The TOW3R Story

ROBOTREMIX#3

Overview

What is Towers Of Hanoi?

Back when I was young, the puzzle game “Tower of Hanoi” was made from wood – one block with three vertical rods and (up to) eight discs of different diameter. Although the rules are simple, at that time it took me a while to figure out how to solve it. Here is how it works:
Goal: Move the entire stack to another rod, obeying three rules:

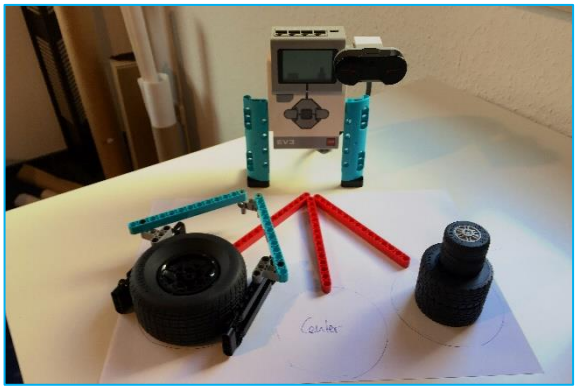
1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

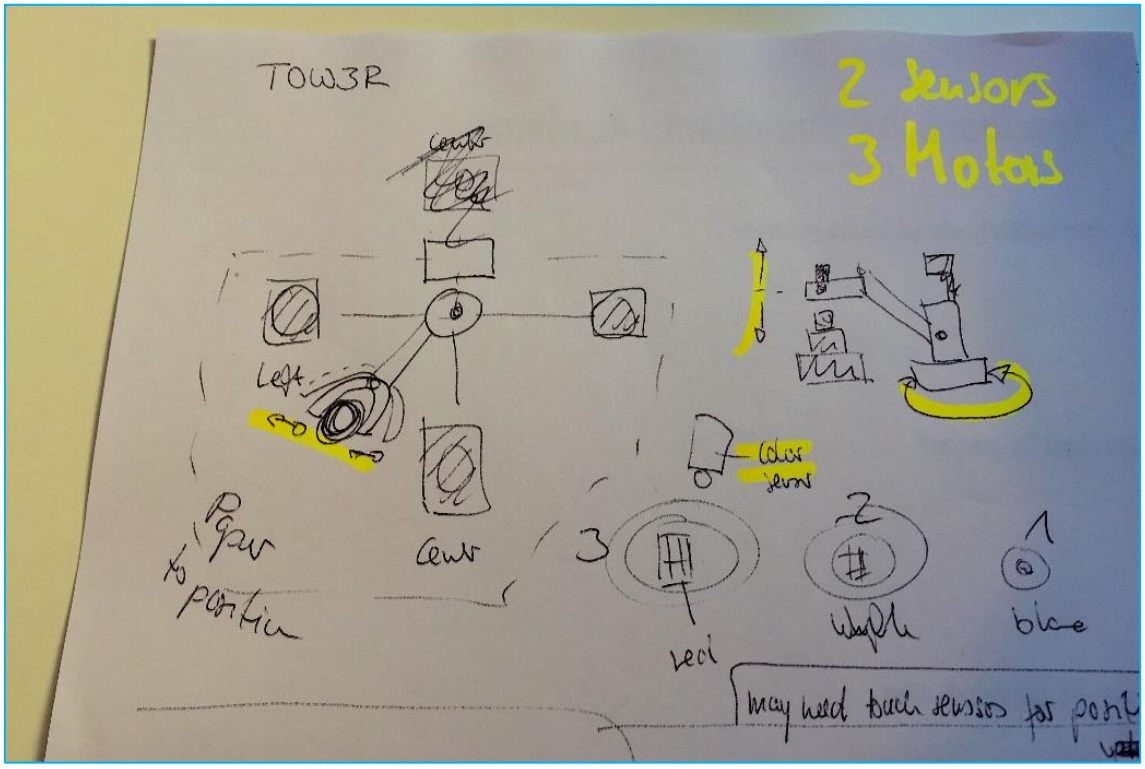
(image from http://www.wooden-toy-store.com/Tower%20of%20Hanoi.asp)

At university, the same thing came back to be in an algorithm class. Regardless of the total number of discs the puzzle can be solved with the same algorithm by using recursion. In short, if you know how to move a stack of *two* to an empty pin, just do that. Than move the third disk and finally move the stack of two again, this time on top of the third disk. Now you know how to move a stack of *three*! Got the idea? Good! For more details check the article on Wikipedia.

Why build it with LEGO MINDSTORMS EV3?

When I saw the huge tires of the LEGO Technic Drag Racer set 42050, I immediately thought about stacking them to become “Tower of Hanoi”. Although there are only three different tire sizes when mixing the Drag Racer set with LEGO MINDSTORMS set 31313, it would still show the fundamental idea of the recursive algorithm. Moreover, I liked the idea to build a robot that moves real objects around.





I proposed “TOW3R” for the remix challenge and became one of the four selected. The first build was fast and led to one big learning: The programming would not be an issue, but getting the mechanical elements to be precise enough was. In fact, precision was the key challenge.

However, let me introduce the concepts first, before I discuss the challenges I faced.

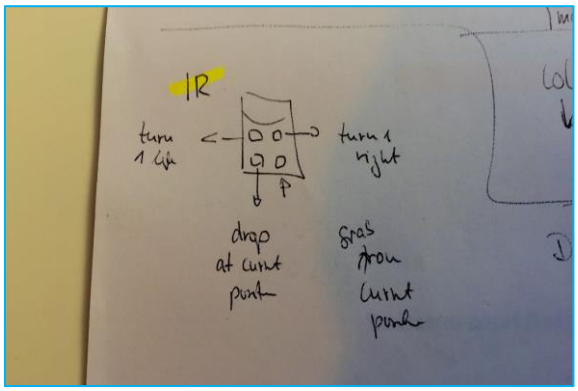
Concepts

Play modes

There are two ways to play. You either steer the claw or lean back and watch TOW3R perform the minimum number of moves.

In both cases, TOW3R randomly selects one of the three positions to start and another as the target. The claw moves to the starting position and you need to position the stack of tires into the open claw. Then the fun begins.

TOW3R is controlled via the IR remote. Using the buttons at the bottom you can grab the tire on the top of the stack or drop a tire on top of a stack respectively. Using the buttons on the top you can move the arm left or right.

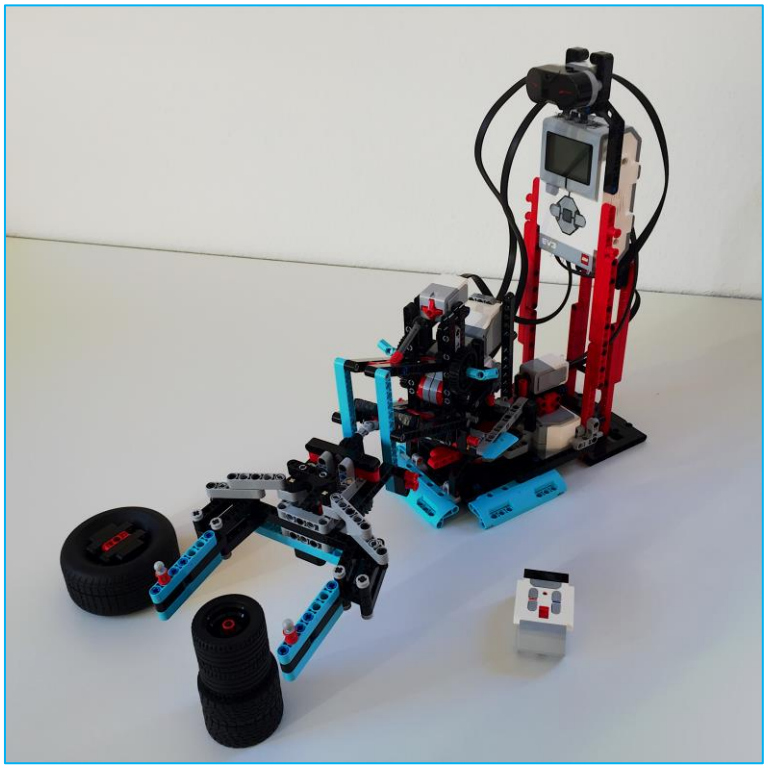


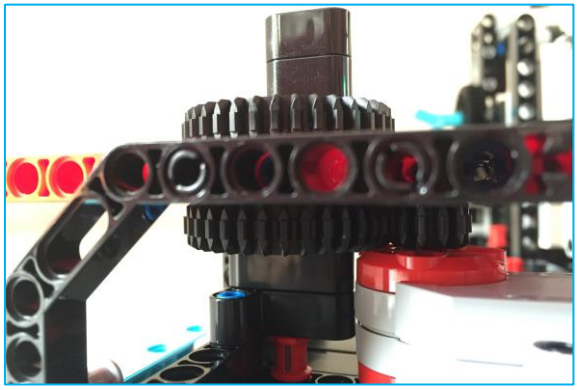
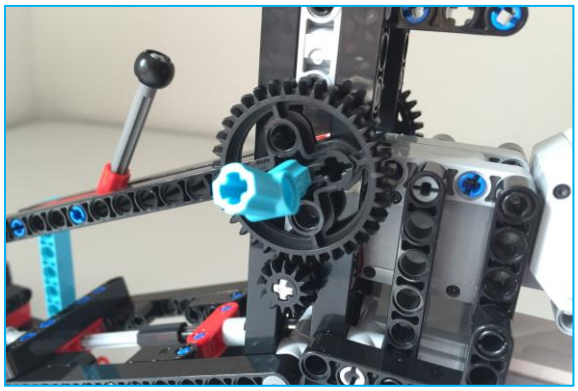
The rules of “Tower of Hanoi” are automatically enforced by TOW3R. For example, if you want to drop a bigger tire on a smaller one, the move is not executed, and instead only a sound is played. Thus you need not focus on the rules, but rather think about your recursive strategy.

In both play modes the moves are counted and the time to completion is measured. After solving the puzzle, the number of moves and the time used are displayed.

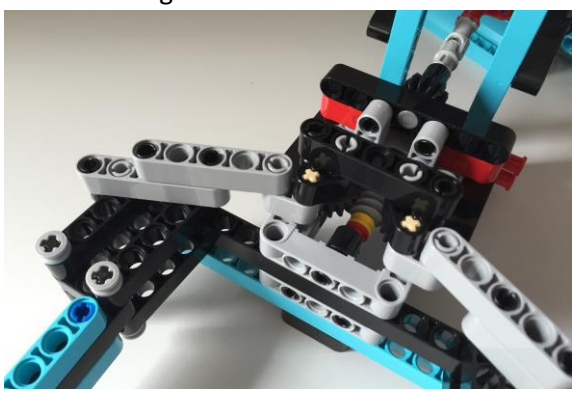
Hardware Design

- The robot has a base including the EV3 brick as a counter weight. The upper part of TOW3R is an arm that can be rotated left/right and moved up/down. At the end of the arm the claw allows to grab and release the tires. All three moves combined allow to play “Tower of Hanoi”.



- The original “Tower of Hanoi” has the three stacks on one straight line. For humans that’s perfect, but for a robot to move between the stacks with precision one would need to build some kind of track. Of course that can be done with LEGO Technics elements, but unfortunately these were not in the mix.
  Thus I decided to position the stacks of tires on the circular line. In this configuration the upper part of TOW3R can rotate on its base to reach the three stack positions. One big motor is built into the base and rotates the upper part with a gear ratio of 12:36 – giving power and precision. One motor used, two more left.
  
- Moving the arm up/down provided me a similar challenge as with left right. If I wasn’t limited in LEGO Technics elements, I would have built a forklift mechanism. However, given the parts I have, I decided to borrow from the concept known as “four link suspension”.
  The four links ensure that the claw moves up and down while staying horizontal. That is important to get a solid grip on the tires. Like with rotating the base I use a big motor with 12:36 gear ratio for more power.
  
- The claw took me longest to build. I don’t remember how many times I went through the cycle of building, testing and taking apart. Very early in the process I realized that the motor to open/close the claw cannot be directly integrated into the claw, because its weight would be very bad for the precision of the up/down movement of the arm.
  Moreover, it became evident that the claw needs to be able to provide huge pressure on the tires. Without that pressure the tires often fell of the claw when moving it around.
  The solution now looks like this: The claw arms open and close driven by a worm gear. The medium motor to drive it is built on the other side of the arm into the rotating part of the turntable. To ensure the power from the motor is transmitted to the claw flawlessly, I use high rotation speed. Then, to drive the worm gear the high rotation is transformed into strong force

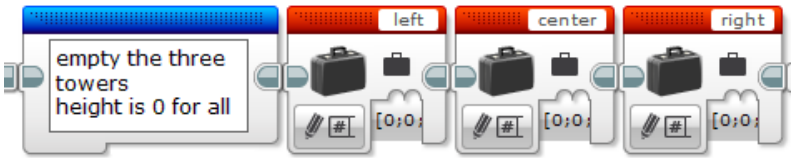
with a 12:36 gear ratio.



Software Design

I build the software driving TOW3R in parallel to the hardware. And as with the hardware it took several iterations of improvements to get it to today's state. Here are some note-worthy topics. If you look into the code you will find comments talking about these and a lot more.

- Very early in the process I realized that I need to represent the tire stacks within the program, because obviously the MINDSTORMS sensors are not able to "see" the current state of tire stacks. Thus I created three numerical arrays representing the stacks left, center, and right. Each array has three variables, representing the bottom, mid, and top position of the respective stack. In total this are nine variables, representing all possible positions of tires. The value of each variable is 0 for "no tire" and 1, 2 or 3 for small, mid-size, and big tire respectively.

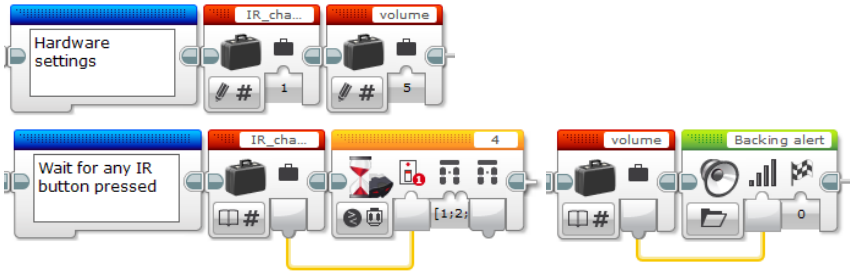


- The software is highly modular, consisting of 20 "My Blocks". The most important blocks are as follows. I'll share more details later when discussing the challenges I faced.

    - *Left*, *Right*, *Put*, and *Get*. They do exactly what the name suggest, namely turning the IR commands into movements of the claw.
    - *Display* shows the current situation on the EV3 display.
    - *Claw* is used to either grab or drop a tire. It uses another "My Block" *Claw_height* to determine how many motor degrees are needed to move the arm down before the claw is opened or closed.
    - *Open_Close* allows the claw to open as much as possible and close as tight as possible, regardless of the size of the tire being grabbed.

- Sounds are used a lot to inform the player what's going on. Moreover, they make TOW3R appear more as an industrial robot, and less as a toy. However, when you do extensive testing these sound can also get annoying. Thus I created one variable to control the volume of all sounds used.
  Another variable is used for IR channel. Assuming TOW3R is used in an environment with other

IR-controlled EV3s, this allows to pick a conflict-free channel.



The final model

Here is the result from building and programming. So lean back, but watch close TOW3R performing the perfect series of moves in “Tower of Hanoi” with three tires.

https://youtu.be/RArtA5xsREQ

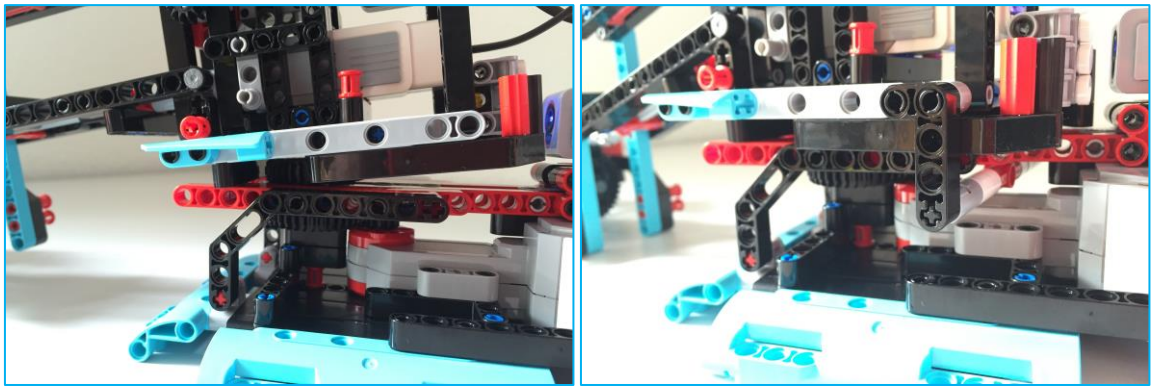
Challenges

Heavy load on the turntable

I started the build by making TOW3R a base with a turntable to rotate the arm and claw. I used the EV3 brick to ensure it would not fall over. This worked well in the beginning.

However, I had to improve the grabbing capabilities of the claw by adding more gears and thus more weight. Moreover, I had to make the arm longer to get more distance between the three stacks. The combined effect of both was that the upper part of the turntable tilted forward. Not only did that make precise moves of the arm impossible, but it also made rotation of the turntable anything but smooth.

To fix this, I had to add extra elements *under* the turntable.



Tire stacks accidentally destroyed

When you try out changes in programming things usually don’t work at the first time. In case of TOW3R this means the stacks may get hit and fall over. If you don’t remember what the exact set-up was before the stacks were hit, you need to start over.

To overcome this, I use the display to show what the current situation is. Remember, that I anyway have three arrays to represent the current situation. Thus I only needed to get the values of the arrays and the position of the claw into the display.

Looking at those numbers was ok for debugging, but to excite the player I took it a step further and turned the numbers into black rectangles of different width. Now you can see the stacks in the display and, if needed, use that to rebuild the tire stacks.

Actually, now you could even play without the tires, by looking only at the display instead. ☺

Grabbing tires of different diameters

When I started working on the software, I thought that for closing the claw I would run the motor for a certain number of degrees. Because the program knows where which tire is, taking the different diameter of the tires into account would not be an issue.

However, when testing it turned out, that the arm and the claw are never *exactly* at the desired position. Thus using a fixed number of motor degrees was not reliable. Instead I needed to invest into a more dynamic solution.

After some experiments I settled for a method based on the resistance the claw meets when grabbing a tire. The implementation is based on the rotation speed of the motor driving the claw. The program measures the speed of rotation and stops the motor when the speed is below a certain threshold, because the claw is grabbing a tire tight enough. The same works during opening the claw when it hits the physical limitation of the mechanics.

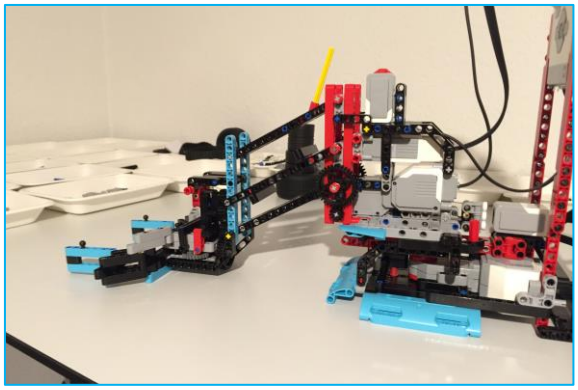
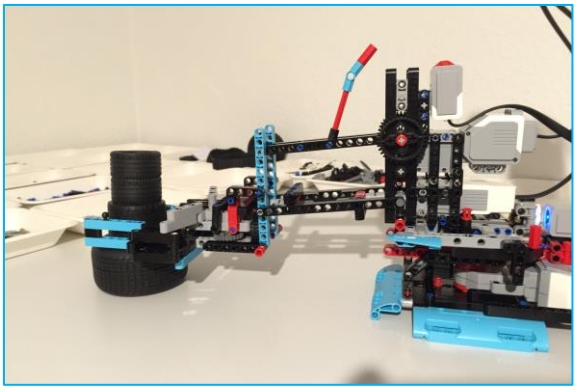
Up/down is also forward/backward

The “four link suspension” works nice to move the claw up/down. However, from testing I learned that it also moves the claw away/towards TOW3Rs base while going up and down. The reason is simple. The claw moves on a circle with radius equal to the length of the arm.

The issue with this is that for stacking tires, you want them to be centered. If they are not centered, the risk is high that the stack falls over or that the next grab is a miss.

The first idea was to make the arm longer, because a bigger radius reduces the effect. However, that was not really an option due to limited parts and because it would have let to even more tension on the turntable and thus to a further decrease of precision.

So I went for a different idea, namely to have the arm in horizontal position, when grabbing tires in the second position from the ground. In my first build the arm had been horizontal for grabbing at the topmost position (left photo). Rebuilding for the middle position (right photo) reduced the error and made is same for the lowest and topmost position.

Initialization

TOW3R is meant to be an easy to use puzzle game played by kids (and adults too). Thus you can assume that not every play ends as planned. Program runs will often be terminated earlier than planned, leaving the arm in some position and the claw in some stage of grabbing.

That can't be avoided, but needs to be taken into consideration when starting the program the next time. In other words, the question is: "How do you initialize TOW3R to always be in the desired starting position?"

The answer is *not* "make each of the three motors rotate to a specific degree". This does not work, because all motor degrees are set to zero whenever you start a program on EV3. Instead I needed to find a solution for each motor separately. For that I used the remaining two sensors.

- To get the turntable into the starting position, I installed the color sensor in a fixed position on the base and added three different color pieces on the rotating part of the turntable: X for left, Y for middle and, Z for right position. During initialization the program checks the sensor value and rotates the turntable left or right as needed.
- To get the arm into starting position, I installed the touch sensor at the top of TOW3R. During initialization the arm is moved up until the touch sensor is pressed. Then the rotation sensor of that motor is reset, allowing down movements by motor degree.
- To get the claw into starting position is easy. As stated above opening/closing is controlled by checking the speed of rotation anyway. During initialization the claw is simply opened "as usual".

Once I had the initialization in place for all three motors, I also used it for the actual moves to further improve precision. Rotation between the stack positions is controlled by the color sensor. The arm always moves up until the touch sensor is pressed and motor degrees at set to zero at that position. And of course the claw opens/closes until speed of rotation drops from increasing resistance.

No recursion with MINDSTORMS

I had planned for the optimal solution to "Tower of Hanoi" to be a recursive program. I actually build it with a "My Block" calling itself. That is the key concept of recursive programs.

Only when I tried to send the program to EV3, the LEGO MINDSTORMS software told me that it would not allow recursion. That was a bummer.

Nevertheless, the player deserves to see the optimal solution performed by TOW3R. Therefore, I implemented the 6 scenarios (3 starting position * 2 end position) as static chains of movements. Thanks to the "My Blocks", this was not a big hassle.

Nevertheless, I’m convinced that just by playing with TOW3R you can still get a good understanding of the idea of recursive algorithms and have some fun.

thomas@nano-giants.net