Patterns of Building LEGO® Mindstorms Robots

KYLE BROWN, IBM Software Services for WebSphere

This paper discusses patterns found in the design and programming of robots built using LEGO® Mindstorms, particularly those suitable for FIRST LEGO League (FLL) competitions. The paper is drawn from observations over five years of watching teams build LEGO Robots as an FLL coach, and three additional years as a competition judge.

Categories and Subject Descriptors: **K.3.2 [COMPUTERS AND EDUCATION]**: Computer and Information Science Education — *Computer science education*

General Terms: Robotics, LEGO, STEM Education

Additional Key Words and Phrases: None

ACM Reference Format:

Brown, K 2013. Patterns of Building LEGO® Mindstorms Robots. Publication Ref TBD, 19 pages.

1. CONTEXT: INTRODUCTION TO FIRST® LEGO® LEAGUE

Interesting children, particularly middle-school age children, in science and technology and inspiring them to eventually pursue careers in STEM (Science, Technology Engineering and Mathematics) fields is a challenging and difficult task. However, one thing that all children share is a love of play, and most also find that they can perform their best in an environment that allows them to learn and grow as part of a team of their peers. One extremely useful mechanism that I have found for accomplishing the goal of creating an interest in science while teaching children skills that will be useful in STEM careers and others has been the FIRST® LEGO® League (hereafter FLL) program. FIRST® is a non-profit originally founded by inventor and entrepreneur Dean Kamen in 1989 to inspire young people's interest and participation in science and technology.

According to their official web site (FIRST, 2013), "The mission of FIRST® and FIRST® LEGO® League is to inspire young people to be science and technology leaders, by engaging them in exciting mentor-based programs that build science, engineering and technology skills, that inspire innovation, and that foster well-rounded life capabilities including self-confidence, communication and leadership."

The FLL program works in the following way: Children are encouraged to form teams, working with adult mentors, of between 2 and 10 of their peers, between the ages of 9-14. These teams may be school-based, neighborhood-based, or organized through existing children's organizations such as 4-H, The Girl Scouts, or others. The program encourages problem solving and creativity by presenting kids with a real-world problem based on a new scientific theme, or *challenge* each year. The program consists of two parts. The first part is the Project. In this, teams must develop a 5 minute project presentation that presents a creative, original solution to a particular real problem within the year's subject area. Teams may perform a skit, a PowerPoint presentation, a song, or choose some other creative way to describe their solution.

2. MOTIVATION: THE FIRST® LEGO® LEAGUE ROBOT GAME AND LEGO® MINDSTORMS ROBOTICS

As if the presentation was not enough, the second part of the program is what truly motivates and inspires most children involved in the program, and is the motivation for this pattern language. This second part is called the Robot Game, and it involves the children in designing, building, programming, testing, and operating an autonomous robot that must perform a series of missions on a specially designed competition table. An example of such a table is shown on the following figure (Fig.1):

Fig. 1. FIRST LEGO League Competition Table

These robots are built using the commercial LEGO® Mindstorms robotics building kits, and are programmed using the LEGO® NXT-G programming language. LEGO® Mindstorms combines traditional LEGO® Technics parts with an intelligent microcomputer brick and intuitive drag-and-drop programming software (NXT-G[1]). NXT-G is a graphical language based on LabView from National Instruments (LabView). The robots must act autonomously while outside the small rectangular area in the lower right of this figure, known as "base". This means that the children have to not only design and build a robot of their own design that is mechanically capable of navigating around the board and operating the various mission models (built from LEGO® parts) to earn points, but they must also design, program and test the programs that instruct the robot how to solve these missions.

The culmination of the approximately 8-week FLL season is often in one or more tournaments, at which the teams compete with their peers in high-energy events that are reminiscent of sporting events, and also have a chance to present their solution to a real-world problem to a panel of presentation judges.

From 2007-2011 I was a team coach for an FLL team, and then from 2010 through the present day, I have judged at FLL competitions. Through this pattern language I hope to capture and formalize my observations about what makes a successful FLL robot and at the same time illustrate the applicability of the pattern form in a unique area of computer science and engineering.

3. THE PATTERN LANGUAGE

This pattern language is written in a modified Alexandrian style. The Alexandrian style has proven to be one of the more challenging to adopt to Software, as opposed to the more popular GOF or POSA styles since it has several elements that do not seem to fit well with the purely intellectual problems that programming patterns introduce. However, given the physical nature of Mindstorms robots, I believe that the Alexandrian style is an extremely appropriate one. In this pattern language (which is a small subset of a larger language that has yet to be fully developed) I will examine patterns in the following areas:

3.1 Robot Design and Strategy

[1] For a tutorial of the proprietary NXT-G language, see (Yocum)

This category of patterns includes those patterns that arise from the central problem of building a robot that can solve the robot game. The robot has to be able to move about the mat, and it also has to have certain engineering attributes like stability and modularity in order to successfully address this problem.

1. COLLECTION OF INTERACTING SUBSYSTEMS
2. DRIVE SYSTEM
3. DIFFERENTIAL DRIVE
4. CASTER
5. TANK TREADS
6. SYNCHRO DRIVE
7. ATTACHMENT FRAME

3.2 Attachment Design

This category of patterns come from the need to manipulate the various mission models that form the basis of the challenge. Different mission models necessitate different strategies for manipulation, resulting in different patterns.

8. PLOW ATTACHMENT
9. FORK TINES

3.3 Navigation Mechanisms

The hardest problem in building an autonomous mobile robot lies in helping it sense its surroundings and navigate among obstacles to carry out its tasks. This category of patterns deal with those issues.

10. ODOMETRY
11. LINE FOLLOWING
12. WALL HUGGING

1. COLLECTION OF INTERACTING SUBSYSTEMS

When you are building LEGO® robots for an FLL competition, determining the team's strategy for the year is the first essential set of decisions they must make. Chief among those strategic decisions is determining what their robot will look like and how it will function. However...

***

FLL Teams view their robots as single, special-purpose implement.

Many FLL teams do not perform at their best in a tournament because they do not have a coherent vision of what their robot is. FLL robots must accomplish several different tasks as set forward in the challenge documents. What's more these tasks change from year to year, and as students grow older, they find that there are certain types of tasks (such as moving an obstacle from one point on the board to another) that repeat over many different seasons.

The issue is that often a team will want to entirely "take apart" their robot at the end of the year. While this is a cathartic experience for the team, it also eliminates the accumulated engineering knowledge that the robot represents. But this is because the team views the robot as a specific object created for one purpose, and one purpose only - that year's challenge. Instead, the best FLL teams have learned to look at their robot in a different way. Therefore:

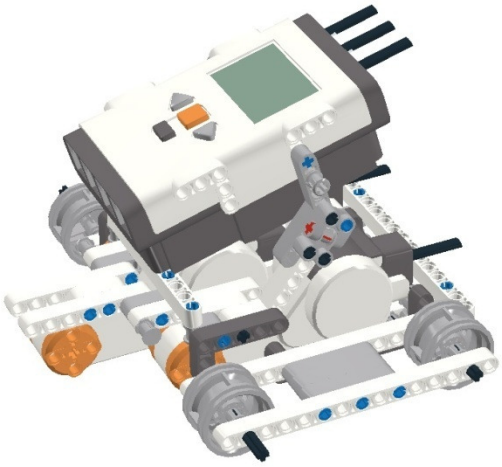
Build your robot as a COLLECTION OF INTERACTING SUBSYSTEMS, some of which will be reusable, and only some of which will be special purpose. Try to reuse the subsystems from year to year.

***

By far the most common subsystems are a DRIVE SYSTEM (2) and a set of one or more attachments such as a PLOW ATTACHMENT (8) that solve various missions in the challenge. Teams that are very advanced also find that programming tasks can also be divided into subsystems, and build reusable procedures using the NXT-G "MyBlocks" feature. These programming modules can also be reused from year to year by team members. For instance, a very common and suggested MyBlock that a team should build is a LINE FOLLOWING (11) MyBlock.

A potential pitfall of looking at a robot as a COLLECTION OF INTERACTING SUBSYSTEMS is that you may miss potential synergies among subsystems if you view them too independently. For instance, you may not see the potential for reuse among subsystems if you follow this principle too closely and create very specialized subsystems to solve every mission model in the challenge.

2. DRIVE SYSTEM



You are building a robot as a COLLECTION OF INTERACTING SUBSYSTEMS (1).

***

Your robot must have a stable base that allows for both navigation and manipulation of objects.

Therefore:

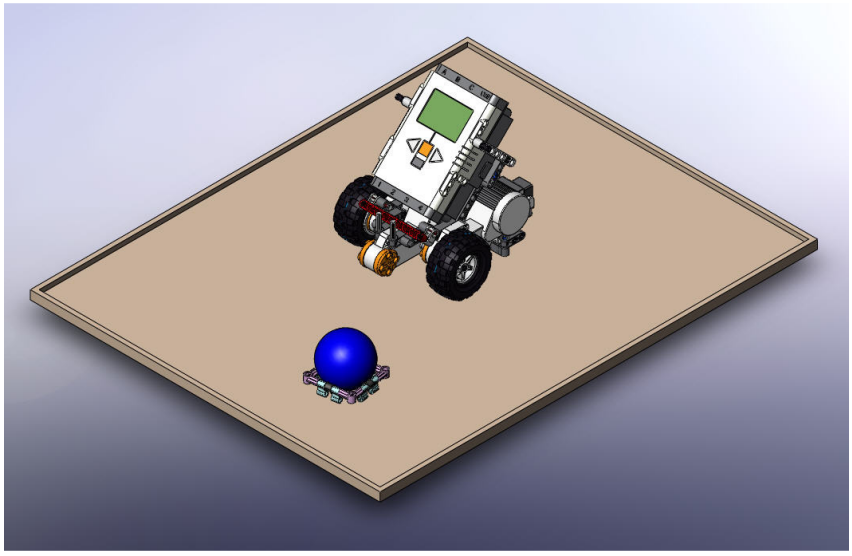
Build a Drive System that moves the robot that is separate from the attachments that manipulate the mission models and other objects.

***

A drive system consists of a sturdy ATTACHMENT FRAME (7) with motors, controller brick , skids, casters or wheels attached to the frame. Your drive system will need to provide attachment points for your attachments to connect onto, such as a PLOW ATTACHMENT (8) or FORK TINES (9).

Given that mobility is a critical attribute for a robot, building a separate drive system has few drawbacks. However, one thing you may want to consider is the possibility of making one other subsystem a part of the drive base itself. For instance, a common strategy involves collecting several objects and bringing them into base. Having a "holding pen" as part of your drive system to hold these objects temporarily as opposed to making that a separate attachment may be a useful strategy in some cases.

3. DIFFERENTIAL DRIVE



You are building a Robot as a COLLECTION OF INTERACTING SUBSYSTEMS (1).

***

You need a simple DRIVE SYSTEM (2) that is both easy to program and easy to build. especially for younger students.

Therefore:

Build a drive base founded on a Differential Drive approach that uses two motors, one on each side of the robot. As both motors turn in the same direction, the robot moves in that direction. As the motors turn in the opposite directions, the robot turns in place around its center.

The figure below (Fig. 2) shows how a simple robot based on the differential drive approach can turn clockwise.

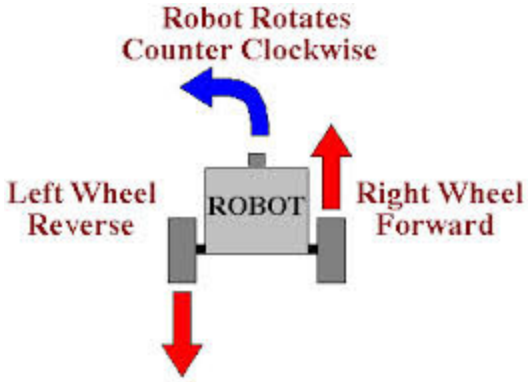
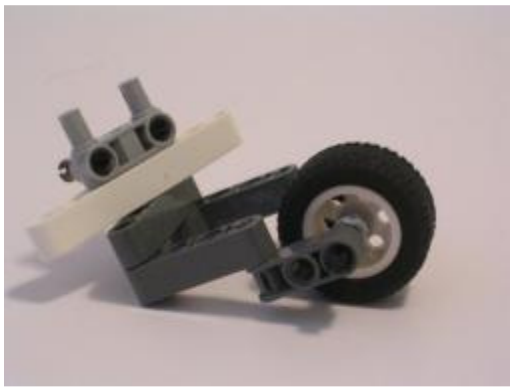


Fig. 2.: Differential Drive turning

This is by far the simplest type of drive system to build, and it is the one used by the Tribot robot whose building instructions are included with the Mindstorms kit. This robot is depicted in the picture above. A simple differential drive will often have wheels directly attached to the motors, but more sophisticated ones will use gearing or other mechanisms for more flexibility in connecting the motors to the driving wheels.

One problem is that a robot cannot stand on two wheels alone. For stability it needs at least one other point of contact with the mat. This can be a simple skid, additional wheels or a CASTER (4). Another potential problem is that a robot build with a DIFFERENTIAL DRIVE pivots around the center of the robot; making it difficult to position some attachments accurately in some cases. In those cases a SYNCHRO DRIVE (6) may be more appropriate.

4. CASTER



You are building a robot with a DIFFERENTIAL DRIVE (3). You have chosen to use wheels as your motive appendages.

***

You need a way to allow the robot to turn more easily than fixed wheels or skids will allow

Therefore:

Build a caster that allows easy motion in its current direction, but facilitates turning in place.

Casters can be built in several different ways, but the most common are to build them around wheels or balls. The key to building a successful wheel caster is that both the wheel axle and the center axle need to swivel freely. The picture above shows a wheel caster with independent wheel axles and caster rotation axles.

The problem with building a successful ball caster (which does not need axles) is in building a cage that can house the ball fully yet still allow for it to both rotate and be connected to the robot. One good solution for that using LEGO® 45 degree Technic beams is shown below (Fig. 3).

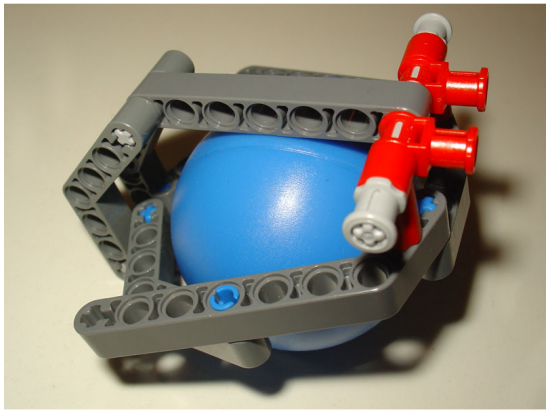
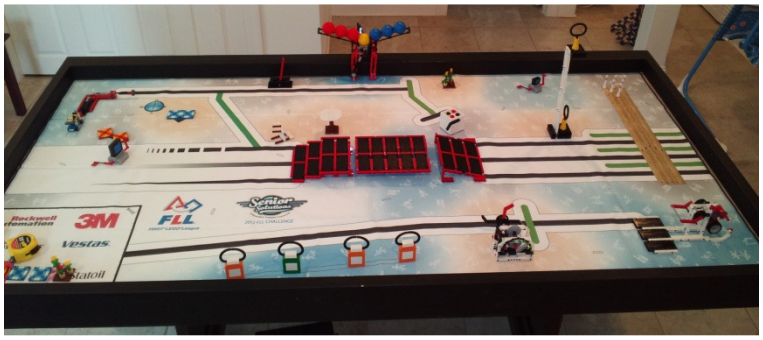


Fig. 3.: Caster built with LEGO® Technic beams

5. TANK TREADS



You are building a Robot with a DIFFERENTIAL DRIVE (3). In many challenges, there is an obstacle that must be climbed over, or onto, such as the bridge that was featured in the 2012 Senior Solutions Challenge shown above. This means...

***

You need a way to allow the robot to easily climb over obstacles, while still being easy to program and build.

Therefore:

Use tank treads as your motive appendages.

Instead of wheels and casters, use tank treads. The Advantages of tank treads instead of wheels are that they provide good grip, especially when your mission strategy requires you to climb over obstacles such as the bridge shown above. Likewise, even on slippery surface, tank treads provide relatively low slippage. An example of LEGO tank treads is shown in Fig.4.

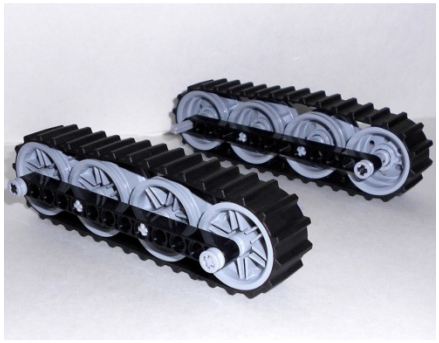


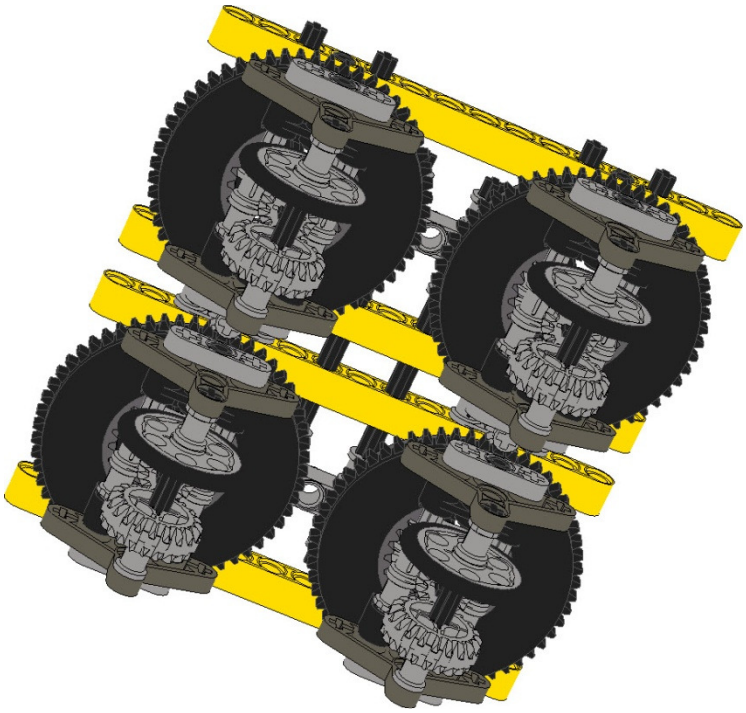
Fig.4.: LEGO Tank Treads

The primary disadvantage of using tank treads (at least in many designs) is that the treads can move independently of the drive wheels that power the treads (e.g. the treads can slip). Tank treads can also sometimes have problems with driving in a straight line as any slippage on one side must be compensated for through navigation approaches or the robot will drift to one side or the other.

***

Tank treads can have problems with slippage that cause unpredictable rotation and can make accurate navigation with ODOMETRY (10) difficult.

6. SYNCHRO DRIVE



Sometimes solving a mission in the most obvious way requires grabbing an object and holding it in a particular orientation relative to the rest of the board while it moves. This is difficult when you have to consider the limitations of existing drive systems. So, if your mission strategy and attachment design precludes the use of a DIFFERENTIAL DRIVE (3), even one incorporating TANK TREADS (5) since you cannot rotate the robot while turning, you need to do something else.

***

You are designing a robot that must turn in place without rotation.

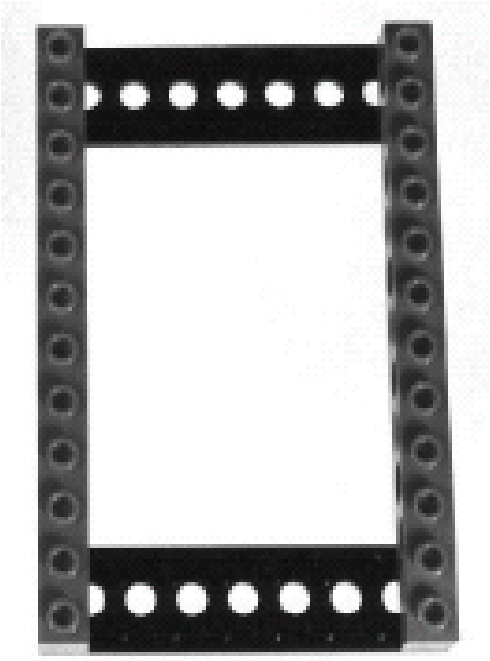
Therefore:

Build a robot with a Synchro drive

In a Synchro drive, all the wheels are simultaneously powered and turned. One motor provides the motive power to the wheels to move the robot forward, while another motor turns the wheels, using a LEGO® turntable to independently turn the wheels. An example of this is shown in the picture above.

The primary advantage of a Synchro drive is that it is extremely accurate and can, as noted, turn in place. Its primary disadvantage is that it is very complicated to build and is well beyond the abilities of most first or second year FLL teams with younger students.

7. ATTACHMENT FRAME



You are building a DRIVE SYSTEM (2) and need to provide a way to house the motors, axles, wheels or other parts of the Drive System.

***

You do not want to attach your motors and other parts of your drive system to the NXT Brick because that makes it difficult to access the back of the brick to change batteries or even to attach the USB cable to the brick to program it.

Therefore:

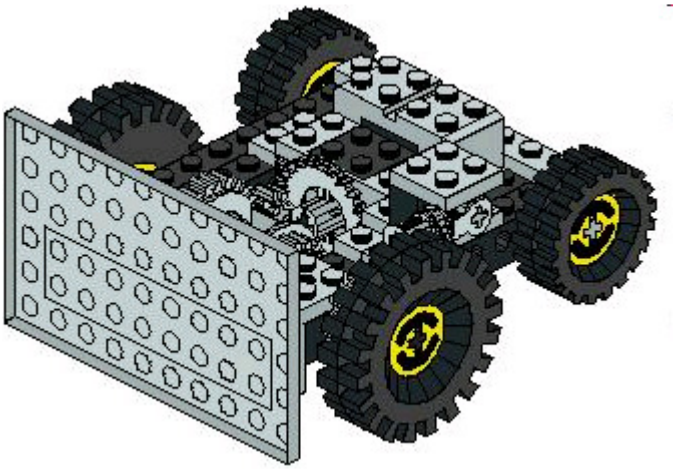
Build a Attachment Frame that provides aligned holes in two different directions to allow for the placement of axles, as well as attachment of motors, wheels and other parts including the NXT brick.

There are many common approaches to building frames that meet the requirements above, but a common one is to connect LEGO® Technic beams with 2-hole plates on the end. For even stronger frames use 2-hole plates on top and bottom.

***

The common alternative to building a Attachment Frame is to build your DRIVE SYSTEM (2) using the LEGO® NXT brick as a structural element. As mentioned earlier, this makes it difficult to change the batteries or even change out the brick if necessary.

8. PLOW ATTACHMENT



You are building a robot that is a COLLECTION OF INTERACTING SUBSYSTEMS (1), that has a common DRIVE SYSTEM (2)

***

One of your missions requires you to relocate an object from one location on the board to another. There is a line of sight from the original location to the destination location

Therefore:

Use a scoop or plow attachment to relocate the item

This is by far the simplest attachment to build, and usually the simplest to program for as well. In fact, problems that can be solved by a plow (or a slight variant that adds a wedge to the bottom forming a scoop) are the best ones to start teams on solving. The FLL mission designers usually intentionally add one or two missions that can be solved through this approach to each year's challenge in order to give new teams, especially of very young students, a chance to feel a sense of accomplishment in solving something.

***

These missions usually take the form of requiring an object placed on the board in a specific location to be moved back to base. If the team can navigate their robot to a point behind the object, then they can move it with the plow to the base by moving the robot as a whole. This type of problem can be solved using ODOMETRY (10), and can be usually solved even with a simple robot built using DIFFERENTIAL DRIVE (3) such as the Tribot that is featured in the instructions for the LEGO® Mindstorms kit. Finally, a plow can sometimes improve the accuracy of LINE FOLLOWING (11) by acting as a light shield against extraneous light sources.

9. FORK TINES

You are building a robot that is a COLLECTION OF INTERACTING SUBSYSTEMS (1). Some common missions require specialization in their solution.

***

Your robot needs to grab an object at a fixed height and deposit it in another location. A common instance of this is the need to relocate a LEGO® Loop.

Therefore:

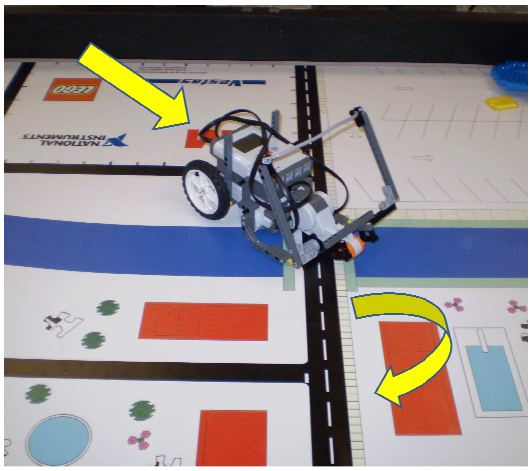
Build a simple fork tine attachment, either powered or unpowered. The simplest version is a fork directly attached to a forward-facing motor. Multiple tines allow for variation in accuracy in getting to the loop. Single tines give more control but require precise navigation.

Loops are a common instance of an object that must be moved to base as part of a mission. However, the designers of the FLL challenge often make it more complex for the teams by placing these loops behind other objects, or on top of other objects, as opposed to simply placing them on the mission mat as shown above. This makes it impossible to solve this mission with a plow attachment. Thus, a fork tines attachment, which can move up or down, can usually address the requirements of this type of mission. An example of a powered fork tines attachment is shown in the picture above.

***

Fork Tines are easy to use with either ODOMETRY (10) or LINE FOLLOWING (11) navigation methods, but in the latter case, care must be taken with the positioning of the light sensor to not interfere with fork operation.

10. ODOMETRY



You are building a robot , probably one with a DIFFERENTIAL DRIVE (3) such as the Tribot. It needs to navigate to mission models, particularly those nearby to base.

***

Your robot needs to make simple turns and drive short distances. You need to direct the robot to perform these tasks in a simple easily understood way, especially one that can be comprehended by young children not familiar with complex programming tasks.

Therefore:

Use odometry as your method of navigation; measure your distances and turn radius and program the robot to move exactly that much.

Odometry is by far the simplest navigation method to program with NXT-G. What facilitates simple program is that the Mindstorms motors have embedded rotation sensors built directly into the motor housings. NXT-G features a Move block that allows you to tell a motor to turn a certain number of rotations. For instance, the following program (Fig.5) tells a robot with motors attached to outputs B and C to move forward 10 rotations, then turn to the right for 2 rotations of the left-hand wheel.

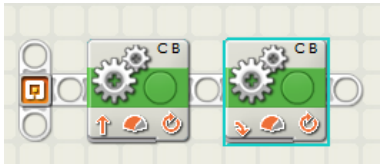


Fig.5.: Sample Odometric program

The problem is that odometric methods are not particularly accurate - they are very sensitive to the design of the robot, even requiring the matching of motors to remove slight differences in how fast they stop or how far they rotate. They are also particularly prone to wheel slippage and center of gravity variations.

***

Since a major problem with odometric navigation is accuracy, WALL HUGGING (12) can improve the accuracy in some cases. Where lines are available, LINE FOLLOWING (11) should be preferred since it has better accuracy.

11. LINE FOLLOWING

You are building a robot (probably using a DIFFERENTIAL DRIVE (3)) that needs to navigate precisely to a particular mission model. There are lines available on the mission mat near those models. For instance, the Food Factor challenge (shown above) featured many straight lines leading right to particular mission models.

***

Odometry has many drawbacks as pointed out earlier. In many cases, you need to more precisely navigate to specific obstacles when a line is available leading to that obstacle.

Therefore:

Use a line following algorithm and a light sensor.

A simple switch-based line following program is available in the NXT Education software instructions (LEGO Education). That program, written in the proprietary graphical NXT-G language, is shown below (Fig.6). It is appropriate for a simple robot with two driving wheels such as the Tribot described earlier.

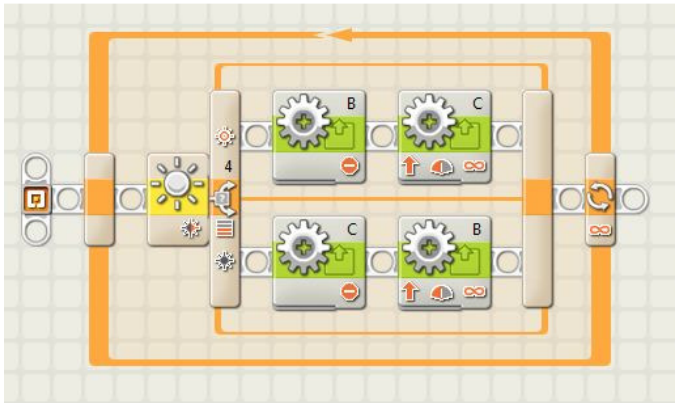


Fig.6.: Simple Line Following Program

Here we see an outermost loop containing an inner switch statement. The two conditions of the switch statements (the top and bottom inside the switch block, each contain two "motor" blocks that can control the activation or deactivation of a motor. In this simple program, which requires a light sensor to be attached to input port 4, the robot will turn to the left (the motor attached to port C will turn on and the motor attached to port B will turn off) if it senses light, and then turn to the right (the B motor will turn on and the C motor will turn off) if it senses darkness. Thus the robot will wiggle its way from left to right following the line - in this case forever, since the loop is an infinite one.

A difficulty that stops many students lies is deciding how long to carry out that outermost loop. Looping for a particular count is not particularly accurate, as small variations on when you pick up the line will often vary the number of turns needed to complete following a line for a specific distance. Instead, adding another sensor (a light sensor or touch sensor) that looks for a separate condition in the outermost loop is usually the most appropriate approach. Another consideration that stops of confuses many students lies in setting the appropriate thresholds for the values of light and dark. Small variations in the light sensor and in the printing of the mat need to be accounted for in setting these values, resulting in differences in how the robot will perform in testing and in competition. While the light sensors have a built-in light source, a light shield is also often needed to ensure that ambient light does not interfere with the light sensor measurements.

More advanced algorithms do more than comparing one value of light or dark - they have multi-way switch statements that compare multiple values and vary the amount by which the motors turn.

One very advanced PID (Proportional Integrative Derivative - a common algorithm used in machine control that integrates feedback) based line following algorithm can be found here: (Sluka, 2013) Others are available online as well for interested students.

12. WALL HUGGING

You are building a robot that is using ODOMETRY (10) for navigation but you want to improve its accuracy.

***

Odometry is inaccurate over long distances, but line following is often considered too difficult for younger teams to master. You need to improve the accuracy of the robot over the long stretches when it moves a great distance down the board.

Therefore:

Use a wall-hugging approach that takes advantage of the fact that the FLL Competition table has a smooth wall a fixed distance from the edge of the mission mat.

Many teams soon realize that while ODOMETRY (10) works over short distances, that over long distances it becomes very inaccurate - and worse, a slight variation on one turn or straight-away will make all the turns and straight-aways that follow go off in an increasingly wrong direction. So a common technique is to mechanically modify the robot in a very slight way - that is to add one or (preferably) two LEGO® pulleys (as shown above) to one side of the robot, and then to introduce a very slight occasional turn into the Move block in the direction towards the wall. This will cause the robot to hug, or follow the wall on that side, thus increasing the accuracy in one dimension by fixing the distance of the robot parts to the wall. Especially when combined with a light sensor or touch sensor to find when to stop its forward motion, this is a simple, easy to program compromise for improving accuracy for odometric methods.

4. APPENDIX 1: PATTERN MAP

The following pattern map (Fig.7) illustrates the connections between the patterns. An arrow indicates that a pattern either is referenced by another pattern in its resulting context, or specifically references that pattern in its context. The patterns naturally fall into the three major categories of robot design and strategy, attachments and programming and navigation as seen earlier in the pattern language.

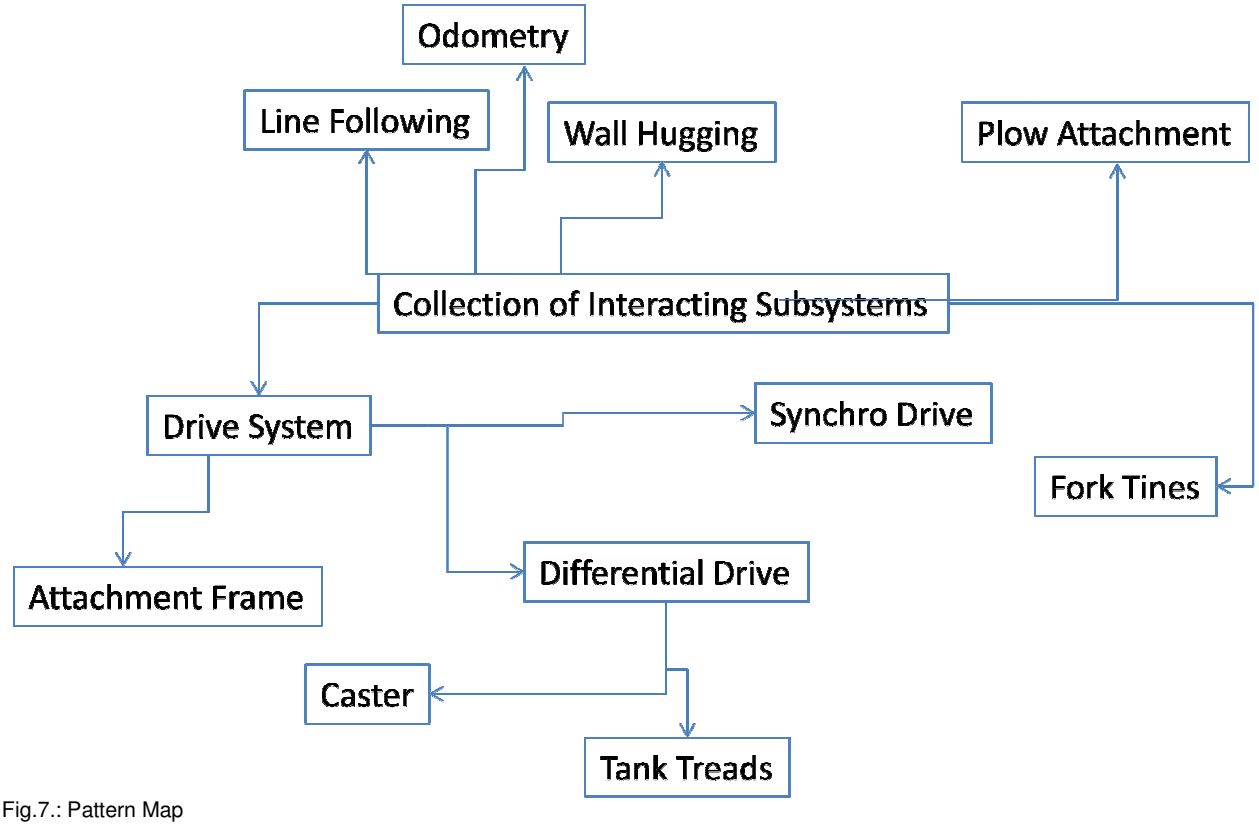


Fig.7.: Pattern Map

5. APPENDIX 2: OTHER PATTERNS

In this paper I have examined only a small subset of the full set of patterns that I have identified as being applicable to FLL robot design. Others include:

Robot Design and Strategy
Mission Zones
Combined Zone Program

Attachment Design
Holding Pen
Spike
Pincer
Forklift
Dumper
Pusher Piston

Navigation Mechanisms
Light Shield
Color sensing

Finally, there are additional detailed patterns of LEGO® design that can be helpful to students building sophisticated drive and attachment systems. These include:

Linear Motion
Rack and Pinion
Piston Rod
Lead Screw
Scissor Arm

Rotary Motion
Gear Train
Gear Ganging
Stop Bushings
Idler Gear
Bevel Gear
Crown Gear
Worm Gear
Clutch Gear
Ratchet
Pulleys and Belts

FIRST. (2013, June 17). *http://firstlegoleague.org/*. Retrieved from FIRST LEGO League.

LEGO Education. (n.d.). *LEGO NXT Education Software*. Retrieved July 29, 2013, from LEGO Education: http://education.lego.com/en-us/lego-education-product-database/mindstorms/2000080-lego-mindstorms-education-nxt-software-v-2-0/

Sluka, J. (2013, July 17). *PID Algorithm Description*. Retrieved from NXT Programming Website: http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html

Yocum, D. (n.d.). *StemCentric NXT-G Tutorial*. Retrieved July 29, 2013, from STEMCentric: http://www.stemcentric.com/nxt-tutorial/