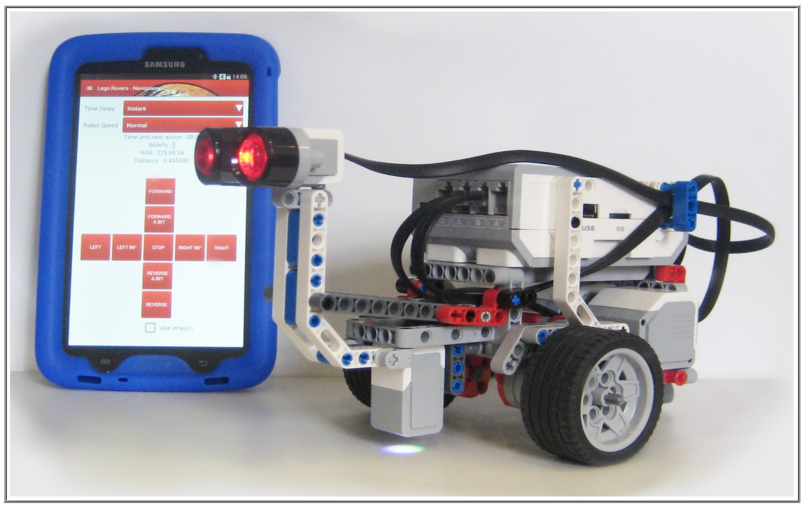
Welcome to Lego Rovers

AIM: To control a Lego robot!

HOW?: Both by hand and using a computer program.



In doing so you will explore issues in the programming of planetary rovers and understand how roboticists work to overcome these.

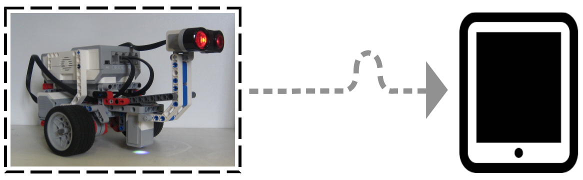
First things to do

Firstly, you will need the following:

1. Install Lego Rovers on your Android Device and on your Robot.

2. Build your robot.

3. Pair your robot with your Android Device and connect to it for Internet Usage.



Website The Lego Rovers website is at: http://legorovers.csc.liv.ac.uk.

Instructions

There are **instructions for setting up your robot** and Android Device Lego Rovers Website in the Downloads section and in the Parent/Teacher handbook (also available from the Downloads section of the Lego Rovers website).

Build instructions for your robot are also available from the Downloads section of the Lego Rovers website.

For latter parts of this activity, you will need:

3. White, Black and Blue pieces of card.

4. The Line Following mat and Path To Water mat. These can be printed out from the Downloads section of the Lego Rovers website.

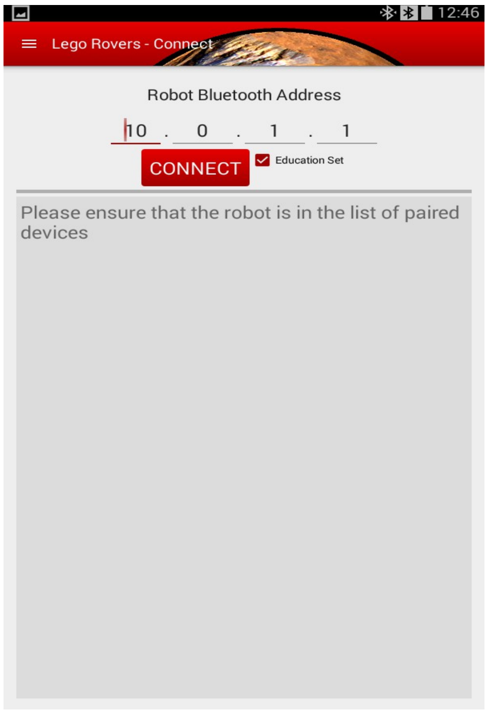
Running the Lego Rovers App

1. Start the Lego Rovers application on your Android Device.

The first screen you see should be the *Connect Screen*. You may need to get rid of the Android keyboard by tapping the back button on your device.

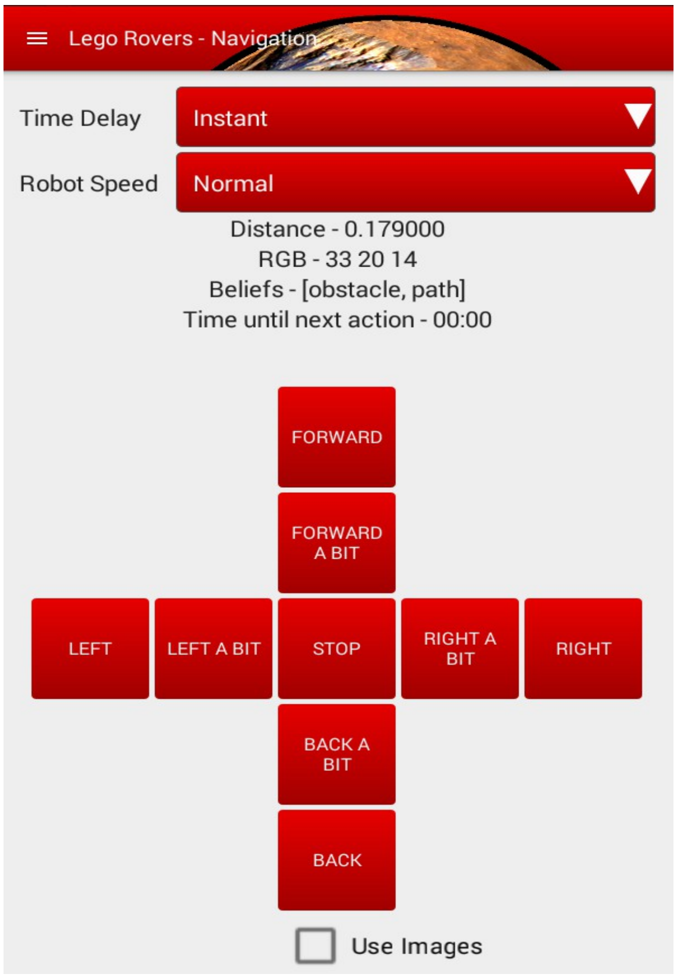
These instructions are for use with the EV3 Education Set so make sure that the box marked "**Education Set**" is checked.

Then press **CONNECT**.



Please ensure that the robot is in the list of paired devices

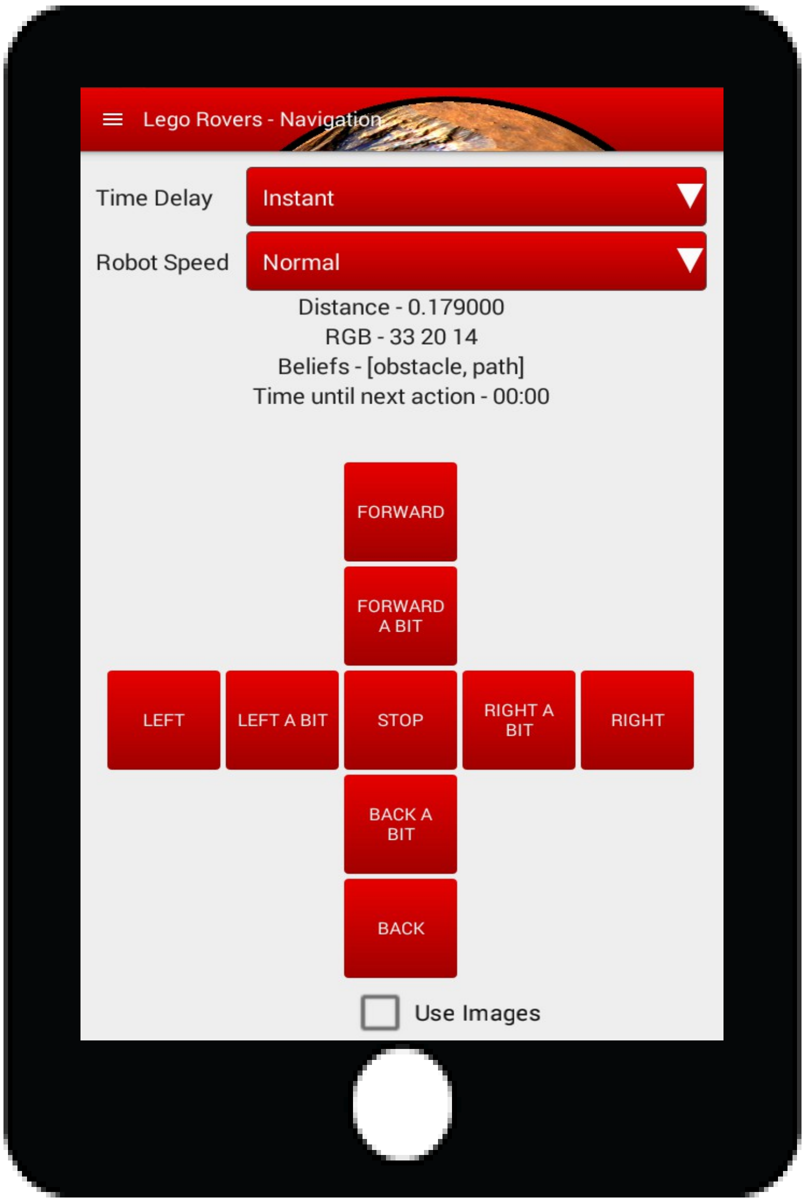
Now you will need to use the Menu for the Lego Rovers app to change to the **Navigation Panel**.



Time Delay: Instant
Robot Speed: Normal
Distance - 0.179000
RGB - 33 20 14
Beliefs - [obstacle, path]
Time until next action - 00:00

Operating a Lego Rover by Hand

The **Navigation Panel** lets you remotely control the robot from the tablet.


Lego Rovers - Navigation

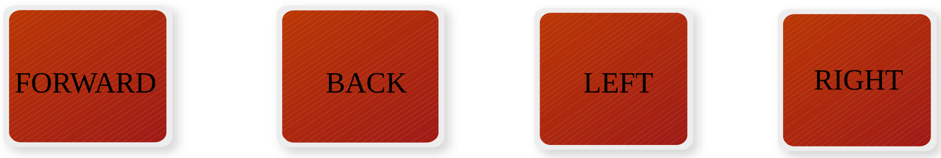
Time Delay: Instant

Robot Speed: Normal

Distance - 0.179000
RGB - 33 20 14
Beliefs - [obstacle, path]
Time until next action - 00:00

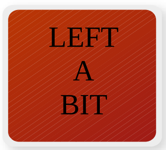
There are two types of control:

1. Controls like FORWARD, BACK, LEFT and RIGHT:



move the robot *until you press stop* or *give it another command*.

2. Controls like FORWARD A BIT and LEFT A BIT:



make the robot move *a set distance* (approx 30 degrees for left a bit and right a bit) and *it can't be interrupted*.

press **LEFT A BIT**

press **STOP**

What do you see?

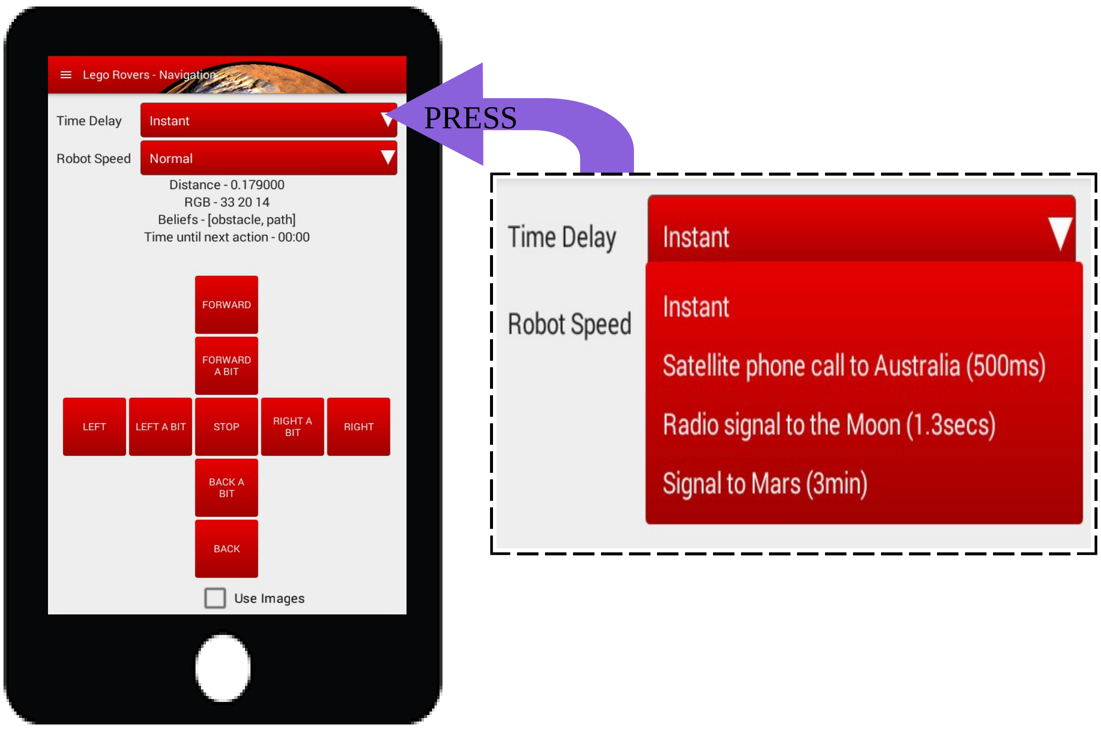
The robot doesn't respond to the stop signal until it has turned through 30 degrees.

Experiment now on changing the speed the robot moves. As the robot gets faster, it becomes harder to control.

- Set up some basic tasks, such as driving around an obstacle.
- Find out how fast you can get the robot to move while still being able to control it accurately.

Working a robot like this is called <u>teleoperation</u> which means operating something from a distance. Lots of robots today are controlled like this.

Operating Robots on Other Planets



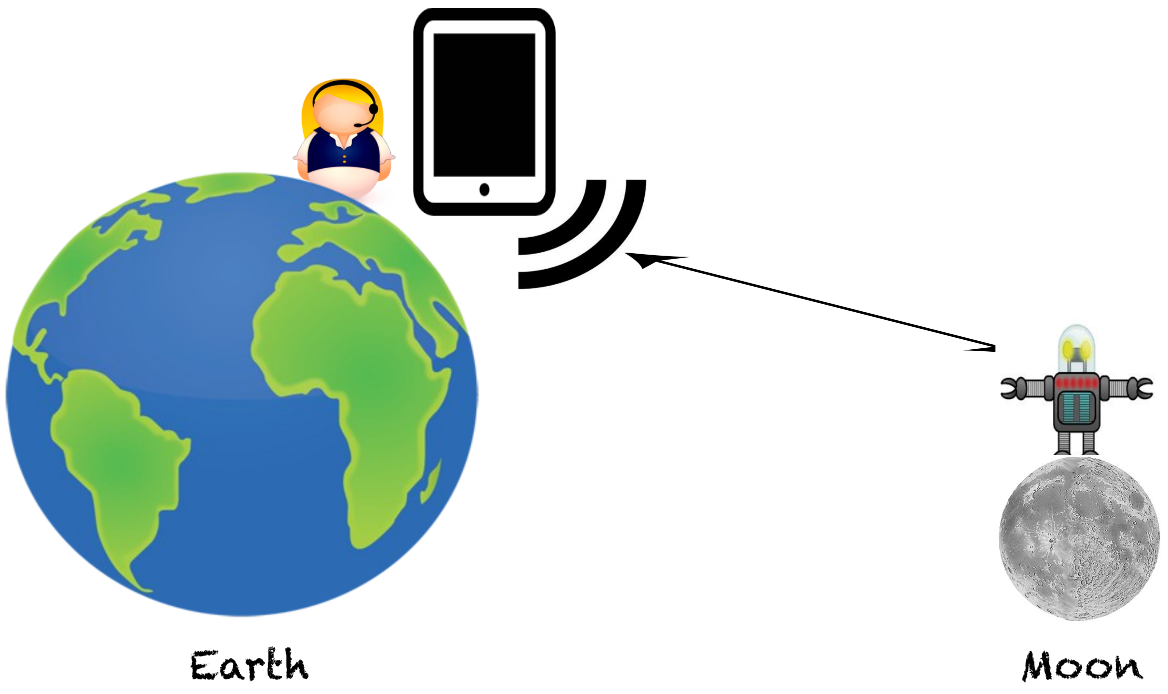
Lego Rovers - Navigation

Time Delay: Instant

Robot Speed: Normal

Distance - 0.179000
RGB - 33 20 14
Beliefs - [obstacle, path]
Time until next action - 00:00

The major problem space agencies face when controlling a planetary rover (that is robots on the Moon or other planets) is the time taken for radio signals to travel over large distances.



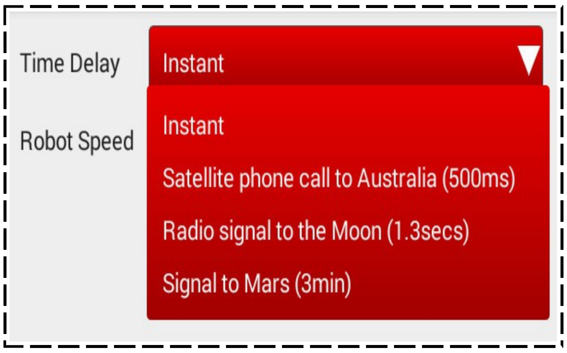
On your control panel there is a **Delay** setting. You have a choice of *four* delays:

1) Instant

There is no delay.

2) 500ms (half a second)

This is the time it takes for a satellite phone call to reach Australia.


Time Delay
Instant
Robot Speed
Instant
Satellite phone call to Australia (500ms)
Radio signal to the Moon (1.3secs)
Signal to Mars (3min)

3) 1.3s

This is the time it takes for a radio signal to reach the Moon.

4) 180s (three minutes)

This is the shortest possible time it can take a radio signal to travel between the Earth and Mars assuming that their orbits are currently bringing the two planets close together. In practice it normally takes longer for a signal to get from scientists at NASA to Mars Curiosity.

Give it a try!

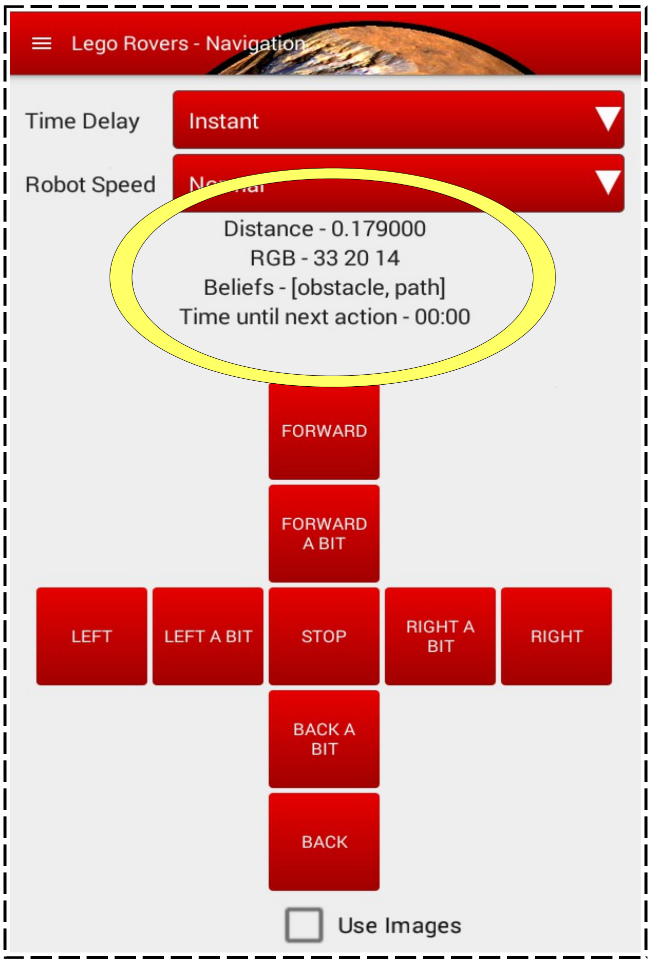
Experiment on the tasks you devised for testing speed (routes with obstacles) with the first three time delays (Instant, Satellite Call, Earth to Moon).

- Can you still do these with a time delay?
- How much slower must the robot be before you can accurately achieve these tasks with the delay?

Project: **Investigate communication between Earth and Mars.**

Why can the time vary? What's the longest time it might take for a message to reach a planetary rover on Mars?

Sensors and Beliefs


Lego Rovers - Navigation

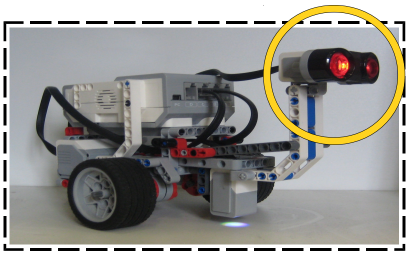
Time Delay: Instant

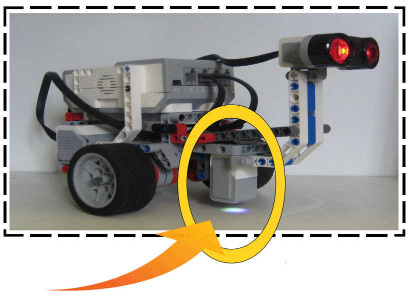
Robot Speed: Normal

Distance - 0.179000
RGB - 33 20 14
Beliefs - [obstacle, path]
Time until next action - 00:00

The Lego Rover has **two sensors** attached:

1. The **ultrasonic** sensor at the front uses sound in order to decide how far away any obstacles may be. If it hears no echo, then it reports that any obstacle is at ``infinity".



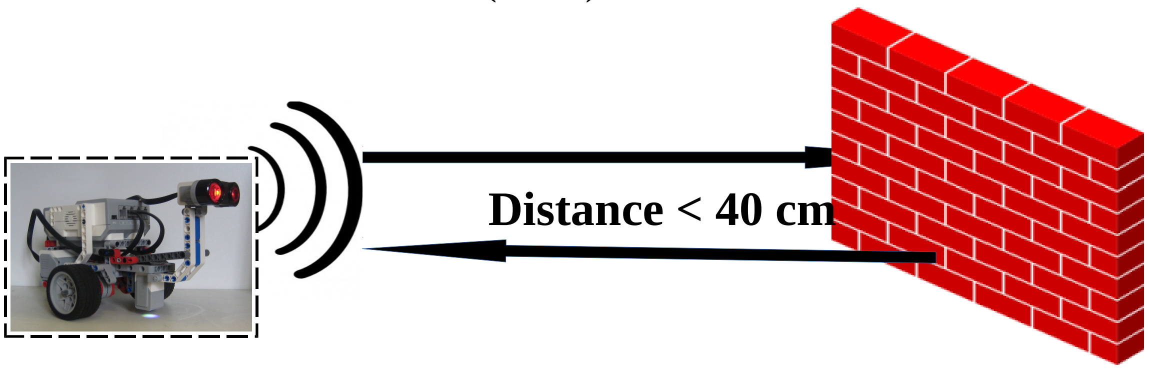


2. The **light sensor** shines a light downwards and measures the light reflected back and gives values for how much **Red**, **Green** and **Blue** there is in the light. This can be used to detect the colour of the ground beneath it.

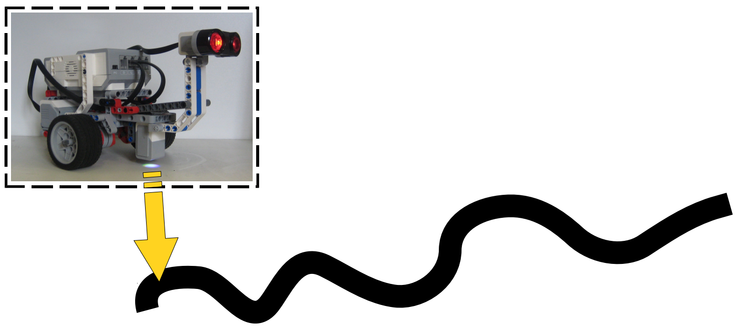
In order to use values from sensors to control the robot, we convert them into **“beliefs”**. The possible beliefs the robot can have are **OBSTACLE**, **PATH** and **WATER**. Beliefs appear whenever sensor values cross some threshold.

Examples:

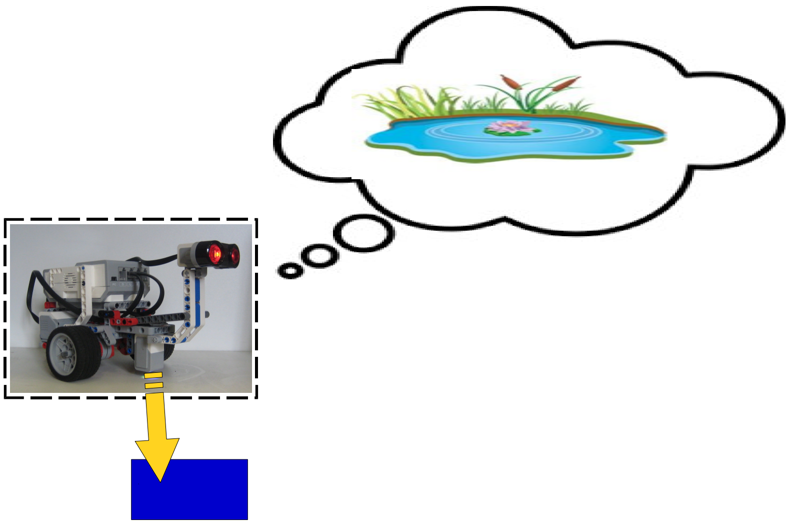
- The robot believes <u>there is an *obstacle*</u> whenever the ultrasonic sensor returns a value below 0.4 (40cm).



- It believes <u>there is a *path*</u> whenever it detects a black surface beneath.



- It believes <u>there is *water*</u> whenever it detects a blue surface beneath.



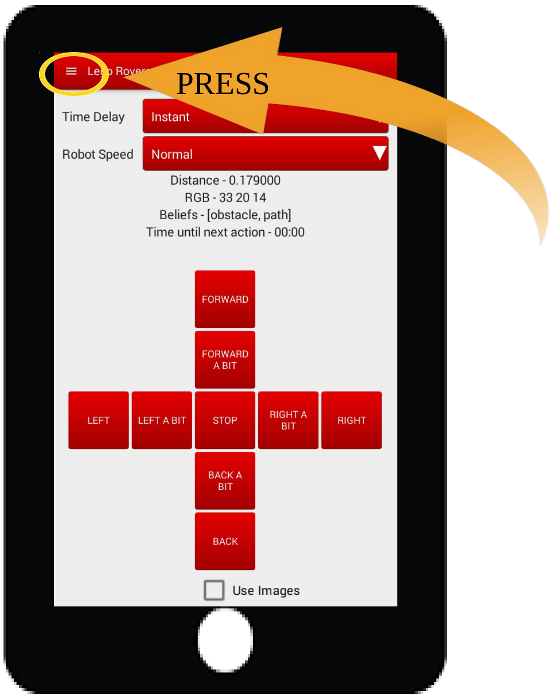
Sometimes, especially in dark rooms, the light sensor *doesn't work* very well. If it isn't detecting blue and black surfaces properly it may be possible to change some settings to help. This is explained in the parent/teacher guide in the section on troubleshooting.

Give it a try !

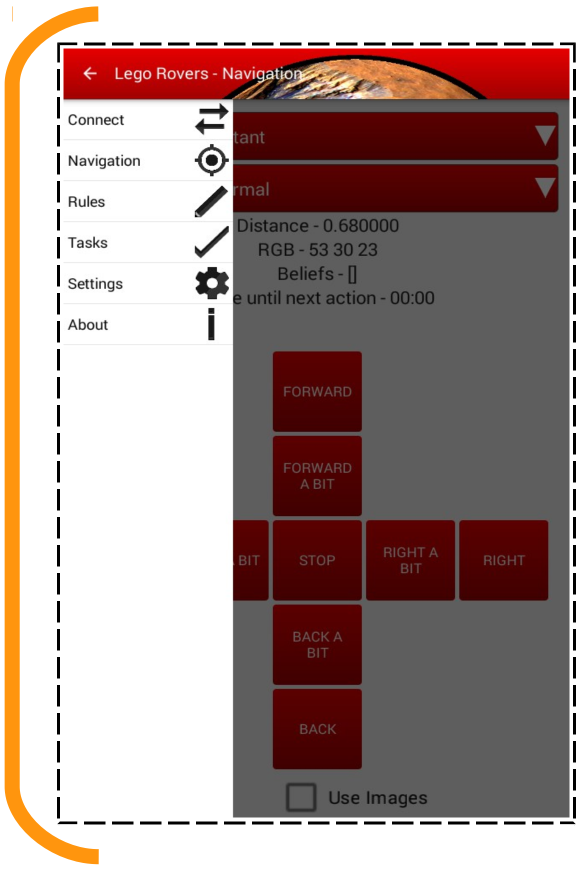
Experiment with the sensors to see if your robot can correctly detect obstacles, white, blue and black surfaces.

Programming with Beliefs and Rules

In the **Menu** for the Lego Rovers app

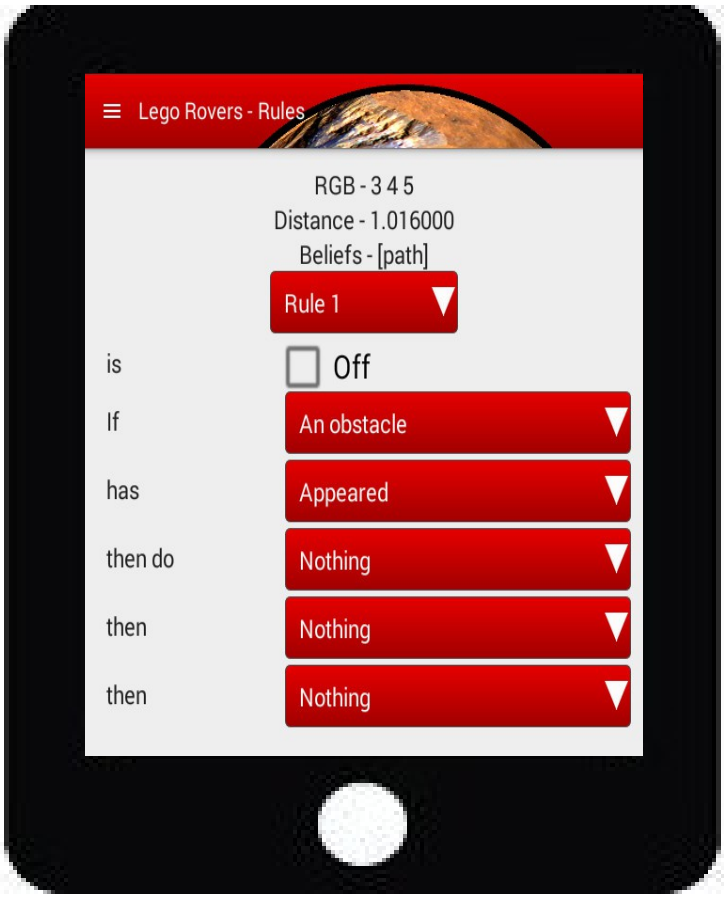


Time Delay Instant
Robot Speed Normal
Distance - 0.179000
RGB - 33 20 14
Beliefs - [obstacle, path]
Time until next action - 00:00



tant
rmal
Distance - 0.680000
RGB - 53 30 23
Beliefs - []
e until next action - 00:00

you will find a **Rules Panel:**


Lego Rovers - Rules

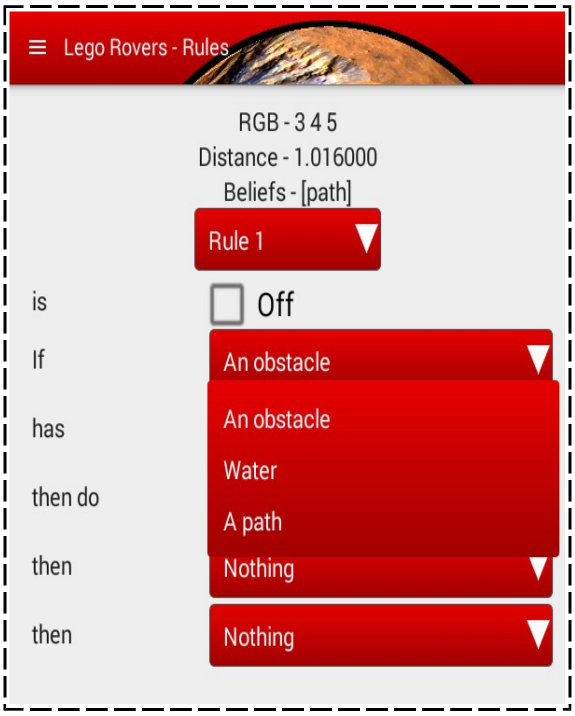
RGB - 3 4 5
Distance - 1.016000
Beliefs - [path]

Rule 1

is	Off
If	An obstacle
has	Appeared
then do	Nothing
then	Nothing
then	Nothing

We can use the robot's beliefs to control the robot using **rules**. There are **three steps** involved in creating a *rule:*

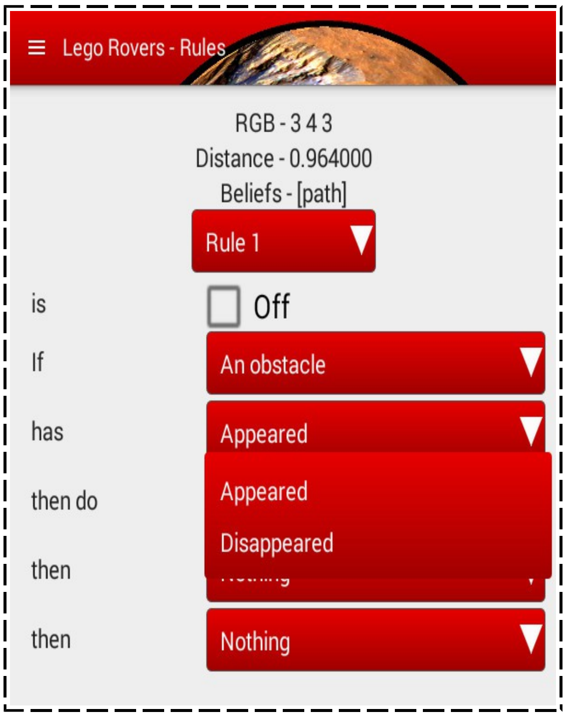
1. First you need to decide which **event** the rule should react to. Events are the **appearance** or **disappearance** of a **belief**.


Lego Rovers - Rules

RGB - 3 4 5
Distance - 1.016000
Beliefs - [path]

Rule 1

is	Off
If	An obstacle
has	An obstacle / Water / A path
then do	
then	Nothing
then	Nothing

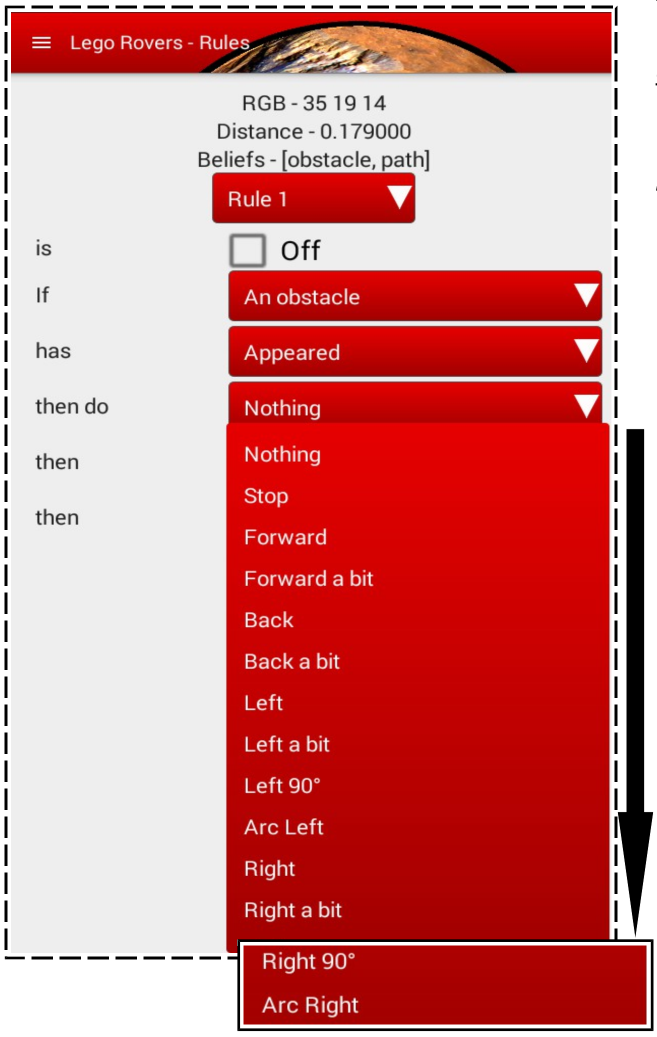

Lego Rovers - Rules

RGB - 3 4 3
Distance - 0.964000
Beliefs - [path]

Rule 1

is	Off
If	An obstacle
has	Appeared
then do	Appeared / Disappeared
then	
then	Nothing

2. Then you need to decide what the robot should do in reaction to that event. You can select up the three **actions** for the robot to do in order but we will start by *only selecting one*.

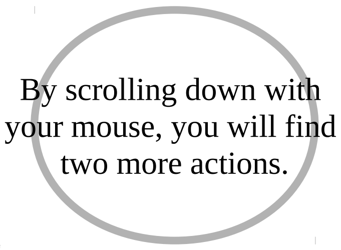

Lego Rovers - Rules

RGB - 35 19 14
Distance - 0.179000
Beliefs - [obstacle, path]

Rule 1

is	Off
If	An obstacle
has	Appeared
then do	Nothing
then	
then	

Nothing
Stop
Forward
Forward a bit
Back
Back a bit
Left
Left a bit
Left 90°
Arc Left
Right
Right a bit
Right 90°
Arc Right


By scrolling down with your mouse, you will find two more actions.

What each action means:

[Nothing] The robot will do nothing.

[Stop] The robot will stop whatever else it is doing.

[Forward] The robot will move forward until instructed to do something else (e.g. go backwards or stop).

[Forward a bit] The robot will move forward a short distance.

[Back] The robot will move backward until instructed to do something else.

[Back a bit] The robot will move backwards a short distance.

[Left] The robot will turn left on the spot until instructed to do something else.

[Left a bit] The robot will turn left for 30 degrees.

[Left 90] The robot will turn left 90 degrees.

[Arc Left] The robot will turn left while moving forward slightly until instructed to do something else.

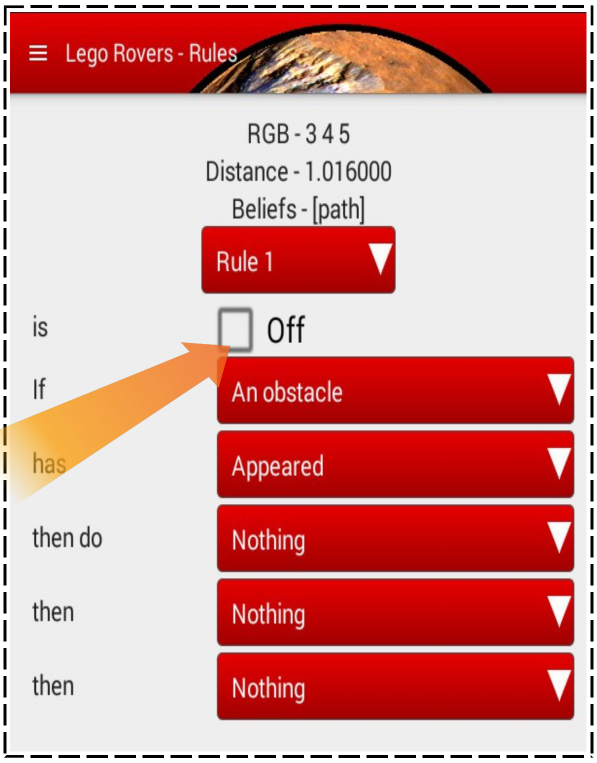
[Right] The robot will turn right on the spot until instructed to do something else.

[Right a bit] The robot will turn right for 30 degrees.

[Right 90] The robot will turn right 90 degrees.

[Arc Right] The robot will turn right while moving forward slightly until instructed to do something else.

3. Lastly the rule must be made **active** by **checking the box**. Although you may have several active rules for an event, **only the first** will be used.

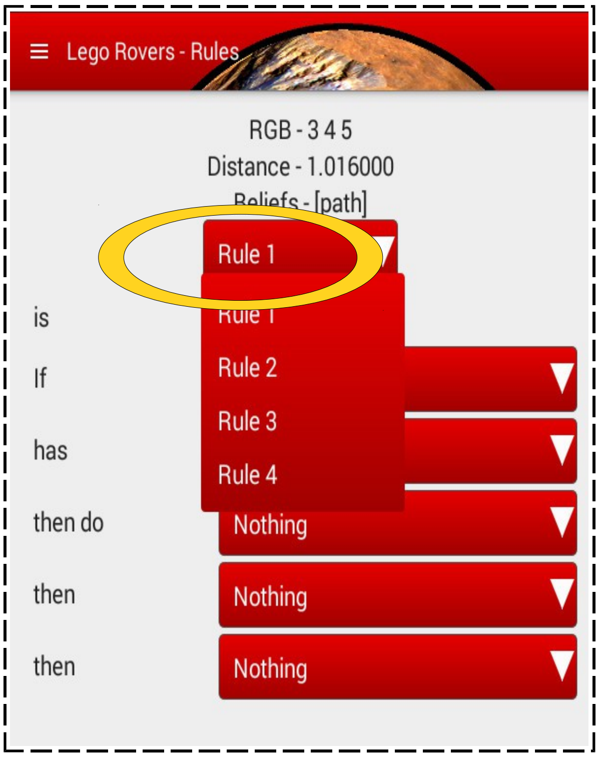


Lego Rovers - Rules

RGB - 3 4 5
Distance - 1.016000
Beliefs - [path]

Rule 1

is	Off
If	An obstacle
has	Appeared
then do	Nothing
then	Nothing
then	Nothing



Lego Rovers - Rules

RGB - 3 4 5
Distance - 1.016000
Beliefs - [path]

Rule 1

Set up **Rule 1** to make the robot move <u>backwards</u> when <u>an obstacle appears</u>. Set all the other actions to "<u>nothing</u>".

Set up **Rule 2** to make the robot move <u>forwards</u> when <u>an obstacle disappears</u>. Set all the other actions to "<u>nothing</u>".

- What happens when both rules are active at once?

When the robot acts according to a *Rule* we say that the rule has **fired**. So when an obstacle appears we say that Rule 1 fires and causes the robot to move backwards. It then carries on moving backwards until either Rule 2 fires or you switch back to the Navigation Panel and take control of the robot yourself.

Rules with more than one Action

Let's deactivate Rule 2, so you just have one rule for what to do when an obstacle appears.

We will now make the robot do three things in order when an obstacle appears. To do this it is important to remember the difference between:

- *actions that continue until instructed to do something else*
- *actions that move the robot a fixed distance or do a fixed turn.*

Example: Tell the robot to do the following

1. Back

then **2. Right 90**

then **3. Stop**

when **an obstacle appears.**

⚠ You will not see the robot move back at all, because that action is immediately interrupted by the "right 90" action.

You need to make sure that an action which carries on until given a different instruction is the last action to appear in any rule.

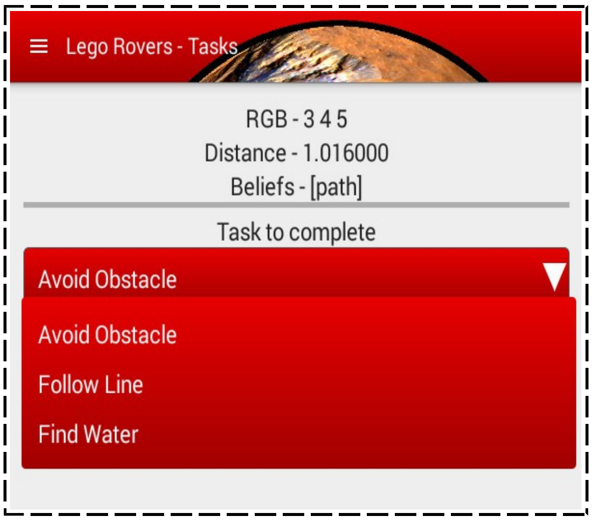
Give it a try!

Figuring out what is happening when Rules are active: You can use the readouts for beliefs and sensors at the top of the **Rules Panel** to help you understand when a Rule is firing.

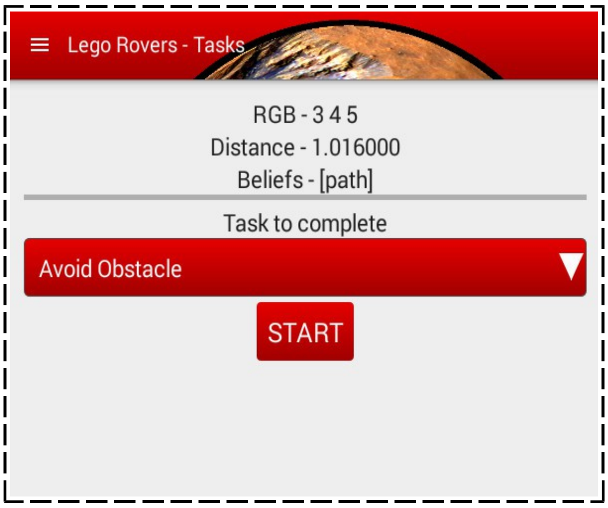
- Experiment with Rules that have more than one action in order.
- Can you make your robot turn around 180 degrees and move off in the opposite direction when it detects an obstacle?

Tasks

You will find a **Tasks Panel** in the menu for the Lego Rovers app.



RGB - 3 4 5
Distance - 1.016000
Beliefs - [path]
Task to complete



RGB - 3 4 5
Distance - 1.016000
Beliefs - [path]
Task to complete

The Tasks panel contains some programs we have put together to demonstrate things the robot can do. Make sure this is set on "**Avoid Obstacle**" and press **Start**. The robot should now move around your space on its own avoiding obstacles.

Can you program the avoid obstacle task using rules you have created yourself?

Paths and Water

In this section you will learn how the robot detects either a Path or Water.

Go back to the **Tasks Panel** and select "**Follow Line**".

Place your robot on the **Line Following** mat.

- What happens?
- Can you work out how to write this program yourself using the rules?

Hint: *You will need the Arc Right and Arc Left actions.*

The line following algorithm is a very famous algorithm from Control Engineering which is all about how to create and control machines that react to input from sensors. It can be very finely tuned so it hardly looks like the robot is turning at all.

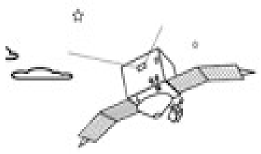
Go back to the **Tasks Panel** and select "**Find Water**".

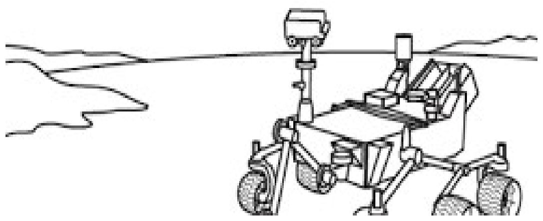
Place your robot on the **Find Water** mat.

- What happens this time?
- Can you work out how to write the Find Water program yourself using rules and assuming the robot starts on the line?

Projects

In this section, we have some projects for you.





Project 1: **Research planetary rovers.**

- Which was the first robot sent into space to investigate the Moon or another planet?
- What robots are out there at the moment?

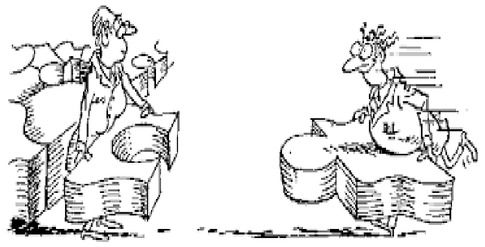
If you succesfully completed this project and want to go even further, we next have an advanced project for you.

Project 2: **Research Line Following robots**

- How can you make their behaviour smoother?
- Does it help if you add more sensors?

Have you completed the advanced project and want more? Here is the final project for experts!

The Final Project:



- Why can't you reproduce the Find Water task in your own program if the robot starts a long way from the path?
- Is it just that you don't have enough rules or would you have needed something else? If so what?

Have you succesfully completed all the above projects? Then, congratulations!

You can now consider yourself as an

expert in controlling a Lego Rover!