulation (only for the simulated robot)

SENSOR TAB



The **Sensors** window allows you to choose the robot sensors that will be used. This window also allows you to set some speed parameters of the robot.

- 

Detect if the robot is moving or not
- 

Line tracking sensor
- 

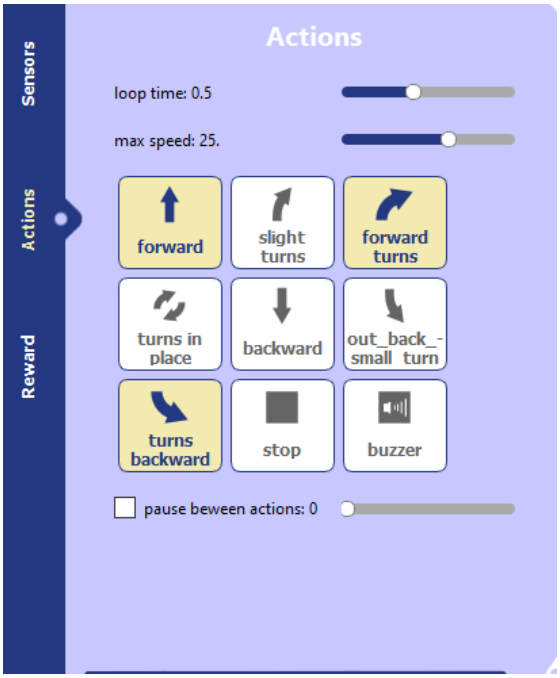
Ultra sound: returns the distance between it and an obstacle
- 

Ultra sound: returns a binary response depending on whether it detects an obstacle in front of it or not (to be defined with a certain threshold)
- 

Robot camera (different image formats are available)
- 

Last action: all the actions chosen for the robot will be added to the network entries. When the robot does an action, it will be activated in the entries at the next step.

ACTION TAB

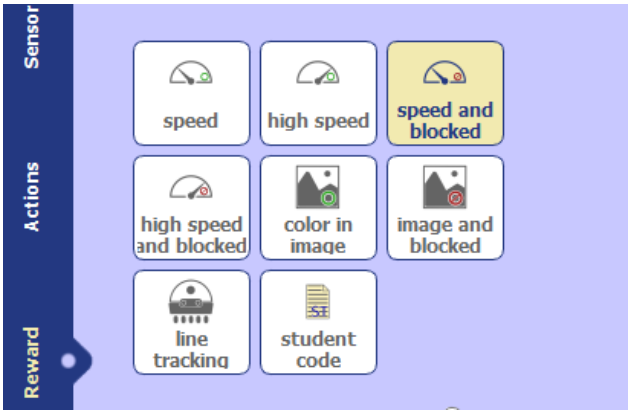


The **Actions** window allows you to choose the actions that the robot can perform.

(The asterisk means that the action will appear twice in the list of actions: one for the right and one for the left)

- Go straight ahead
- Turn*
- Slight turn*
- Turns in place*
- Turns backward*
- Backward
- Stop

REWARD TAB



The Rewards window allows you to choose the type of reward during reinforcement learning.

- 

Reward if the robot goes fast
- 

Reward if the robot goes fast but high speeds are even more rewarding
- 

Reward if the robot goes fast, punish if it stops
- 

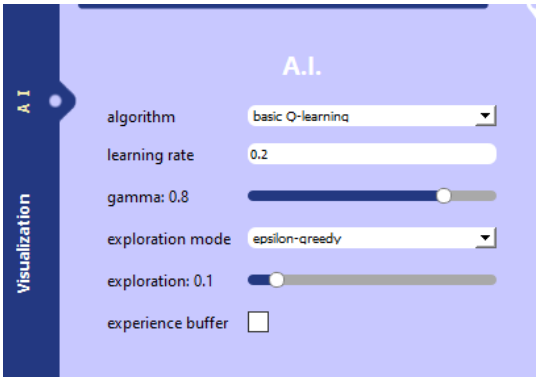
Reward if the robot goes fast, punish if it stops, but high speeds are even more rewarding
- 

Reward if many pixels of the camera are of a certain color (you can choose the color by varying the parameters that are then displayed)
- 

Reward if many pixels of the camera are of a certain color, punish if the robot stops
- 

Reward if black is detected just below it

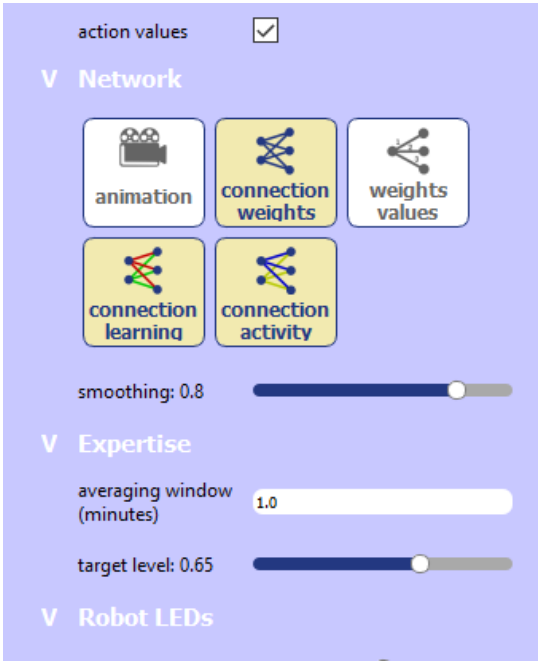
A.I. TAB



The A.I. window allows you to choose how the robot is trained, in particular to choose between supervised learning and reinforcement learning (Deep Q Learning). It also allows to set the parameters of the different algorithms.

- - **Algorithm** : Select the AI algorithm used
- - **Framework** : Select the Python framework used to build the neural networks
- - **Optimizer** : Optimization algorithm used to evolve the neural network
- - **Learning speed** : Increase for faster learning... but decrease if divergence errors appear
- - **Gamma** : Adjust the importance given to immediate rewards (value close to 0) compared to more distant rewards in time (value close to 1)
- - **Exploration mode**: Select the exploration strategy
- - **Exploration**: Frequency of exploration (value between 0 and 1)
- - **Intermediate layers of neurons** : Number of neurons in each intermediate layer (for example : put nothing to directly connect inputs to outputs, put "100 50" for two intermediate layers of respectively 100 and neurons)
- - **Convolutional layer** : Configuration of convolutional neuron layers : layer is defined by three parameters : number of kernels, filter size, and stride (i.e. image size reduction factor). The parameters of a layer are separated by "," while the layers are separated by commas ",". For example "4/5/2, 8/5/2" configures 2 layers, the 1st with 4 kernels, the 2nd with 8 kernels, and both with filters of size 5 and stride 2.
- - **2 neurons per binary variable** : Check to have 2 neurons for true or false inputs (always one and only one will be activated) ; Uncheck to use only 1 neuron
- - **Regularization** : Positive or null parameter limiting the intensity of the connections.
- - **Neuronal** bias : Check to allow neurons to adjust their activation threshold (this means that all neurons receive a constant input that they can adjust, not represented in the graphical interface)
- - **Experience memory** : Check to allow the AI to continue learning from past actions and rewards

VISUALIZATION TAB



The View window allows you to choose what is displayed or not on the main screen.

- 

Animate the activity in the network (movement from inputs to outputs)
- 

Show network connections
- 

Display the weights of the connections
- 

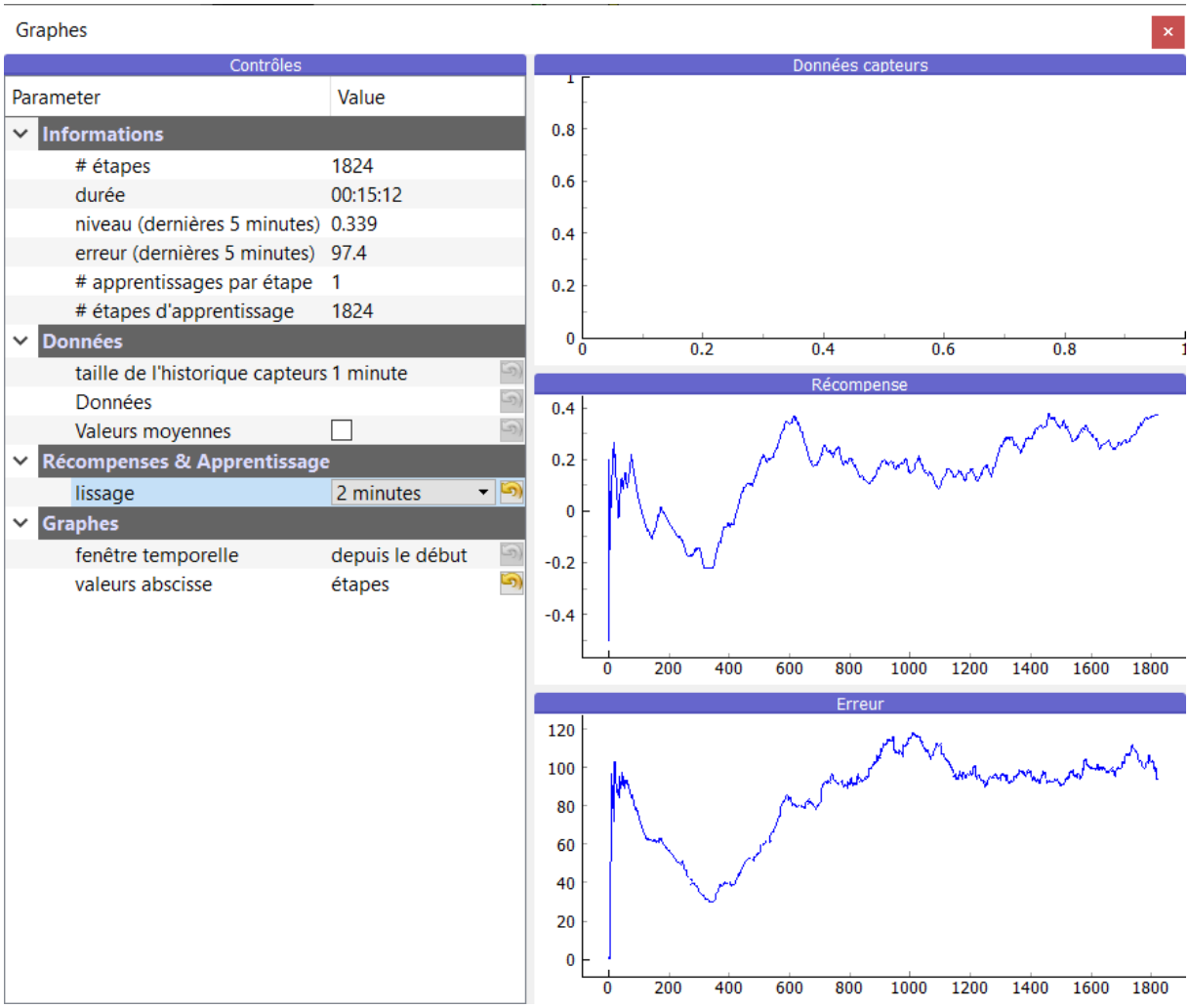
Display learning (green / red colors for connections that intensify / diminish)
- 

Display the activity in the network (yellow / green color for excitation / inhibition activities)

PROGRESSION GRAPH

This graph allows to follow the progress of the robot by displaying its rewards and the error of its predictions over time. If the robot is progressing well, the reward graph should increase and the error graph should decrease.

These graphs are not necessarily very readable, in this case you can average them, for example over 2 minutes, by clicking on smoothing - 2 minutes



ROBOT LEARNING SCENARIOS

We will now present different scenarios that give an overview of what it is possible to do with the software and what it is possible to teach the robot.

For your own experience with the robot, we recommend that you run these scenarios one after the other in the order described below. Then you can build your own sequence of scenarios, and of course invent your own scenarios to fit the time you have with your students, the target audience, the level of detail of what you want to teach, etc.

By default in most scenarios, the 

, 

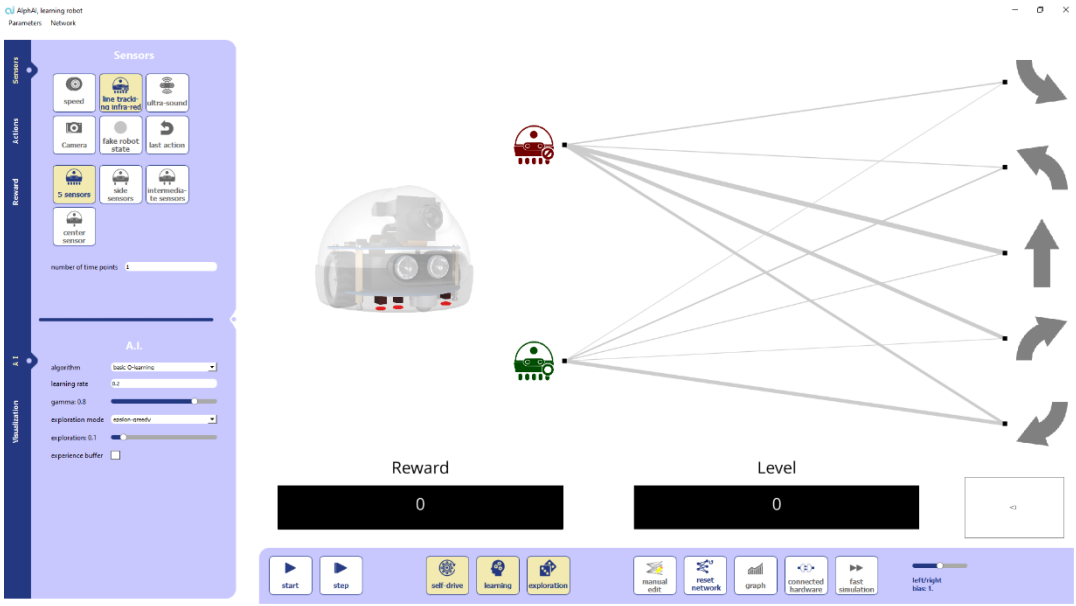
, 

icons in the bottom tab will be enabled, and the possible robot actions will be  . If this is not the case, it will be indicated in the instructions

GETTING STARTED WITH THE SIMULATED ROBOT

When the software is launched, the robot is not yet connected. But you can still run the software with a simulated robot. This allows to test the different methods without having to use the real robot.

Launch the software.



On the bottom right you can see the simulated robot. The robot is represented by a triangle, the smaller side being the front of the robot, and the arena by a large rectangle.

Start the simulated robot. 

You should see the simulated robot move. Usually it starts by turning on itself, then "discovers" the straight line, then "discovers" to turn around when it is blocked by an obstacle.

Notions learned:

We note that the behavior of the simulated robot has improved over time. The great progress made in the field of Artificial Intelligence at the beginning of the 21st century is essentially due to the improvement of **automatic learning** techniques, also known as **machine learning**.

We are also beginning to get acquainted with the **graphical interface of AlphAi**: for the moment, we can be satisfied with paying particular attention to the Start/Stop button, to the improvement of the simulated robot's behavior and to the corresponding increase in its level.

You don't need to understand everything at this level: it will be more interesting to do so when you switch to the real robot!

GETTING STARTED WITH THE REAL ROBOT.

Now we will connect to the real robot! To do this, we must first set up its arena.

Set up an arena: If you are building your own arena, here are some tips:

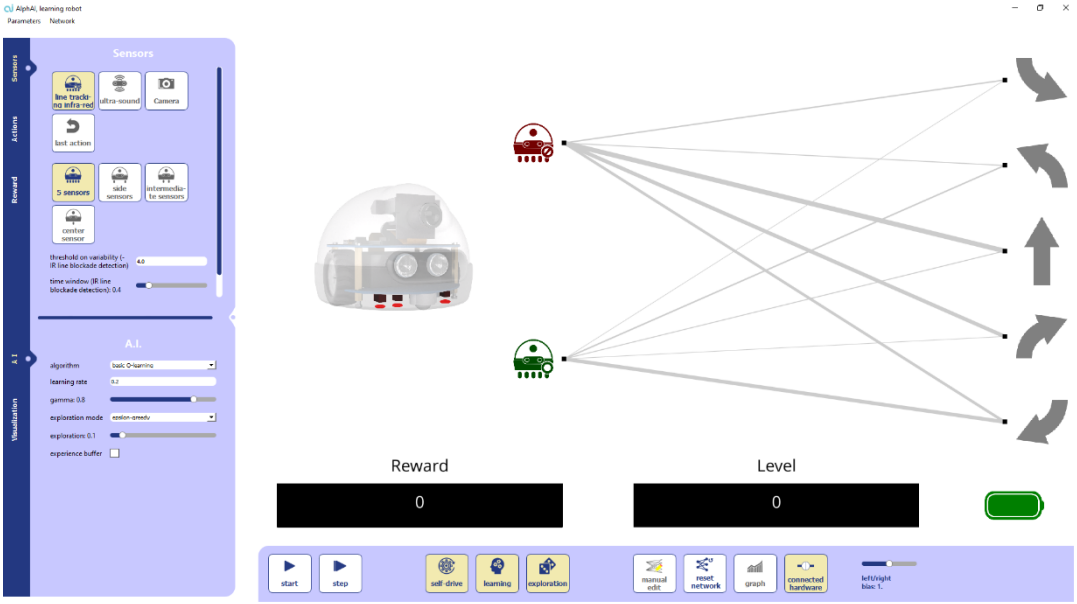
- Dimension: square or rectangle of 80cm to a few meters side.
- Walls can be boards or any type of object (ream of paper, etc.). Prefer surfaces that can slightly absorb shocks.
- For effective learning when the robot navigates with its camera, make sure that all walls have the same color, which is a different color than the floor.
- Prefer a slightly textured floor. Indeed, when the floor is too smooth or too uniform in color, it can happen that the robot has difficulties to determine if it is moving (the detection of the movement is done by the infra-red sensors located under the robot, it is the same problem as when one uses a computer mouse on a too smooth surface).
- Please make sure that there is no dust on the floor, as it damages the motors. Dust the floor with a cloth before use, do not run the robot on carpeting.

The arena proposed by Learning Robots

Turn on the robot (the switch is on the bottom). It makes a small movement when it is ready. Connect to the robot's wifi (look for the wifi that starts with ALPHAI: the password is identical to the wifi name).

On the software, press the connect button 

You are now connected to the robot. The battery level should be displayed at the bottom right (check that the robot is well charged)



Start  the robot. Check that the robot is moving correctly. You can also give commands directly to the robot by pressing the arrows on the right of the screen or by using the directional arrows on the keyboard.

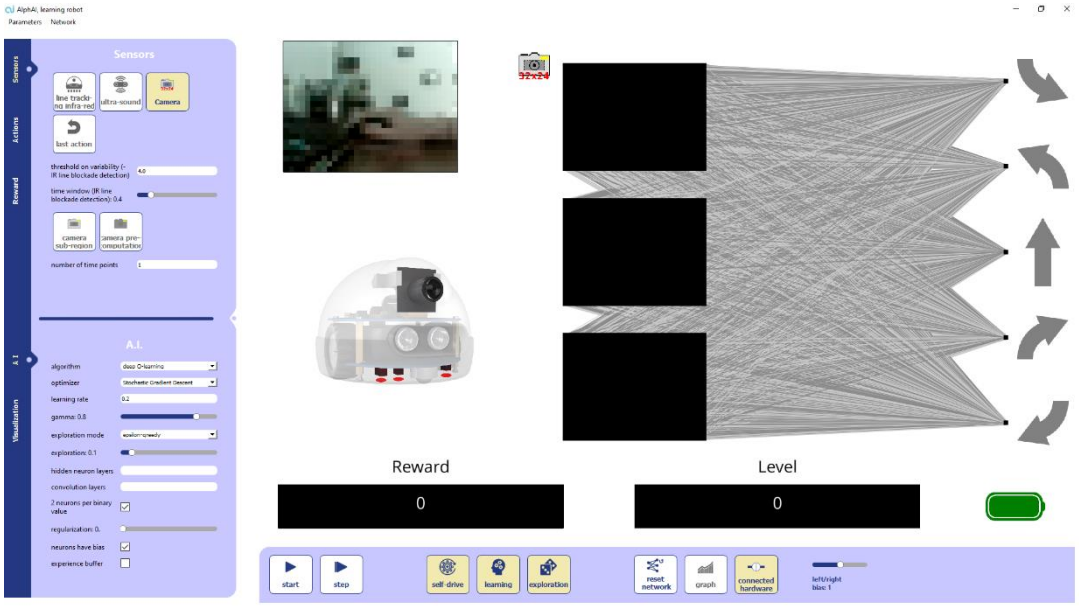
IMAGE RECOGNITION WITH THE CAMERA

We are going to see one of the most used learning categories in artificial intelligence; supervised learning. The goal is to use the robot's camera to make it recognize people and categorize them. At first, we will have to tell the robot how to classify each person, but then it will be able to recognize them by itself.

Load the "Image recognition with camera" configuration. Or to select parameters yourself, start the program and:

- In the AI tab:
  - Select "Supervised learning"
  - Set learning speed to 0.05
  - Check memory of experience
- In the Sensors tab, select:
  - 

"camera" (for instance 32x24).
- In the action tab, put as many actions as people/objects to be recognized.



- Hold the robot in your hands, so that it does not move even when its wheels are turning.

- Start 

the robot.

- Point to a person or an object with the robot's camera, and press several times on one of the arrows on the screen (on the right of the network). This arrow will be used to identify the person or object.

- Repeat this operation on other people or objects, assigning them different arrows each time.

- Look back at the different people or objects you pointed to; the robot will be able to recognize them by pointing to the correct arrow that identifies them (check this by looking at the arrow that appears on the screen).

You can point out to the students that the robot does not really recognize the person or object because it analyzes the whole image, including the background behind it. If you change the place of a person or an object, the robot may not recognize it anymore.

Notions learned:

What is now called Artificial Intelligence is the automatic learning capabilities of computers (also called "machine learning").

The robot trains on training data where the right answers are given by the human operator, then it is able to generalize to new test data.

SUPERVISED LEARNING - NAVIGATION WITH CAMERA

We will see how to use supervised learning in order to direct the robot well in the arena. We would like the robot to go straight as often as possible, but avoid hitting the wall. As we activate the camera, the robot will be able to spot the walls. As before, the robot must first be shown how to act before being allowed to act by itself..

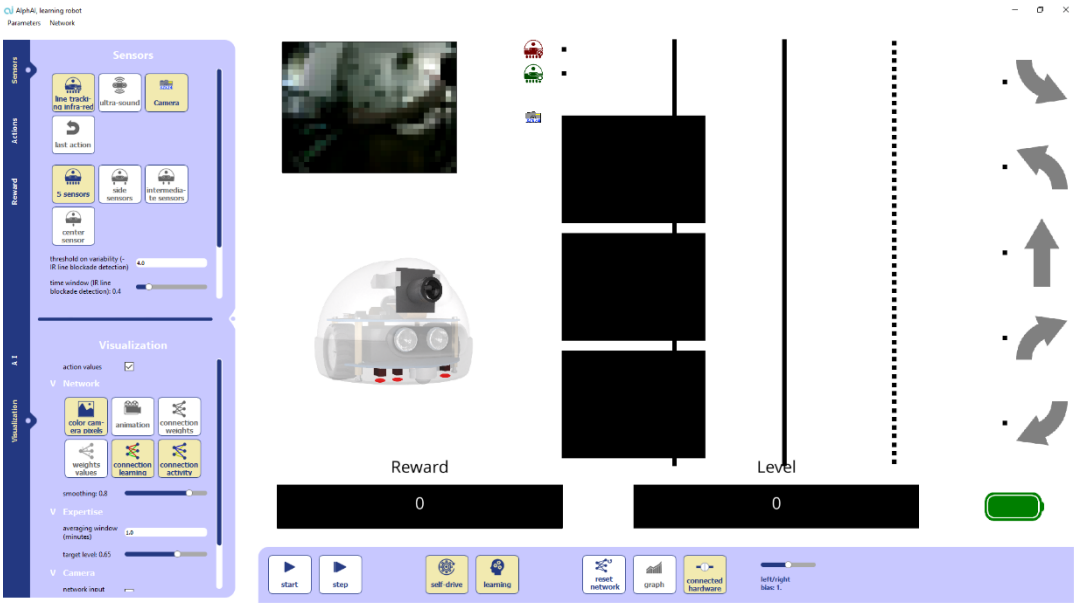
Load the "supervised learning - navigation with camera" configuration.

Or to select the parameters yourself, start the program then:

- In the AI tab:
  - select "Supervised learning"
  - in "intermediate neuron layers" put “100 100 50”
- In the Sensors tab, select:
  - 

“blocked/moving”
  - 

“camera” (for instance 32x24)
- (not required) In the Visualization tab, :
  - uncheck the neural network display options to ensure that there are no display delays due to the large number of connections.



- Start 

the robot.

- As long as the robot is not stuck, press go all must

- If the robot approaches a wall, press turn left/right

- If the robot gets stuck, press back left/right

Be careful; if you ever tell the robot to turn too early or too late, the robot will learn from a distorted database and will act in any way. Since it is not necessarily easy to press just at the right time, you can remove the self-

drive 

from the robot and give the robot step-by-step instructions. Then reactivate the autonomy and continue to correct the robot if it makes mistakes from time to time.

Notions learned:
In supervised learning, it is important to give correct instructions in the learning phase

MANUAL EDITING OF A SIMPLE NEURAL NETWORK

We will see in more detail how the connections in the neural network of the robot work with a simple case.

Load the configuration "manual edition of a simple neural network". Or to select the parameters yourself, launch the program and then:

- In the AI tab :
  - Select q-learning simple
- In the Sensors tab, select :
  - 

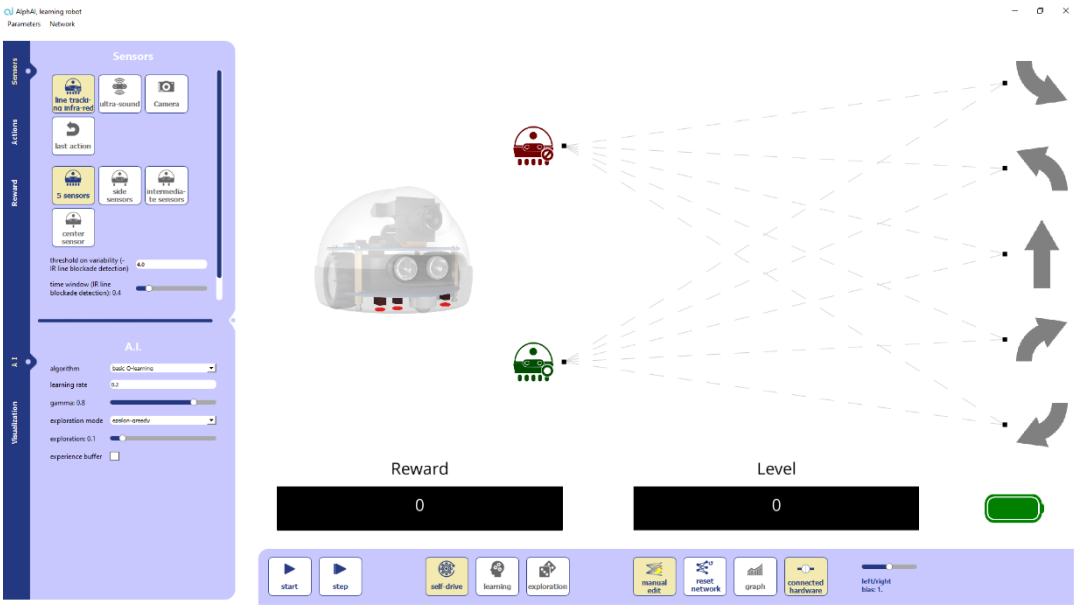
"blocked/moving"
- In the Reward tab :
  - choose "high speed, stop punishment" 

- In the lower section :
  - check the "simple manual editing" mode 

  - click on "reset network" 

  - uncheck learning 

and exploration 



The possible entries are :

- The robot is not blocked
- The robot is blocked

At the output, 5 actions are possible: back to the right, turn to the left, go straight, turn to the right, back to the left.

Here there is no learning, it is up to you to create the appropriate links between these inputs (sensors) and outputs (actions) to generate a coherent behavior of the robot. If you create a link between a sensor and an action, it will mean that as soon as the sensor is activated, the robot will perform the related action.

Draw the connections you think are good, then start 

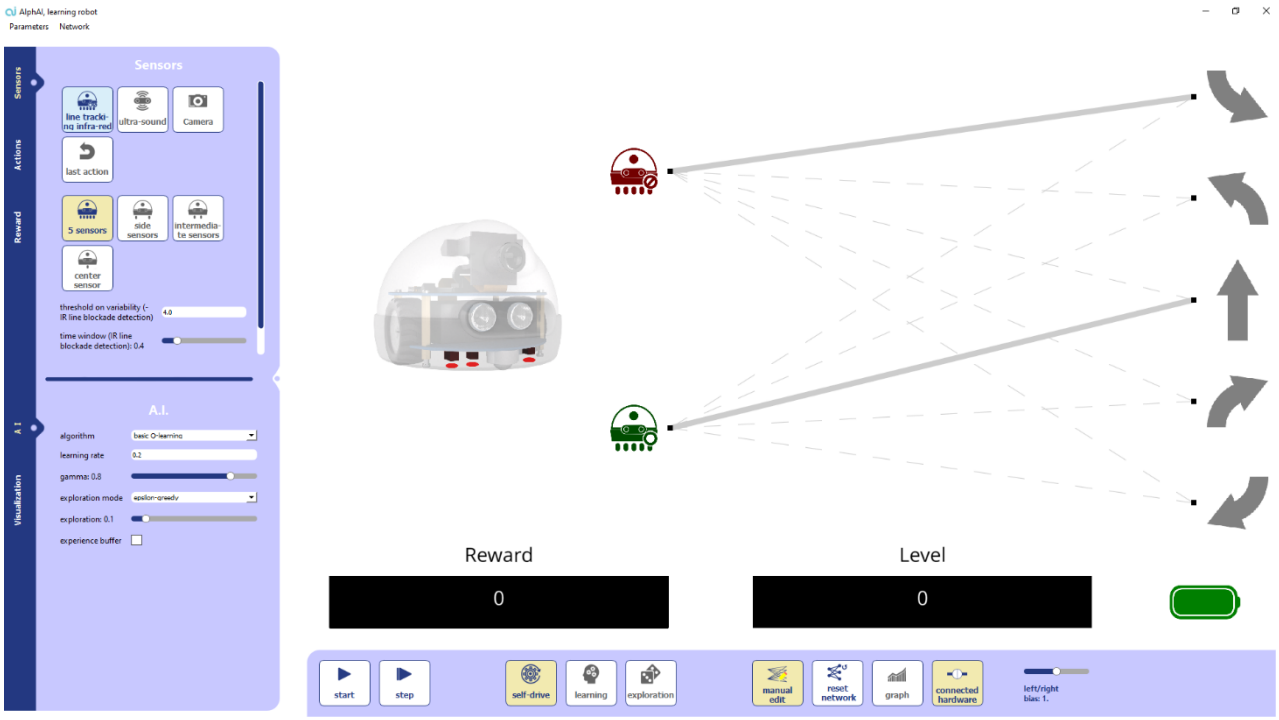
the robot. You can also change the connections after the robot has started.

Objective: To set up the connections that allow the robot to get the most rewards and thus reach the highest possible level. Hint: The robot gets rewards when it moves forward.

Solution: We would like the robot to go straight ahead as often as possible, and to go backwards as soon as it gets stuck. To do this, we need to tell the robot :

- If it is not blocked, go straight ahead
- If it is blocked, move back.

This is translated in this way:



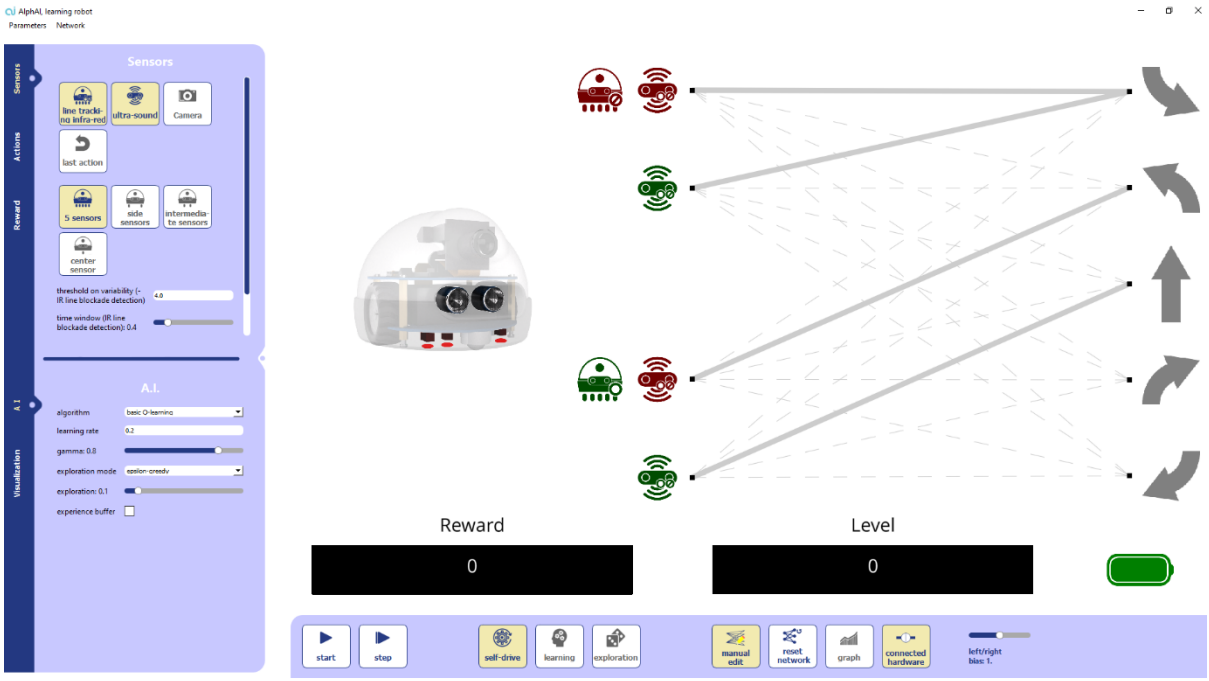
With these connections it should have the desired behavior.

Variant: to add a little complexity, we can take into account the ultra-soundsic sounds. In the sensor tab, select absence/presence of obstacle 

Now the robot can detect the walls thanks to these ultrasounds, so we can use them to make it anticipate the walls and take a turn before.

Solution : We have to tell the robot :

- If it is not blocked and there is nothing in front of it, go straight ahead
- If it is not blocked but there is an obstacle in front of it, turn
- If it is blocked, go backwards.



Start  the robot; it should behave as expected. The robot may have trouble anticipating steps that arrive in a perpendicular fashion. If the robot does not anticipate any walls, you may need to change the obstacle detection distance.

Notions learned:

In order for the robot to behave, we need to create connections between these sensors and the different actions it can do

REINFORCEMENT LEARNING - BLOCKED/MOVING NAVIGATION

Previously we have even created the network of connections so that the robot acts as we wanted. Now we are going to see that we can use artificial intelligence to make the robot build the same network by itself without human intervention. To do this we will use reinforcement learning.

The principle of reinforcement learning is to encourage the robot to do certain actions by giving it rewards or punishments. The robot will then modify its behavior in order to maximize its level, which is the average of its rewards over the last 2 minutes.

Here, the faster the robot goes, the more reward it will get, but if it stops, it will get a punishment. To maximize its level, the robot should go straight as often as possible and stop as little as possible.

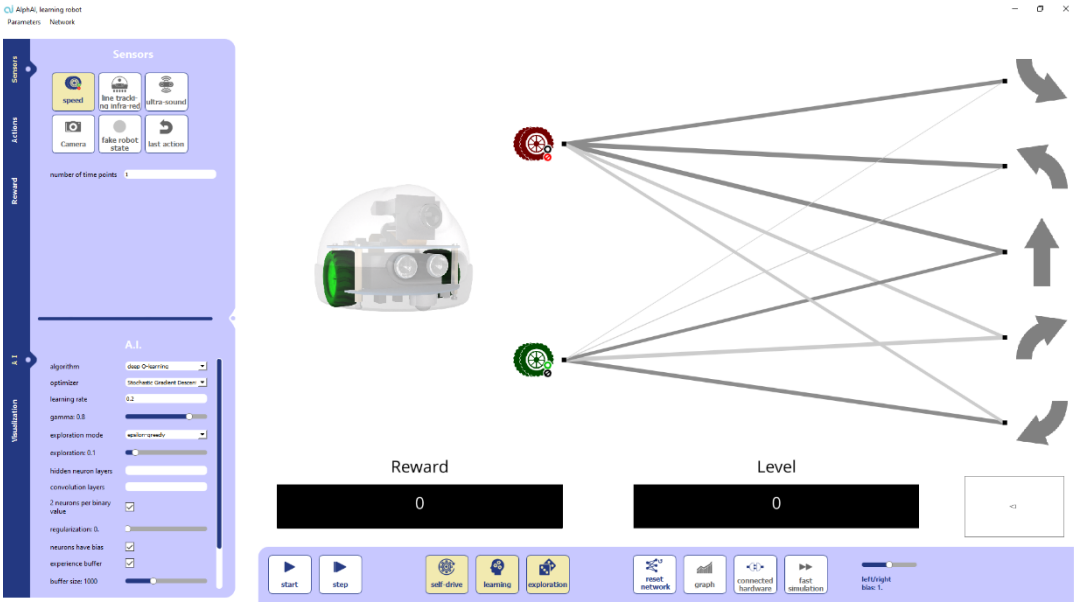
Load the "learning by reinforcement - navigation blocked/moving" configuration.

Or to select the parameters yourself, launch the program and then :

- In the AI tab:
  - Select Simple Q-learning
- In the Sensors tab, select:
  - 

"blocked/moving"
- In the lower tab:
  - Uncheck exploration 

- In the Reward tab:
  - select: "high speed and blocked" 



Unlike last time, the sensors are not connected to a single action but to all of them at the same time. Each connection has a different weight that corresponds to the reward the robot expects to receive by using this connection. For now, the weights are randomly initialized.

Start 

the robot. You can also launch it step by step 

to better understand what happens at each step.

Initially the robot is not stuck, for him the best action, the one that would bring him the most reward (the one with the strongest weight) is to go backwards, so he chooses this action. But since he doesn't get any reward, he decreases the weight of this stop, then he starts again. Since it still doesn't get a reward, it keeps decreasing it, until this weight becomes less than that of another stop. At this point, the robot thinks that the action that gives it the most reward is turning left. So it turns left, gets a reward higher than its prediction, so it increases the weight of this edge until it stabilizes at 30. At this point, the robot thinks that turning left is the best action, and since this action makes it turn without ever stopping, it will continue doing it indefinitely.

So the robot starts to turn left without stopping. But this is not its best action: it would have a better level if it went straight ahead. But since the robot has never tried to go straight, it cannot know this. So we have to force the robot to try.

Activate the exploration 

box in the bottom tab.

When exploration is enabled, the robot will not necessarily choose the edge with the highest weight, but will occasionally choose another direction. When the robot finally chooses the random go straight action, it will receive a high reward and increase the weight of this edge. After a while the weight of the ridge will exceed the weight of turning left, and the robot will start going straight instead of in circles.

After a while, the robot has the desired behavior (going straight as often as possible, backing up as soon as it gets stuck), but this time it has done it entirely by itself, without having to be told what to do.

Notions learned:

To allow the robot to learn without being supervised by humans, it is given more or less reward each time it performs an action. Depending on what it receives, the robot will adapt its choices in order to maximize its rewards.

The rewards in question are simply a numerical value, negative if it is a punishment, that the robot seeks to maximize.

The robot must also be allowed to explore and not always choose the action that seems best, because this will allow it to discover new actions that may prove to be even better.

REINFORCEMENT LEARNING - NAVIGATION WITH CAMERA

The principle is the same as before, but this time we use the camera of the robot. The inputs are very numerous (all the pixels of the camera and their composition of red, blue and green) and we need to add layers of neurons to analyze the image; we end up with a very complex network, that we cannot do by hand or try to understand step by step.

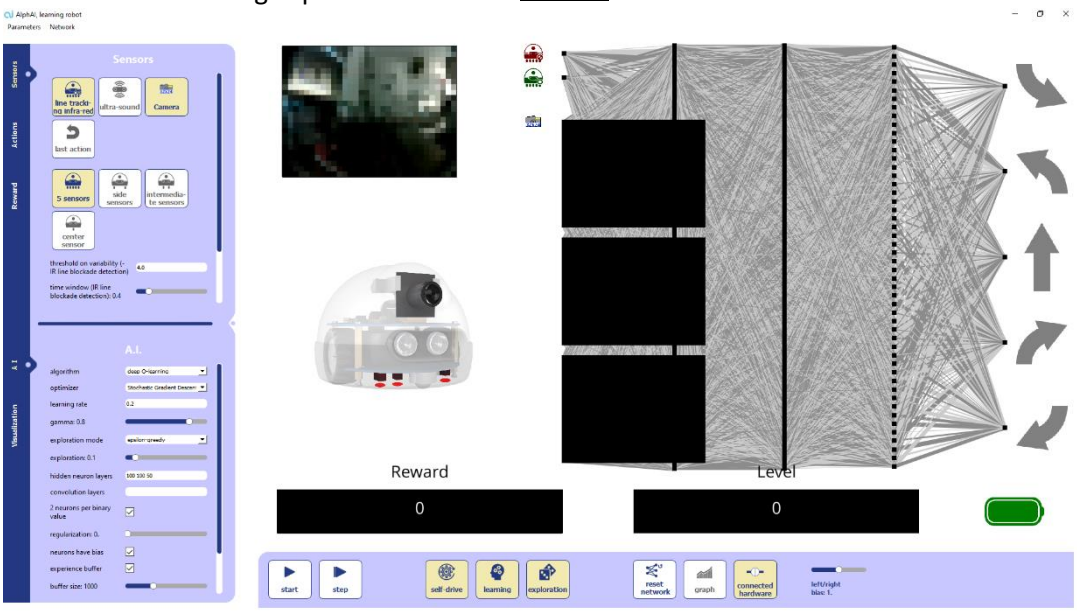
Load the "learning by reinforcement - navigation with camera" configuration.

Or to select the parameters yourself, start the program and then :

- In the AI tab :
  - select "deep Q-learning
  - in "intermediate neuron layers" put "100 100 50
  - check "experience memory
- - In the Sensors tab, select:
  - 

"blocked/moving"
  - 

"camera" (for instance 32x24)
- In the reward tab:
  - choose "high speed and blocked 



Start 

the robot. After a while (about 10 minutes) the robot will eventually run through the whole arena anticipating the walls.

Notions learned:

The beauty of artificial intelligence is that it can deal with huge networks and find the right connections that will allow the robot to behave in a way that would be impossible to try to do by hand or to intuit.

INTRUDER DETECTION

In the previous case we used layers of intermediate neurons to process very complex information. Here we will see a simpler case to understand what these intermediate neurons are used for.

Load the "intruder detection" configuration.

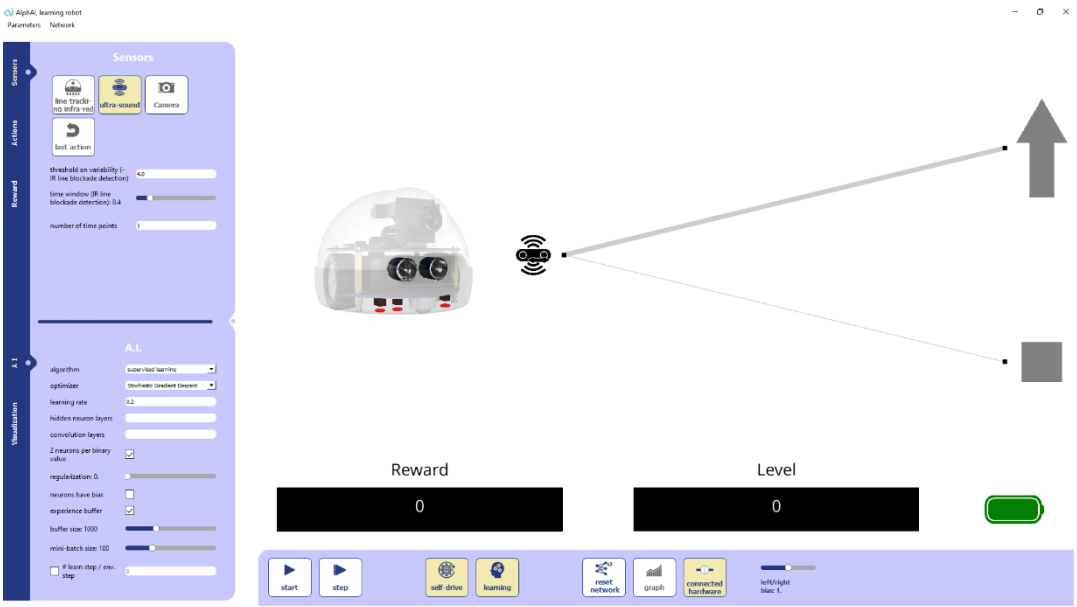
Or to select the parameters yourself, launch the program and:

- In the AI tab :
  - select "supervised learning
  - uncheck neurons have bias
  - check experience buffer
- In the Sensors tab, select:
  - 

ultra-sounds distance to obstacle
- In the Actions tab,
  - 

"forward"
  - 

"stop"



- Place the robot so that it can turn its wheels without moving.
- Place the robot facing a wall or a similar flat-sided object (e.g. a box). Leave some space between the robot and the wall.
- We want to tell the robot that this situation is normal; start 

the robot and press several times on  (the icon on the right)

The intruder that the robot will have to detect will be symbolized by an object with a flat side, for example a box. You can also use your hand if you have nothing else.

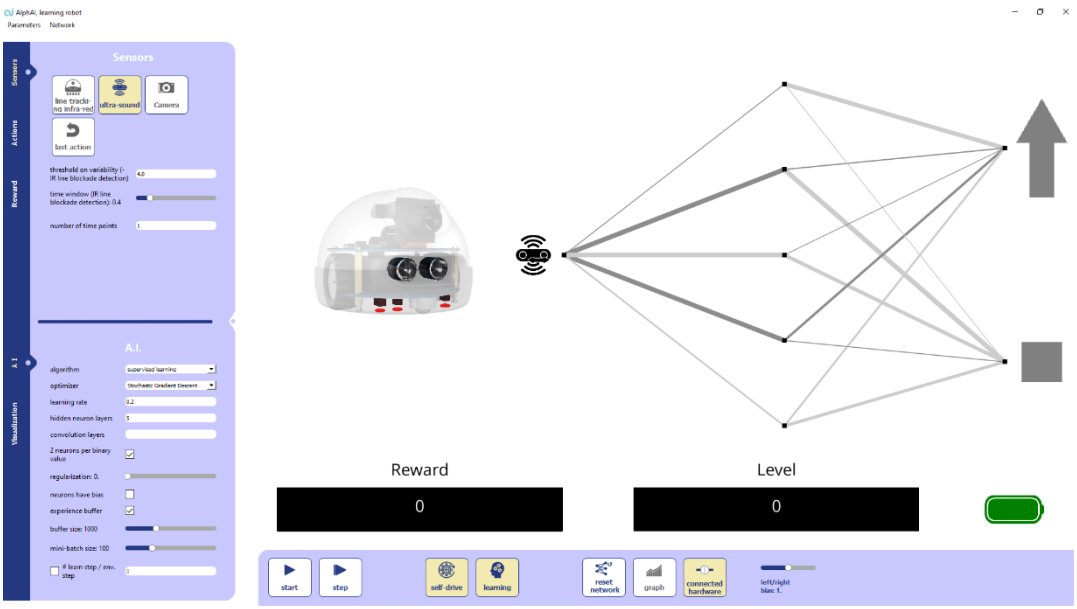
- Place the intruder between the robot and the wall, parallel to the wall, and press  .
- -Remove the intruder

The robot will continue to spin its wheels when it shouldn't. To solve this problem, we need to add a neural network.

- In the AI window, check Neural bias
- Re-train the robot as before.
- -Remove the intruder and place it back, still parallel to the wall; the robot should turn its wheels when it sees the intruder and stop when it is not there.
- -Once the robot is well trained, place the intruder between the robot and the wall, but this time at an angle. This time, the robot can't recognize it.

Because the intruder is at an angle, the ultra-soundsic beams are deflected instead of being sent directly back to the robot. To solve this problem, the robot must learn that if the distance calculated with the ultrasound is either too long or too short, then there is an intruder. To take into account this more complex reasoning, we need to add a layer of intermediate neurons.

- -In the AI window, in intermediate neural layers, type 5.



Teach the robot to behave:

- When there is no intruder, press 
- - When there is an intruder, either parallel to the wall or at an angle, press 

After training, the robot has the expected behavior.

Notions learned:

Adding intermediate neurons is used to process information further, and in particular to process it with linear functions.

BALLOON TRACKING

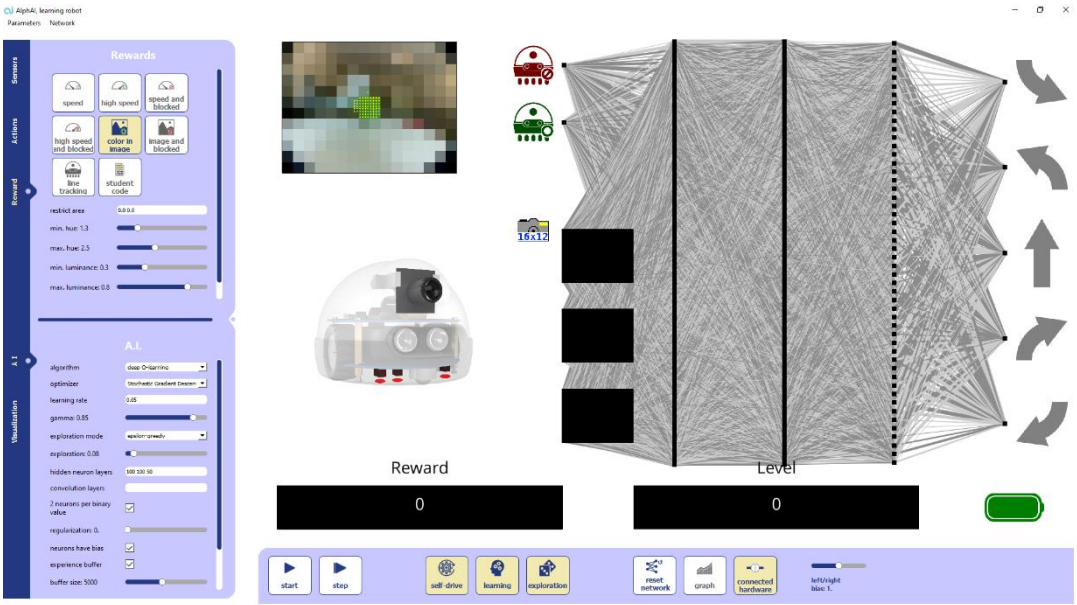
Here is another example of reinforcement learning. By changing the type of reward and placing a new element in the arena, we can succeed in making the robot adopt a completely different behavior.

Load the configuration "followed by a balloon (green)". Or to select the parameters yourself, start the program and then :

- In the AI tab,
  - select "deep Q-learning";
  - in "intermediate neuron layers" put "100 100 50",
  - check "experience memory
- In the Sensors tab, select
  - 

"blocked/moving"
  - 

"camera" (for instance 32x24)
- In the Reward tab :
  - choose "color in the image the robot should not be blocked 



- Place a green balloon in the arena (you can use another color, but you will have to change the color detection settings in the Rewards tab).
- Check that the balloon color setting is correct by placing the balloon in front of the robot; dots should appear on the camera image at the balloon.
- In the reward tab, vary the hue and luminance to match the color of the balloon-Start 

the robot. After a while (about 20 minutes), the robot will start chasing the balloon.

The robot tries to maximize the number of pixels of the color of the balloon on its camera: it will therefore learn to approach the balloon as close as possible, but as it approaches it, it taps into it, so it starts to approach it again and so on.

Notions learned:

In reinforcement learning, to give the robot a specific behavior, the reward must be carefully thought out; so that it rewards the robot only when it performs the desired behavior.

LINE TRACKING

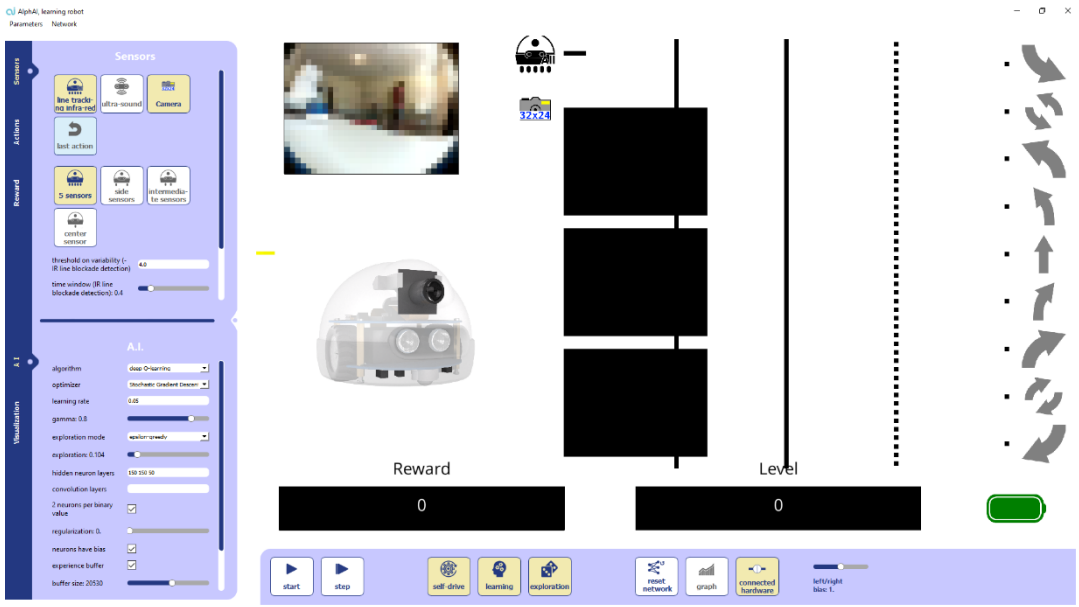
Here is another example of reinforcement learning. By changing the type of reward and placing a new element in the arena, we can succeed in making the robot adopt a completely different behavior.

Load the "line tracking" configuration. Or to select the parameters yourself, launch the program and then :

- In the AI tab,
  - choose "deep Q-learning";
  - in " intermediate neuron layers " put " 150 150 50 ".
  - check "experience memory
- In the Actions tab,
  - put all possible actions
- In the Sensors tab, select
  - 

"camera" (for instance 32x24)
  - 

"IR line tracking sensor - signal from each of the sensors
- In the reward tab :
  - - choose "line tracking". 



- Place black tape in the arena to make a loop (if the tape is not wide, feel free to double or triple the width of the track)
- Start 

the robot. After a long time (about 2-3 hours) the robot will follow the track.

SCENARIO SUMMARY TABLE

File°	Title	Difficulty	Learning Algorithm	Sensors	Concepts	Learning duration	Duration of the animation	Particularities
1	Getting started with the simulated robot	*	reinforcement	Blocked/moving	understand briefly what you see on the screen	2~3 min	1-5 min	simulated robot
2	Getting started with the real robot	*	reinforcement	Blocked/moving	connection with the robot	5 min	1-5 min	set up the arena!
3	Image recognition with camera	*	supervised	camera	Supervised learning (training and testing phases)	immediate	1-5 min	
4	Navigation avec camera	***	supervised	Blocked/moving, camera	importance of the quality of the training data	5~10 min for training, immediate learning	5-10 min	it is necessary to steer well
5	Manual edition	**	No learning	Blocked/moving (ultra-sounds)	connections in the neural network	-	5-10 min	
6	Reinforcement learning Blocked/moving	*	reinforcement	Blocked/moving	Principle of reinforcement learning	2~3 min	5-10 min	The robot finds the same network as the one previously made by hand
7	Reinforcement learning with camera	*	reinforcement	Blocked/moving, camera	Using learning in a more complex case	10~15 min	10-15 min	

8	Intruder invasion	**	supervised	ultra-sounds	Importance of the intermediate neuron layers	2~3 min	5-10 min	
9	Balloon tracking	*	reinforcement	Blocked/moving, camera	Importance of the chosen reward for reinforcement learning	15~20 min		The robot may push the balloon out of the arena, it will have to be watched
10	Line tracking	*	reinforcement	Blocked/moving, camera, ground line tracking sensor	Importance of the chosen reward for reinforcement learning	2~3 h		It is better to launch the robot at the beginning of the session and do the rest with another robot while waiting for it to train