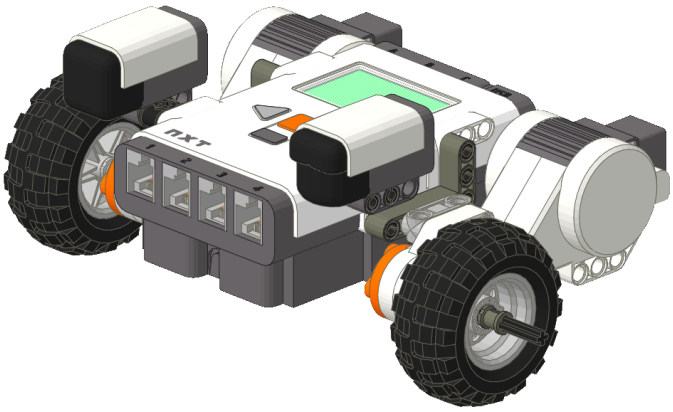
Getting Started with RCJA Soccer

Often we find that many teachers are unsure just how to get started with the Soccer Challenge. This page has been created to give teachers and students a head start on their way to creating fun and competitive soccer robots.

Be aware that the design and programs here are not perfect. How can you change them to make them better?

Building Instructions

The at the bottom of the page will give you step-by-step instructions to build your own soccer-bot. The design is too big to fit the rules for the competition, and it is a little wobbly in place, so you'll have to eventually make modifications to make your robot better.



This design can be made from a single 9797 LEGO Mindstorms kit, with the addition of a Compass Sensor (to find which way you are pointing) and a IRSeeker Sensor (to find the ball).

Programming Instructions

This guide will focus on the NXT-G environment, but the ideas can easily be moved to other languages such as RoboLab and RobotC.

There are two fundamental parts to getting a soccer robot going

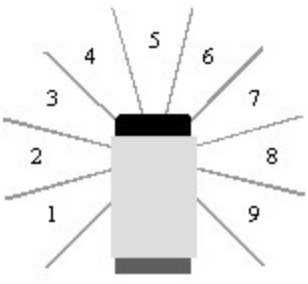
Let's find the ball

Let's kick it in the right direction (don't want to score an own goal)

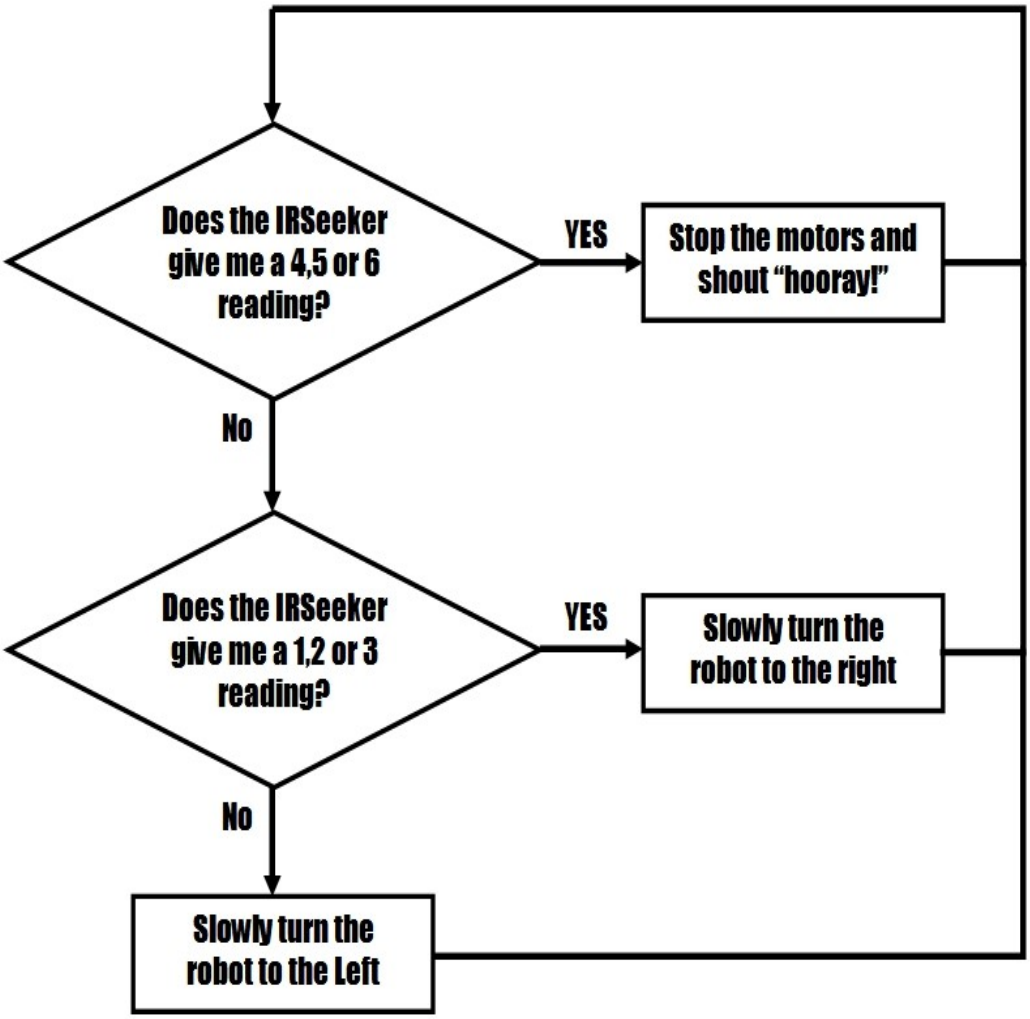
Part 1: Let's Find the Ball!

We will use the IRSeeker to locate the ball. If you haven't already done so, download the NXT-G block from the HiTechnic Website. The IRseeker consists of 5 individual sensors, inside the curved face, that give us readings of where the ball is located.

Depending on its location, the IRSeeker will output a number based on the diagram below.



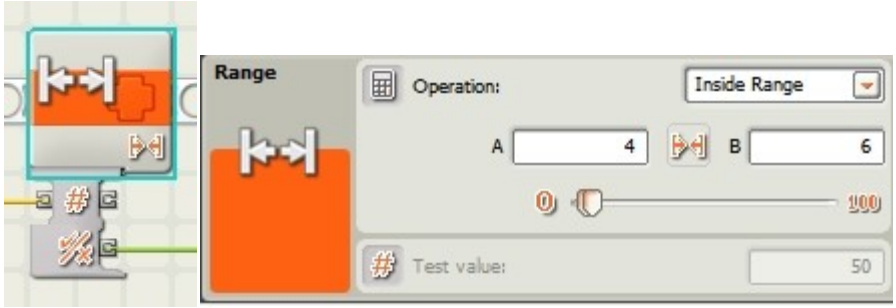
So a reading of '5' indicates the ball is directly in front, whereas a '1' and '9' indicate extreme left and right locations respectively. Before we use this information, we're going to quickly do up a little flow chart to figure out our program flow.



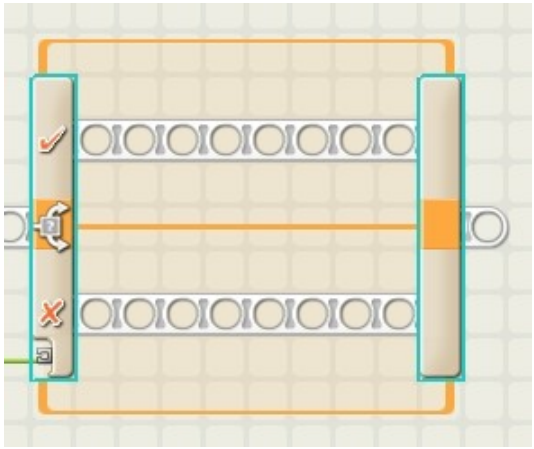
You'll notice that we specified 'seeing the ball' as either 4,5 or 6. This gives us a little leeway to find the ball and it doesn't have to be perfectly straight for the robot to stop. Extension - Can you make a better version, perhaps turning quick if the robot is further away, and slowing down as it closer to the centre?

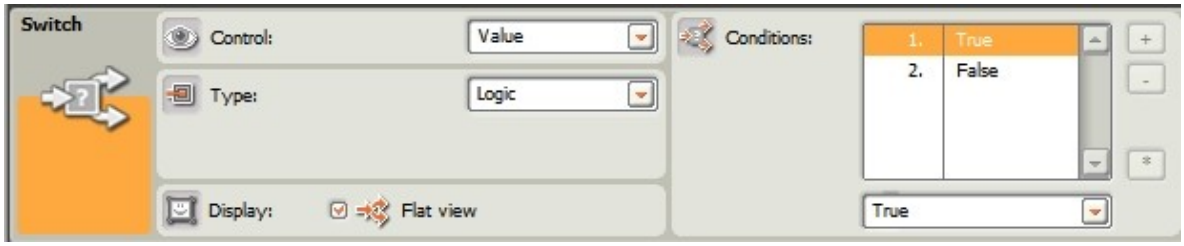
So how do we now put this into NXT-G programming? We are going to utilise 2 different blocks, the 'Range' and 'Switch(logic)' blocks.

The 'Range' block allows us to feed in a sensor and decide if it is within a certain range. If it is in the range, it will output a 'True' signal, otherwise it will give a 'False' signal. In this case we are determining whether the reading is between 4 and 6.

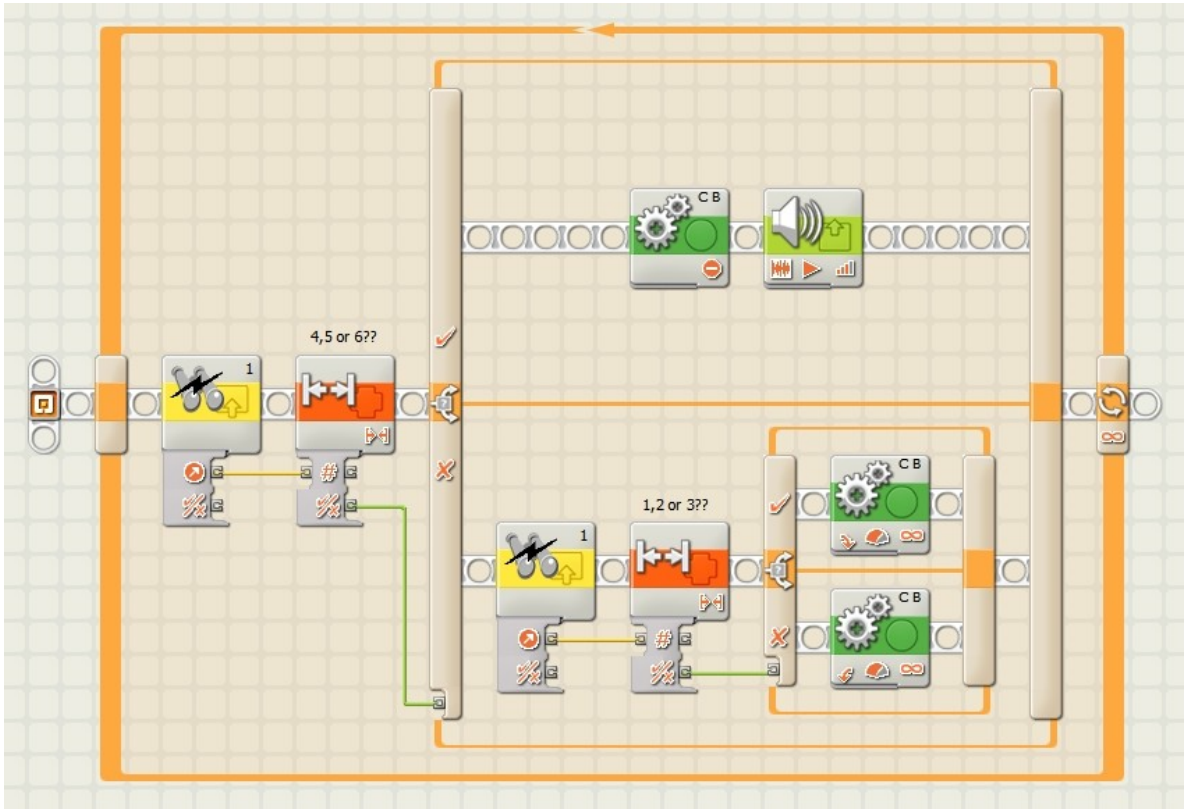


The Switch(logic) block takes in a value, either True or False, and decides what action to do based on that input. If it receives a 'True' it will process the top line, if it receives 'False' it will process the bottom line.





We can combine these two blocks with a IRSeeker Sensor reading block to fill out our flowchart. Don't forget to put it into a loop so that it continually checks the sensors. Also remember to set your turning to be 'unlimited'. We want the robot to keep turning until it finds the ball, not just a predefined amount.



Extension Activities:

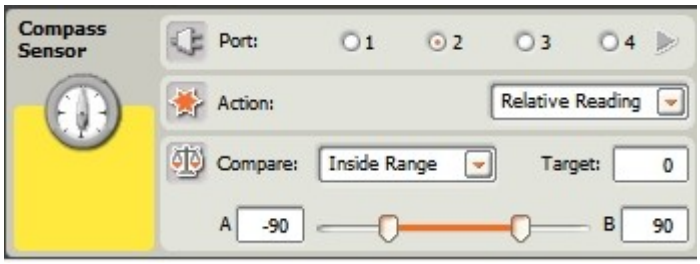
- Instead of saying 'hooray!', make the robot drive forward a short distance to kick the ball
- What happens if the robot cannot see the ball at all? (Hint, the IRSeeker will give a reading of '0')

Part 2: Let's Kick it in the Right Direction

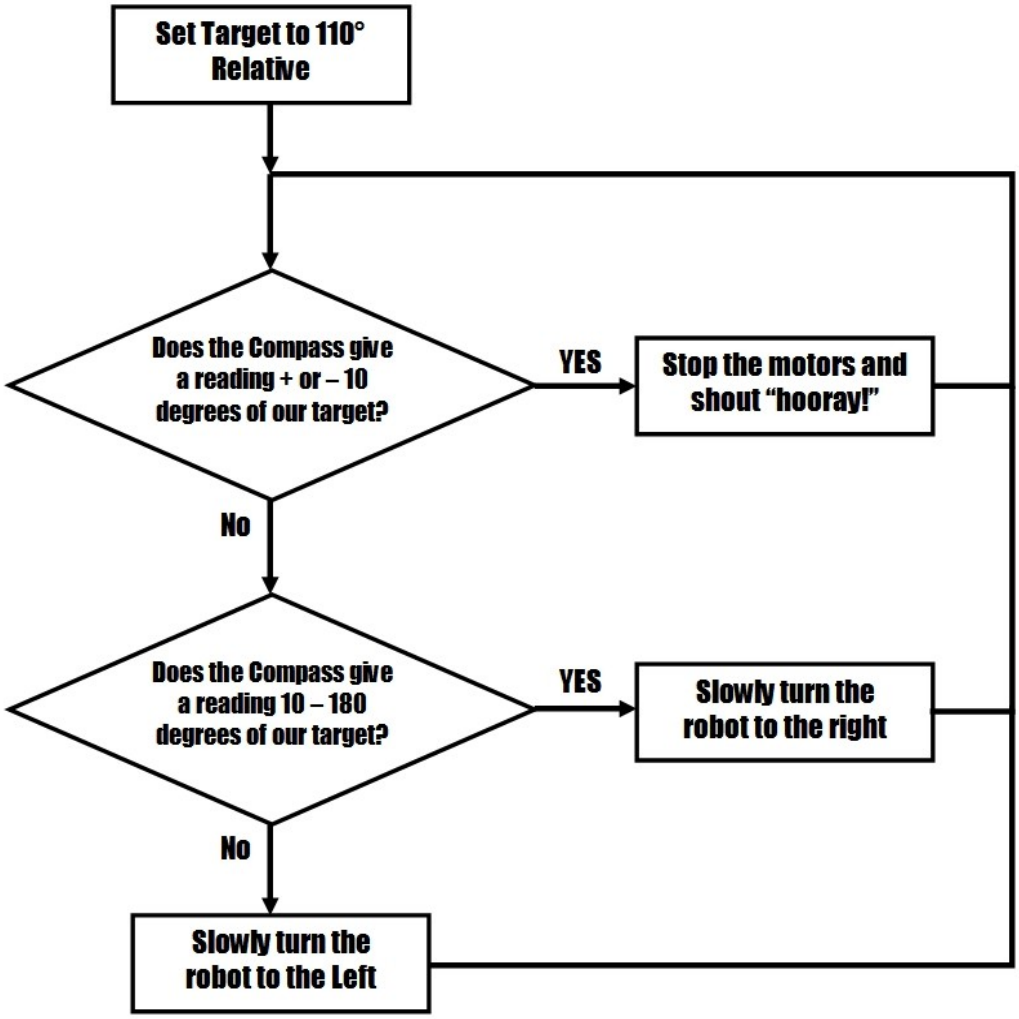
This part is a little more tricky and we will start by breaking down the problem into two parts.

- Am I facing the right direction?
- When I see the ball, am I facing in the right direction?

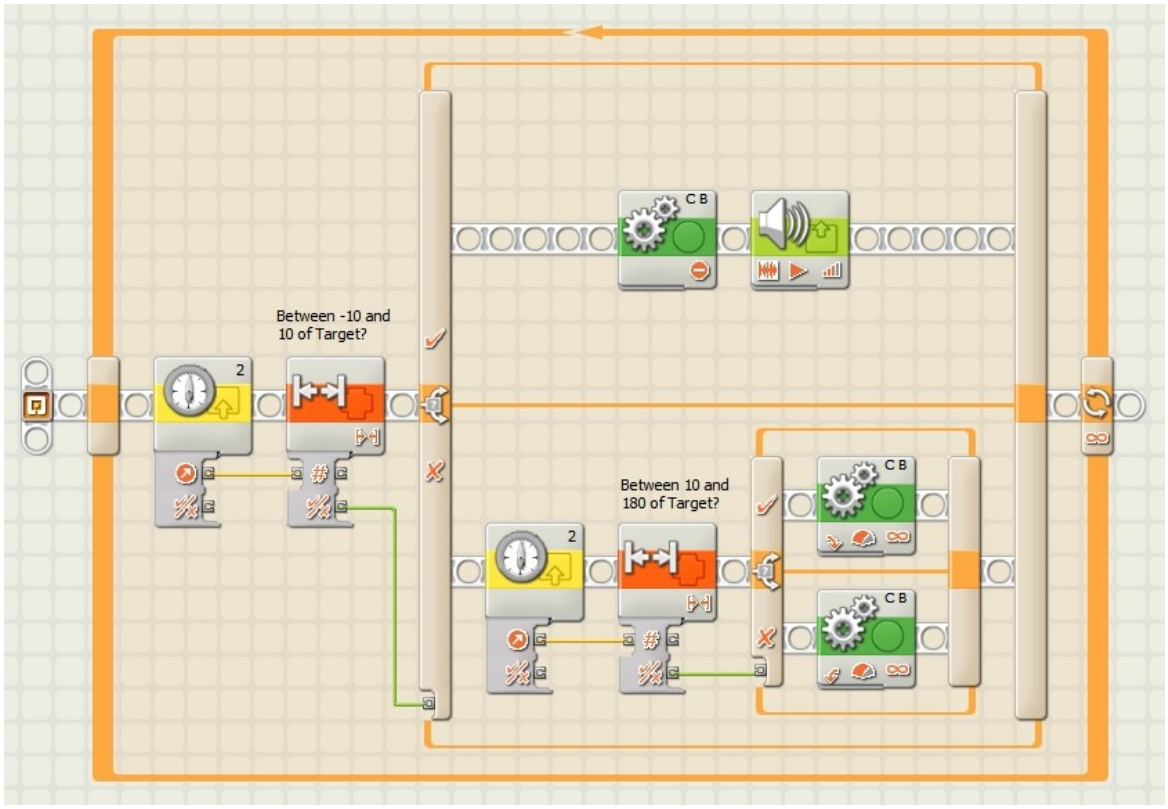
The first part is very similar to finding the ball. We are just going to get the robot to slowly rotate until it sees 'the soccer goal'. Firstly figure out which way our goal is located. For this example I'll assuming the goal is located at 110 degrees from north. Luckily the Compass Sensor block has a 'relative' mode, which allows us to input in a target value (110 in this case) and it will return values relative to this value. On the configuration panel you can ignore the 'Compare' section. We wont be needing it for this program.



So now let us look at a flow chart of how we are going to get our robot to point in the right direction.

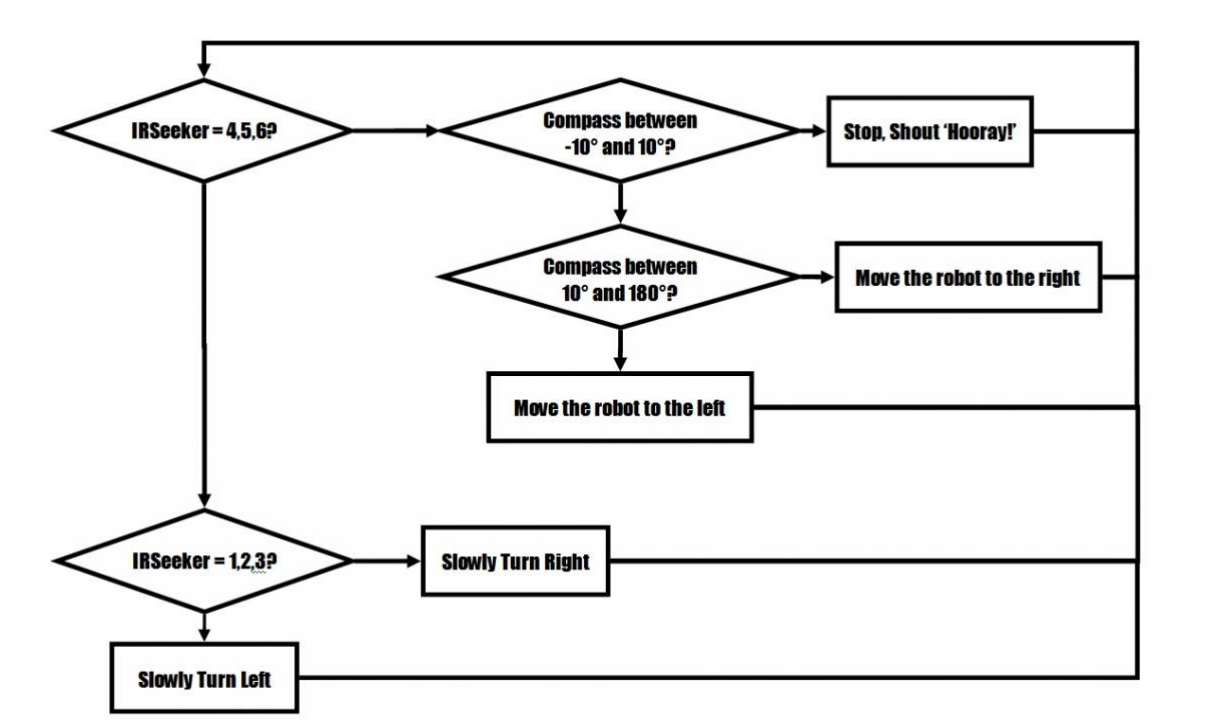


The Program looks very similar to the IRSeeker version.

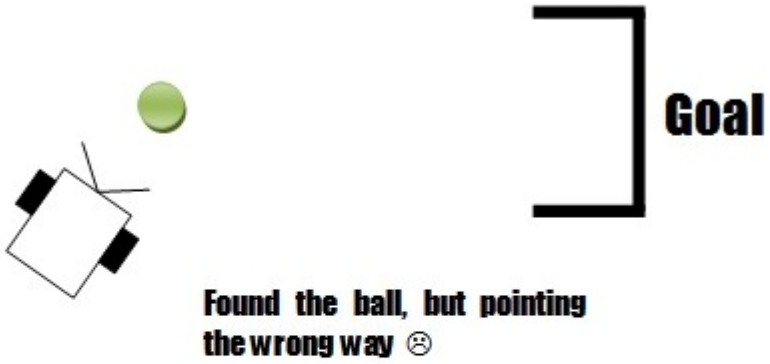


So how do we combine this with the IRSeeker now to answer our next question - "When I see the ball, am I facing in the right direction?"

Let's go back to our flowchart.



What does it mean "Move the robot to the right" and "Move the robot to the left"? Sometimes your robot will find the ball, but on checking the compass sensor will find that it is not pointing the right direction.



Found the ball, but pointing the wrong way ☹

Need to move to the robot to the left and search for the ball again

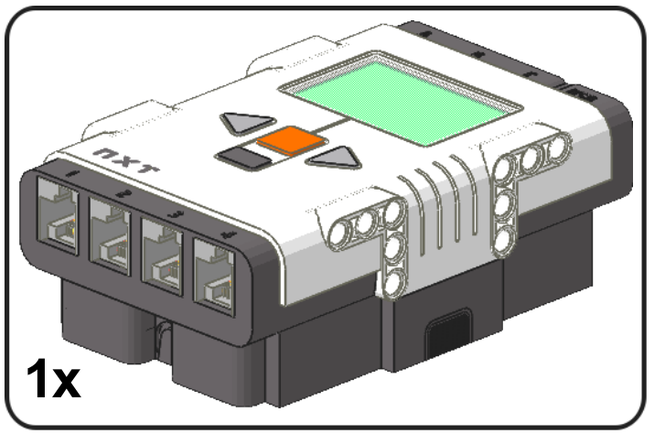
Found the ball, and pointing the right way ☺

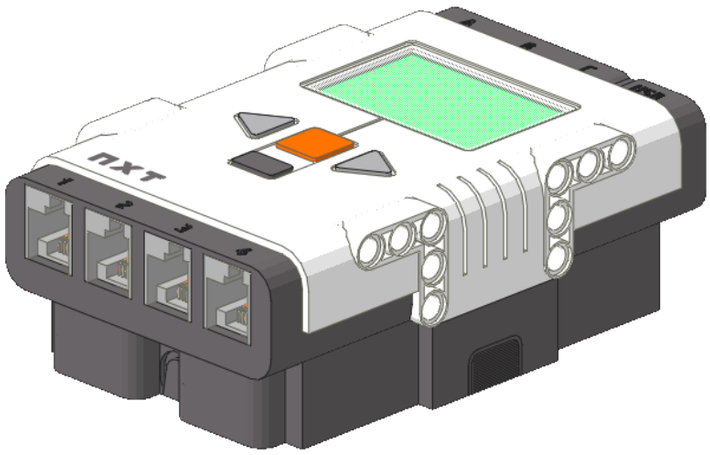
How do we turn all this into an NXT-G program? Look at the flowchart, and try and create your program similar to this. Keep making small improvements and don't try and change too much at one time.

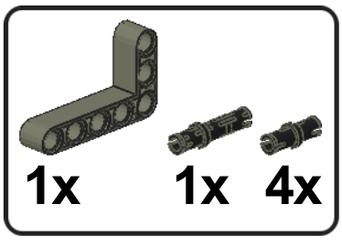
We will leave you to figure that out. We can't reveal all our secrets! :)

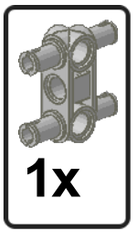
Other things to think about once you have this working.

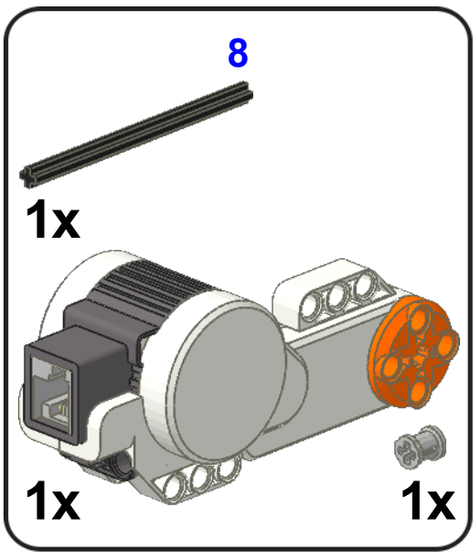
- How do we keep the robot out of the white area? (Hint: You'll need a light sensor)
- How can we make the robots 'kick' rather than 'push' the ball?
- How can we make the robots stronger and fit inside the size limitations?

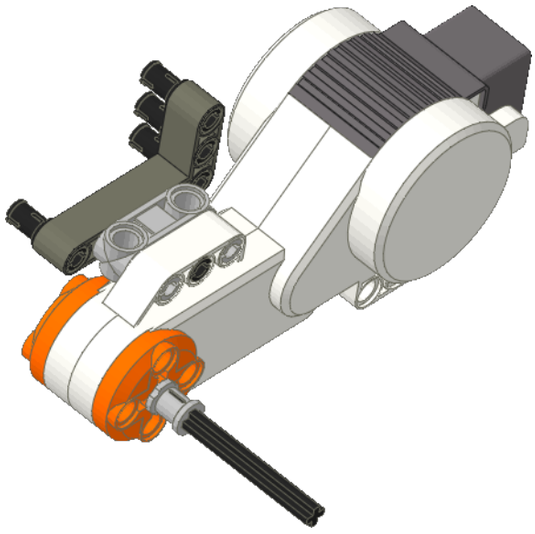

nxt
1x

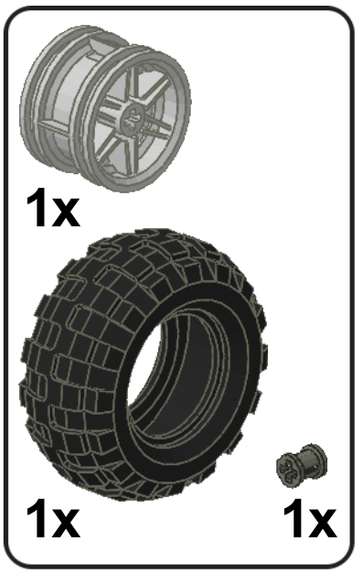

nxt

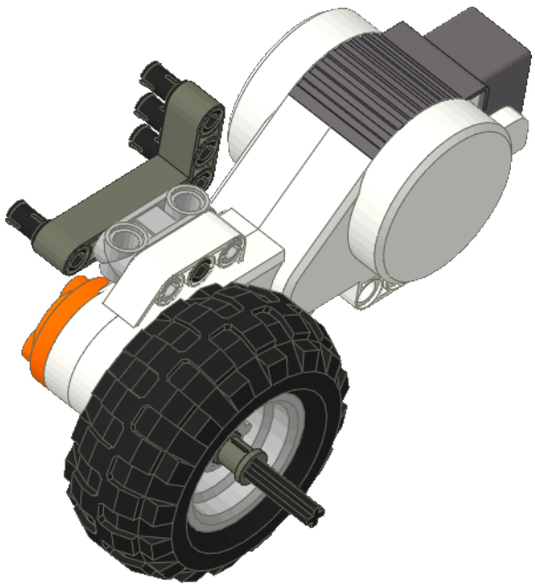

1x
1x
4x

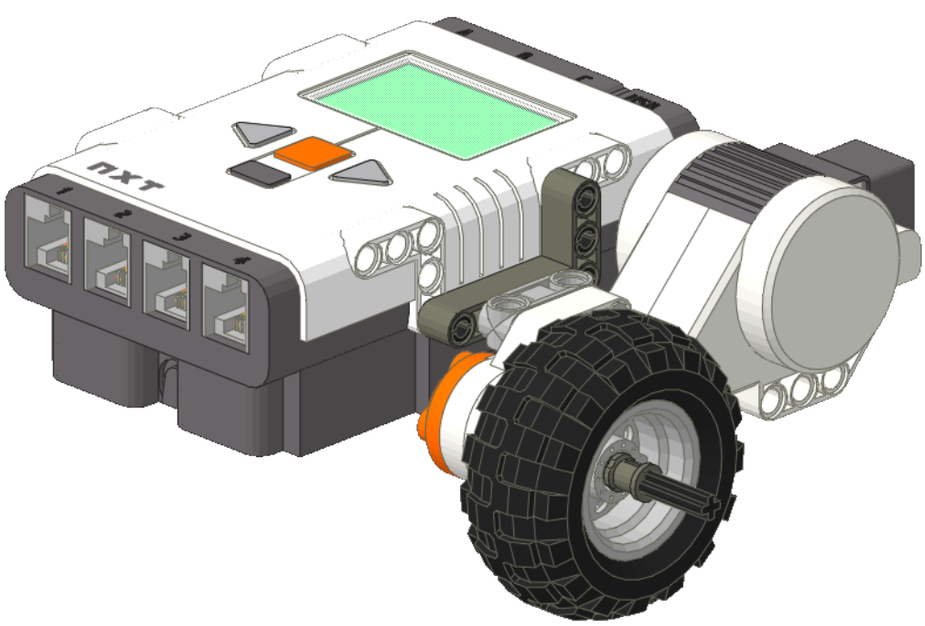

1x

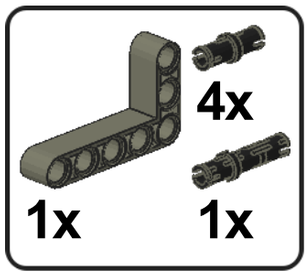

8
1x
1x
1x

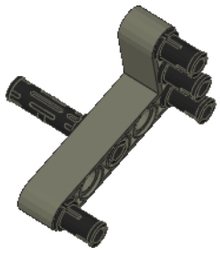


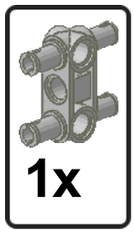

1x
1x
1x

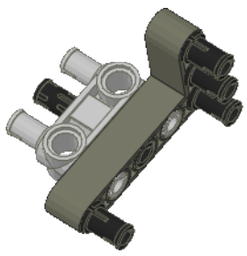


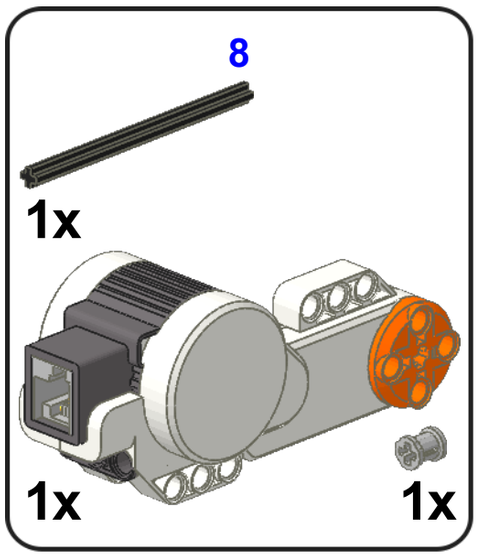

NXT
1
2
3
4

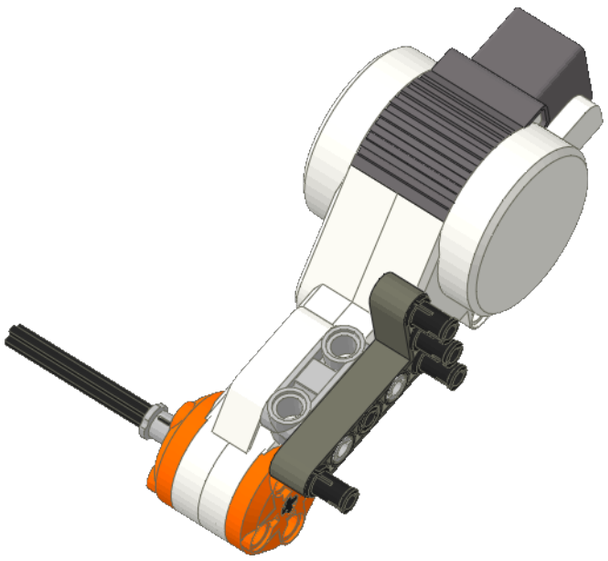

4x
1x
1x

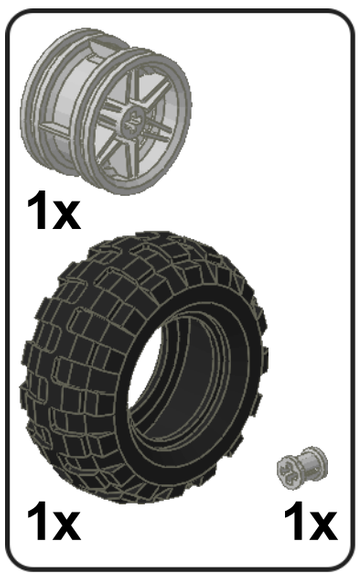


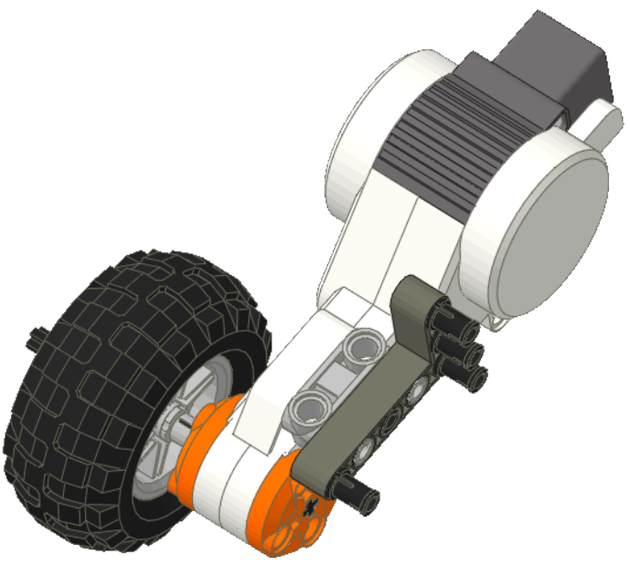

1x

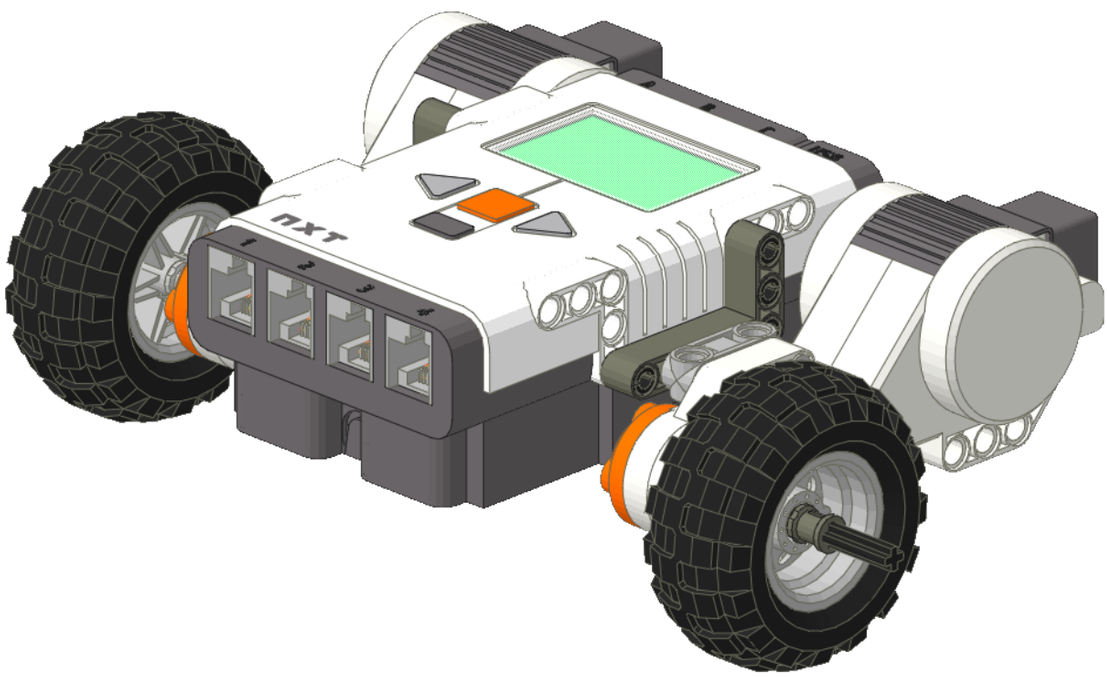


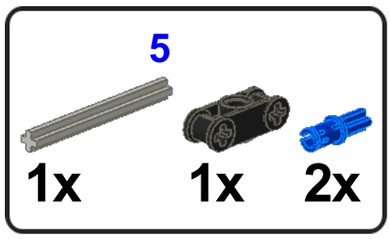

8
1x
1x
1x

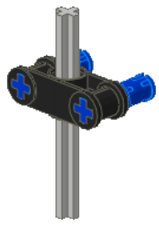


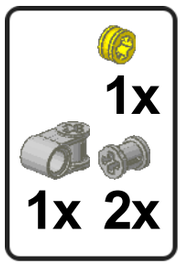

1x
1x
1x

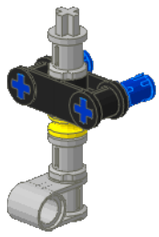


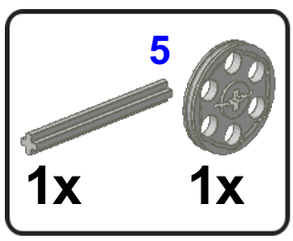

nxt

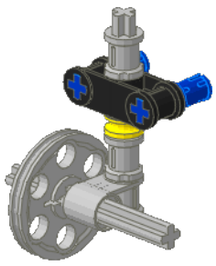

5
1x
1x
2x



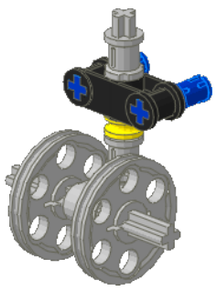

1x
1x
2x

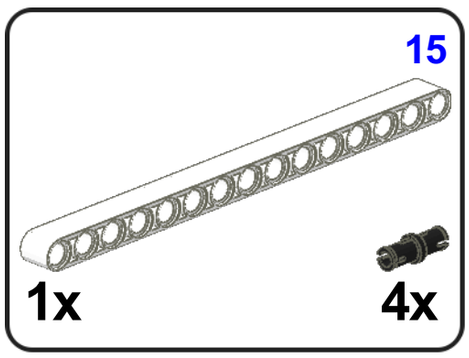


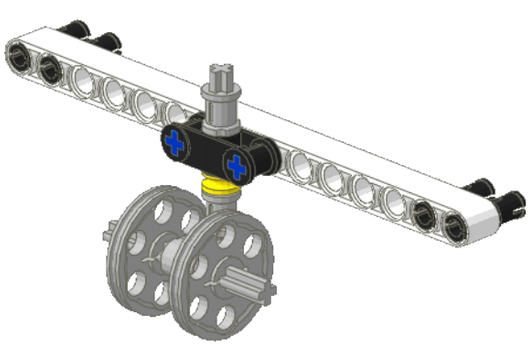

5
1x
1x

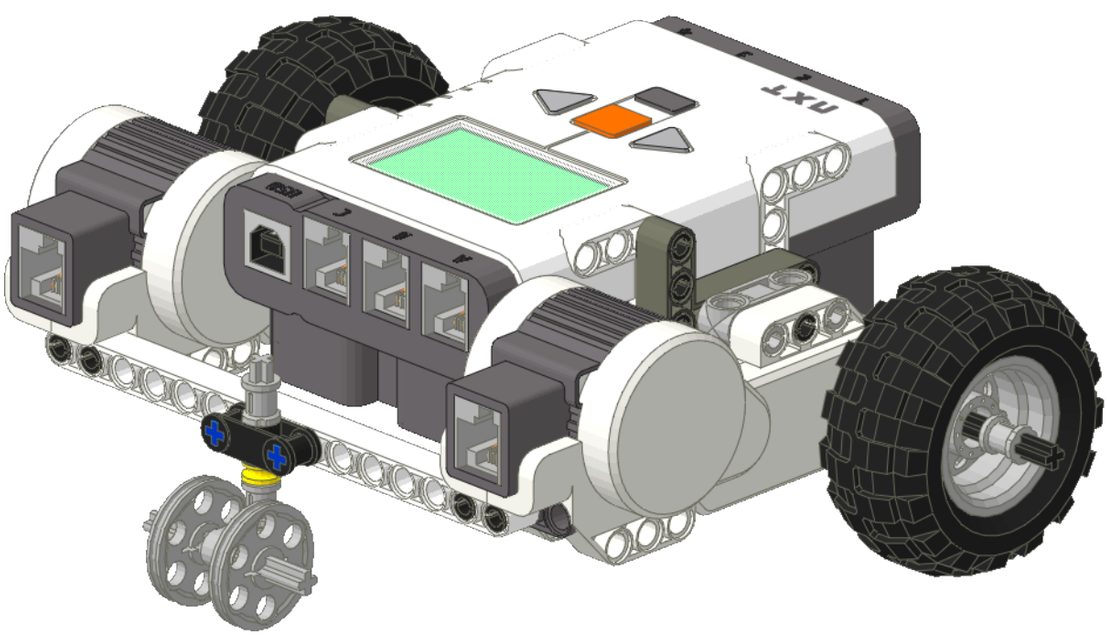



1x

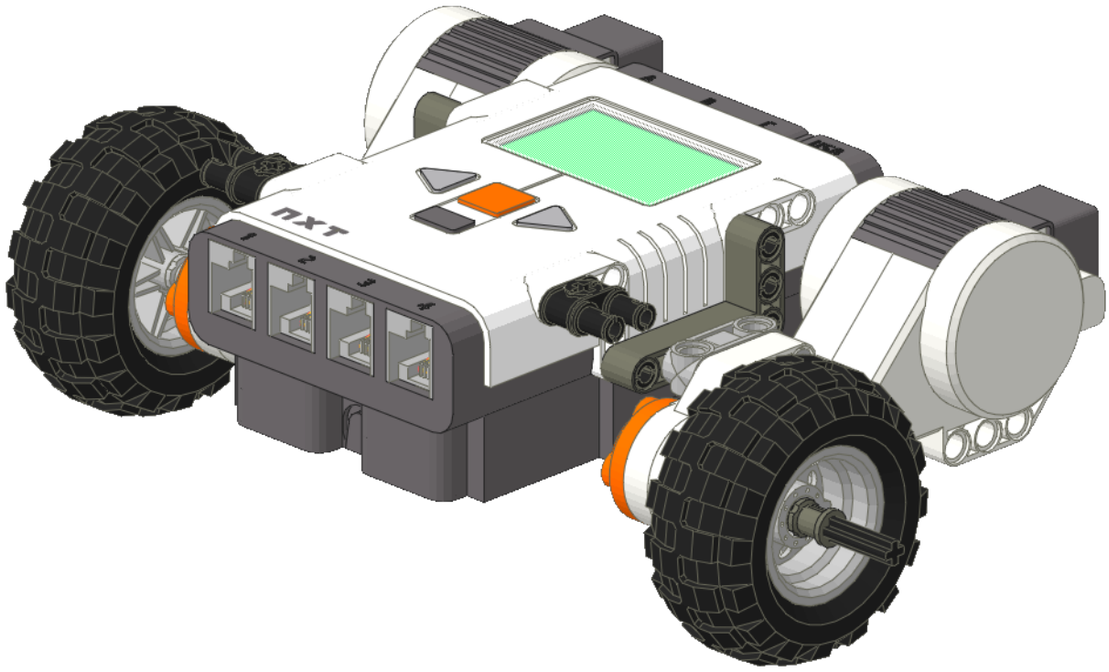


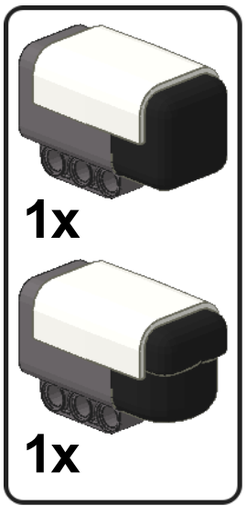

15
1x
4x




NXT


2x




1x
1x