

1 Introduction to FLL -

1.1

1.2 Goals/Objectives

1.2.1 Hard Skills

Hard skills are the abilities we learn that produce direct, immediate physical results that we can see, hear, use, or otherwise enjoy. Generally FLL will be team member's first experience to programming and to solving a problem with a technological device. Some of the hard skills you'll learn with FLL are analysis, strategic planning, experimentation, problem solving, designing, building and basic mechanical design principles, basic programming principles, and the concept of an embedded computer controlling a device. These skills will add value to your robot and to future projects.

1.2.1.1 Learn the Problem Solving Process

One of the most important skills to learn in FLL is the problem solving process. For kids, it can be simplified as following:

- What's the problem?
- Brainstorm solutions
- Evaluate solutions and pick one
- Implement the solution you picked
- Evaluate the solution you built

The most important step in the problem solving process is simply knowing, what's the problem. This seemingly simple statement still fouls up every day engineering problems. No matter what you doing, know why you doing it and what you trying to solve. Is the problem to go around the course? Or to go around the course as fast as possible? Knowing what the problem is will make a big difference in how you approach a solution.

1.2.2 Soft Skills

Soft skills are the abilities we learn that produce results which are harder to perceive, but which are essential to support the hard skills. Examples of soft skills are being on time, tact (don't be negative with criticism), teamwork, focus, confidence, compromise, courtesy, perserverence (never give up), and respect for other solutions and teams. The successful team will learn all of these and more with FLL. These skills will add value to your team, your process, and your future. Don't overlook soft skills. They can make a big difference in the success of your team. You have

probably heard athletic coaches talk about "good team chemistry". That means the soft skills on that team are working well.

-
-

A favorite story is from a coach asking his students at the end of the year what was the most important thing they learned from the season. One particularly bright child said "I learned that other kids can have good ideas too."

1.3 1st-year Team Realizations

- So as not to become overwhelmed, try not to coach more than one team.
- Co-coach with another parent/volunteer and try to find a technical professional as a mentor.
- Keep a positive attitude.
- Always, always stress to the kids that getting a solution and showing it at the tournament makes the team a winner.
- Stress the process of getting there.
- Never, never make winning a priority because you only end up with disappointed kids (and parents).
- Have fun!
- Do not lose sleep at night worrying about your team's solution in comparison to others. The team is solving the problem, not you.
- Do not rack your brain to think of your own solutions. This makes coaching more frustrating than it already is.
- There is help and support for you, check the web pages or call your xxx!

You cannot do your job without the proper materials, make sure you have your own copy of the FLL Rulebook, LEGO Mindstorm programming guide and constructopedia.)

2 How to Start

2.1 Understand the Mission

The mission of FIRST™ LEGO® League is to provide an inspirational learning experience for children that celebrates technology. By bringing together interested adults, technology professionals and high tech tools, FLL strives to engage children in the wonders of technology and provide them a window of opportunity to discover the excitement a future in technology may hold.

2.2 Understand the Commitment

Coaching a group of adolescents can be a challenge for anyone! For a FLL team to be successful, both the coach and the team members must be committed. Without commitment, the significant opportunities which abound through the problem solving process will be lost. Problem solving is a challenging adventure into the unknown. Unlike exercises with predetermined outcomes, problem solving requires risk taking, experimentation and failure. {Fred's comment: this previous

sentence is extremely important, it should remain.} Researching, designing, building, and testing are the key elements of the problem solving process and to do these well takes time. You must be ready to invest significant time with your team for them to be successful. .{Fred's comment: and so is this one, it should remain.}

2.3 Organize your Team

Listed below are a few successful approaches to defining an FLL team. As you go through this information, our hope is that you get a handle on the initial directions that must be pursued to get an FLL team up and running. For further support , log onto www.hightechkids.org/fll to access more information on team formation as well as a comprehensive coaches support guide

2.3.1 Identify a Coach

Coaches can be teachers, parent volunteers, or technology professional mentors. Greatest success happens when a combination of the above as volunteers combine their specific skills to generate the richest experience. The selection of a coach will be a determining factor in defining the teams operational profile. In other words, if the primary coach is a teacher, practicing during the school day may be viable. If the primary coach is a parent or technology professional, the after school profile is probably more likely. Don't worry if you're not a "techie". Successful FLL teams have been coached by non-technical folks. Remember, success is not defined by how many trophies you win.

2.3.2 Team Operational Profile

Teams operate under several categories. They can be associated with a school, managed by school personnel; or they can exist as an independent club operated by Scout Troops, Park and Recreation volunteers, or a group of interested neighbors.

School Team: School teams can operate as co-curricular or extra-curricular activity; or a combination of both. Following are scenarios that can be adopted.

After school profile: Students meet after school for practices. Practice times/days vary according to defined goals. A general standard here is two days a week to start, working to three or four days a week as the deadline nears, with each practice lasting 1.5 to 2 hours. This is probably the most common method for first-year teams.

During school/daily profile: Students have a 45 minute project period written into the normal school day.

During school/weekly profile: Students are taken out of their normal class one day a week for a project period of 2 to 4 hours.

Club Team: Many schools allow outside clubs to operate during the school day following the scenarios listed above. If this is the case, it may be advantageous to adopt the school time as practice time. If the club is totally independent of the school the after school practice profile will fit the best.

2.3.3 Cost

To participate in FLL, all teams must pay a non-refundable registration fee and purchase a base kit of materials. There is an additional cost for materials needed to build a table and playing field. These materials are available at a local hardware store. The table and base kit are reusable each year, cutting a team's second-year investment by approximately one-half. (Please visit www.usfirst.org for specific cost requirements.)

2.3.4 Practice Venue

An adequate practice venue is contingent upon having access to the necessary computer hardware and the space to build and test the robot. Eventually, each team will want to set up a practice field to give your team a chance to test actual project performance. The practice field can be as simple as clearing an area on the floor and placing any challenge components within the space, or as elaborate as a full competition table. Some teams have found it advantageous to share access to a practice field with other teams in the area.

2.3.5 Team Size

Team size is a function of resources and the team operational profile. A computer lab and site license for the MindStorm software can put 20 kids on programming at the same time. With a team of volunteers, multiple "Exploration Stations" can be set up with 2 to 3 students per station. If you are overwhelmed with students interested in the program and have adequate funding and coaching resources, you can consider forming multiple teams. With only one kit, a team of thirty students is still possible but you may have to limit the building and programming to a core group of students and engage the others in development of the hypothesis and team identity (promotions, fund-raising, T-shirt design, etc.). In all cases, your organizational design goes a long way in answering the question of how many students to involve.

2.3.6 What's Ideal?

A common question is, "what is the ideal team size?" Or "what is the ideal practice place?" As discussed earlier in this section, there is no ideal size but a typical size is 5-7 team members with one or two coaches. A larger team than that really requires a school environment which supports a team of that size. Namely a large classroom or lab space with a number of computers and a number of adult coaches. It also takes more organizational skill to manage a team of that size since you need to have multiple activities to keep all the team members occupied. A larger team really becomes a classrom, which is great IF your school is ready to take that on. Section 3.4

gives an excellent overview of how to structure a team of 6 team members.

When and where you practice is again a function of your team structure, but typically practices occur in a school classroom after school or in the evening. You will need access to at least one computer. If the coach is a parent or technical professional practices can occur in a house. Obviously this would only work if you had less than seven team members (unless of course you have a really big house). Many a coach or parent has had their basement or family room turned into a FLL design lab during the last few weeks of the season. If you practice in a school, ideally you would like to have at least some small space that you can claim as your own during the season. If nothing else you will need space to store your robots and competition set-up. Space is always tight in a school. Remember custodians are your friends. They always have a storage space hidden somewhere. Hauling everything home after every practice gets old. Also think about where you would practice on a weekend.

As a minimum, you need one computer system for your team. At a minimum, you also need one Robotics kit. Having more than one computer or Robotics kit is certainly useful but not mandatory. The computer needs to meet the minimum specifications for running the software. It does not need to be a laptop, but if a parent or someone else on the team can provide a laptop at least for the tournament, it makes transportation easier.

2.4 Roles and Responsibilities

2.4.1 Coach

- **First of all, relax!**
- It is your **SOLE** responsibility to instill the FLL spirit in your team. Remember, if you emphasize the score as a measure of their success, you could be setting up your team to feel like failures. Please emphasize the learning process, the experience, and FUN are the worth while goals to achieve.
- FLL is about the process of preparing the solution -- not the result of a competition. It is counter-productive to try to set goals like "winning the competition. This way, the kids can feel they have met their objectives for the year regardless of the outcome of the competition.
- Organize the team if you have not already been so
- Organize practice schedules
- Make sure a course is built for practice. This could be done by parents or other adults. Make sure the kids help as much as possible.
- Understand the rules. Ask for clarifications if you don't. . Help the kids to understand the Challenge as thoroughly as you do.

- Contact?
- Remember, this is the kid's solution not yours
- If you feel lost, ask for help online. Chances are someone else is in the same position.
- Celebrate your accomplishments!!
- Set up a regular meeting schedule and give a written copy to all the parents.

- Team-building is important. It is difficult to be creative "on schedule". Sometimes, just letting the kids have some fun together will allow them to develop better communication and respect – which will lead to smoother progress when work resumes.

- Respect for teammates is important. It is a good idea to create Team Rules such as "Team members can disagree with a proposed idea, but should never call each other names". The process of creating these rules can be a nice "team building" exercise.

- The coach should help the team set up a timeline for task completion and keep team on task at meetings

- The coach must deal with the parents. Parents can be either a big help or a big problem for an FLL team. With some parents, it will be important to define the limits of their role -- namely that the Coach has the final word on how the Team should be run.
 - Meet with parents to be sure they understand the FLL philosophy, goals, & general rules.
 - Provide parents with all tournament plans. They will need copies of maps, schedules, and procedural information.
 - Enlist the help of parents.
 - Coaches should communicate to parents and team members the value of appreciating other team solutions rather than resenting them in comparison to their own.

- Maintain contact with the school administration.
- Good public relations with administrators and boards of education will help gain their support & cooperation.
- Keep administrators, board of education members and staff aware of the team's progress, news coverage, and success.

- **Have fun!!**

2.4.2 Mentor

A mentor is defined as a wise and trusted counselor or teacher. A mentor is not defined by age. A mentor may be a high school or college student, a technical professional, or just someone with a lot of experience building Legos or robots.

Their some general guidelines all mentors should follow. Remember you are not the coach. Let the coach handle that team organization and structure issues. You may advise the coach on these issues but do so privately.

It is very important that you try to build a robot yourself, if you have not done so previously. This is true for coaches as well. While you may be an accomplished programmer or engineer, you need to understand the capabilities and limitations of the MINDSTORMS kit. Besides, it's fun! If you don't think it's fun, you should rethink being a mentor.

2.4.2.1 High school and/or college student

Students can make very good mentors. They are closer in age to the adolescent team members and the team members relate differently to the students than they do to adult volunteers or coaches. The combination of younger and older mentors can be very powerful.

Some guidelines for student mentors:

- It is best to work with an adult coach. It is possible for perhaps a college student to be the sole coach if they are experienced with this sort of program.
- Try to work with a small group of team members focusing either on programming or robot construction techniques. A valuable role you can play is to help the team members with structured experimentation rather than just randomly changing things. Remember the problem solving process!
- It is not unusual for high school or college students to have better computer skills than the coach. Help with technical difficulties. Don't worry, there will be some!
- If you are a college student, you probably have more programming experience and theory than a high school student. Help the students with programming concepts.
- Older students have quite a bit of experience with research papers and projects. Help the team members with their hypothesis research. What are good sources of information? How you determine whether the Web site you are using for information is reliable? How do you present your research in an understandable manner?
- Let the coach or adult volunteer deal with discipline problems.

2.4.2.2 Technical Professional

"Techies" can be found in many walks of life. They may be engineers or computer scientists. They may be a graphic artist with lots of Web design experience. They could be a scientist or a college professor. But generally the technical professional mentor has developed technology for

living and understands the process. The confidence and skill the mentor brings to the team can be invaluable.

This section contains a list of things to consider as a mentor. In general as a mentor you should help the team with the specific skills that you know (programming, mechanical design, etc.). Don't hand the team a solution, let them experiment and help them set up experiments to determine different approaches. Many times kids will try many solutions without any structure to their attempts. Help them structure the experiment and then help them analyze the results. One of the most valuable things you can provide as a mentor is helping the team develop good decision-making skills. Technical problem solving is all about defining the problem and then developing good criteria to evaluate possible solutions. As a professional you do this everyday. Bring that experience to FLL.

- Remember, the kids are not engineers. They are kids. The work environment in FLL is probably more chaotic than your workplace.
- Help the kids set up experiments with only one independent variable. For example don't change the programming and the gearing mechanism at the same time.
- Help the team set achievable goals. Remember, FLL is not building NASA-quality robots.
- Try to get at the kids level, both literally and figuratively. Don't use words like *autonomous* or *conceptualize* or any other buzzwords from work.
- Being a mentor is a good chance to try out skills you may have learned during work training sessions. Chances are you have taken classes on quality or team building or management practices. There is no better management training than coaching a kids team.
- Remember FLL is not the cutthroat competitive environment you probably work in. Try not to come directly from work to practice. It's difficult to make that transition quickly.
- Meet with the coach on a regular basis and plan out strategy. Perhaps you can share some other responsibility like checking Web sites regularly for changes.
- Think of the sports skill camp analogy. For example, a kid may go to a baseball camp and have a coach talk to him/her about batting (stance, grip, strike zone), pitching (gripping the ball, mechanics, rhythm), and fielding (approaching the ball, getting ready to throw). Think of yourself as one of those skill coaches. One practice focus on programming (code reuse, programming with a sensor). At another practice focus on different methods of robot locomotion (gears, pulleys, tracks vs. wheels,). Remember you can't play baseball without learning the basics. The same goes for building robots. And just like in athletics, practice, practice, practice.
- Remember you are mentoring the coach as well as the kids.
- Be patient with both the kids and the coach.
- Be prepared for kid time vs. adult time. You know, like dog years vs. human years. For you something a year away is seven years for your dog. It's sort of the same way for kids. The tournament for you may be two weeks away, but for the kids it's two years.

- Make sure you are there on tournament day! You can never have too many adults helping the team that day.
- If the coach is a teacher, remember the teacher is not an engineer, he/she is a teacher. Teachers have a classroom of kids all day, they're not sitting in front of a computer all day. So email is not their normal communication mechanism. Nor is fiddling around with a computer usually their preferred activity .
- The kids are solving the problem not you.
- Always direct the kids by asking questions. Even if you see they are going down the wrong path, let them go and figure the problem out on their own.
- Explain the kids basic principles such as what does a loop in programming mean. If they continue to get stuck suggest ways around the problem.
- Trying to solve a problem where the solution is unknown is not a typical teaching environment but it is a typical technology development environment. Bring your confidence and skill in dealing with that type of work environment to the practice session.
- Realize that like any development problem, 20 percent of the work takes 80 percent of the time (usually crammed into the last week). And teams of kids are just like teams of adults, different personalities that don't always get along and some kids will do more work than others.
- Share your war stories with the kids and listen to theirs. The kids love to tell theirs.

2.4.3 Parent/Volunteers

- Parents need to make sure their child attends all practices.
- Parents can act as a resource to teach team members skills needed to solve their problem.
- Parents should attend tournament for their support and applaud their teams accomplishments.
- Parents should help with fundraising, carpools, refreshments, and other non-problem solving activities.
- Support the coach and the tournaments director. Don't "bad mouth" the coach or other teams in front of your kids. Regardless of how bad the coach may screw up. The coach is a volunteer and putting an a lot of time.
- Try to be upbeat and stress the positives, even if your child did something like just dropping a robot.

2.4.4 Core Team Members

- Do your best
- go to practice
- have fun
- cooperate with the teammates
- share your knowledge and skill with your teammates
- remember to bring snacks if it's your turn
- don't make fun of your teammates
- don't create a clique within a team
- have fun

2.5 How to Name Your Team (a team building exercise)

2.6 Fundraising Ideas (MN- will work with teams to obtain specific examples)

2.7 Sponsorship and Costs

The following cost outline is one base example. Your costs will vary according to the definition that best fits your team. If funding is an issue, look to local technology companies. With the high demand for future technology professionals, many corporations are excited to invest in FLL programs with both sponsorship and mentors. Also, solicit your parents to see if any are owners/employees in the high tech industry. Team parents can be a valuable resource for securing a sponsor. The school PTA may be willing to help with funding. Just like with youth sports, local companies are usually willing to help sponsor a team from their community. A company may pay for T-shirts if their logo is on it.

Registration fee: \$25ea. Team

Kit of Parts: \$341ea. kit/team

Team T-shirts: \$10 ea

Practice Snacks: \$1-3ea. student/meeting

Practice Field: \$50 - \$500

3 Gearing Up

3.1 Practice

How often you practice really depends on how your team is structured, if it is an after school or during school activity, or if it is with a community group or just a neighborhood. Typically teams practice twice a week for a couple of hours each time. You may want to put in some long building times occasionally like an all-day Saturday, robot-building, pizza eating, FLL-fest. Prepare for extra practices and longer practices the last few weeks.

An important role for the coach is to set milestones for the team. In other words something like "by November 1st we will have the locomotion method working".

The most important thing to remember about each practice is to have a goal or an agenda. We've all been to meetings without an agenda where nothing gets done. Each practice try to have a goal and work towards that. Your job as coach is to help the kids determine that goal and keep them working towards that during a practice. Try not to put too many goals in each practice.

Each team member should eventually have a role on the team. In the beginning you may have several sub teams building and testing but eventually you need to focus on one robot. It's difficult to have more than two or three people working on one robot so the other kids will need to find something to do. Besides bugging each other that is. Don't forget the hypothesis is an important part. Have them create some costumes or props to help tell their story. Working on the course is something that can occupy time during the first part of the year. Organizing things like creating T-shirts, working on fundraising topics, and other activities can help.

Some other tips:

- Focus on team building. Remember the four stages of a team (forming, storming, norming, and performing)
 - Help the kids get to know each other
- Have all the kids go through the tutorial so they know the basics
- Have the kids pick a role on the team. Remember this will likely change as the season goes on.
- Have them build a robot from the constructapedia and have them practice programming it.
- Help them visualize the prob the the lem
- Simplifying the rule book for them.
- Remember that the first several weeks of practice may seem very unproductive. But hopefully during this time your team will be learning basic skills, building sample robots, learning how to work together and whose skills are best suited to what activity, and generally just enjoying the process. It is not unusual for the final robot to be built during the last week.
- Chaos is normal!

3.2 First Year Team of Younger kids

The following is an example of a practice schedule for a team of young students (4th and fifth-grade) leading up to an FLL season. If you are a first-year coach or have younger students, this structure may help you. Thanks to Richard Atkins for this lesson plan. Richard is a technical professional from Honeywell working with Loring Elementary in North Minneapolis.

This lesson plan was done as an 11 week course in a K-5 elementary school. This course was designed to start with really basic building skills, and drill these skills with repeated practice. The course is highly structured, and does not get the kids into any decision making at all --- they learn **how** to build before they have to make decisions about what to build. Design and decision

making skills are worked later in the fall. Working together as a team is the biggest emphasis in every lesson, and that is why there is so much team/role structure in this.

3.2.1 General ideas

1. Assume 6 kids, 3 subteams of 2 kids each.
2. Three kinds of teams: build, program, and test. Each team has 2 kids and 2 roles. In a build team there is a builder and finder. Test team: operator, recorder. Program: typer, helper.
3. Rotate the kids with different team partners each meeting. Rotate roles during each meeting. Have the kids stay in same teams during free build.
4. Do the same agenda each meeting: setup 5 minutes, group meeting 10 minutes, lesson plan 20 minutes, free build 20 minutes, cleanup 5 minutes.
5. Each meeting do a 1-on-1 with one kid during free build time. Use that for software programming instruction.
6. The lesson plan time starts with the coach calling out the steps while the kids build it. Then they take it apart, change roles and build it again, hopefully without my help. Then we do a challenge build. [See the challenges later in this manual for ideas] The challenge build is a race. The team that completes the challenge the fastest wins a prize. I use 15 cent tootsie-pops.
7. The lesson plans concentrate on subassemblies that don't need the RCX built in. So 3 pairs of kids can each build it, and the RCX can be attached to each one in turn.
8. Treats during the group meeting. Send a note home asking parents to send treats. We want the juice boxes, not cups. Packaged snacks. Get a sign up list for each meeting day accounted for.
9. Send a flyer or leaflet home every meeting with general news about the club. Keep the parents aware of the club. It doesn't matter what we say --- just something to keep it in their minds.
10. Have the lesson plans be independent at first, with 3 build teams in parallel.
11. Later have staged lessons that gradually build a single robot, using build, program, test teams. Assume a simple task, like running out to a line and coming back. Eliminate free build time and make that part of the lesson plan during staged lessons.
12. After that, have team challenges where the team has to get the robot to accomplish some task.
13. Finish up the class with review challenges.

3.2.2 Group meeting topics (first 10 minutes of each meeting)

- 1) Explain the roles of the build team.
- 2) Show how parts are sorted.
- 3) Part sorting quiz.
- 4) Read the contract. Review club rules.
- 5) Robots. What does autonomous mean.

- 6) Torque. Power verses speed.
- 7) Explain about the FLL competition in the fall.
- 8) Explain the roles of the program and test teams.
- 9) Explain about the 3 kinds of sensors.
- 10) Explain what the RCX is.
- 11) Interview practice. Pass the robot around the circle and ask the kids questions about how it works.

3.3 Structure

As a coach or mentor it is very important to put some structure into the FLL challenge for the team members. If you are coaching a youth sports team like soccer or baseball you just don't throw ball on the field and tell the kids to play. It's the same thing with FLL, the coach needs to provide some structure and basic theory.

The previous section described some basic things to teach the kids before starting. It is important that they understand all the components of the MINDSTORMS kit. While you don't need to go into exhaustive detail here and you don't want to curtail possible creativity by saying a certain piece is only useful for a certain thing, the team members do need to feel comfortable with what all the basic pieces are (RCX, motors, sensors, and programming environment).

The biggest problem kids will have is building robots that don't fall apart. The constructapedia has some very good suggestions for building solid robots. The beams with holes in them and the black pegs can be used to build a very solid chassis.

The other biggest challenge will be programming. This challenge will be more or less depending on the coach's experience and the kids experience. It will also depend on which programming environment you use. The standard MINDSTORMS programming environment is probably more intuitive and faster to learn for kids but in the long run both RIS and RoboLab provide essentially the same capability.

Remember, you will bring your computer to the competition so you can change the program during the day. The challenge in getting kids to solve this may not be a technical challenge but just getting them together enough to spend some significant time on the problem.

Remember the engineering maxim: KISS. Keep It Simple Stupid. Build a simple robot first.

Generally FLL problems can be broken down functionally into three parts, locomotion, navigation, and robotic action. The following sections to some sample questions you can ask your team members. Let's look at each of the three parts individually.

3.3.1 Locomotion (or how do I get from here to there).

Locomotion deals with the drive mechanism and the method of transferring that energy into movement.

What are the different ways that your robot can move? Should you use wheels or tracks? Do you want a robot that is fast or slow? What effect will different size wheels have? What happens if you have a small gear driving a large gear? Or vice versa? Do you need a robot that has a lot of power? How do you get more power? Less power?

Talk about different ways to turn. For example a no radius turn means that essentially you turn in one spot, like a tank. A car, on the other hand, has a radius that it can turn in. which do you want? How many motors do you want in the drive mechanism?

Remember this is the real world. I'm sure everyone took a class in physics at some point where the problem said "ignore friction". Well we have it here. That means that everything you do will be somewhat trial and error. You can calculate things but given the nature of the programming and the physical world, some trial in error is inevitable.

3.3.2 Navigation (how do I steer this thing)

While locomotion is almost strictly a mechanical thing, navigation is much more programming.

Let's look at the different sensor types that you have:

Light sensor. Briefly describe how it works. How might it be used to help navigate? Is it reliable and accurate? Will it work the same every time?

Touch sensor. Briefly describe how it works. Same questions as for the light sensor. Remember this is the real world again. If you hit a wall at full speed with the touch sensor and expect to sense it, what will happen?

Rotation sensor. Again briefly describe how it works (it counts rotations of the axle through it). Same questions as before.

In all cases for the sensors, it may be worthwhile to write a simple one step program which has a sensor watcher on for the sensor in question. Push the view button until the little arrow points to the sensor input that you have connected and programmed. Run your program on the RCX and watch what happens on the little view screen on the RCX when you manipulate the sensor. This is a good way to visually show to the kids how the sensor works, particularly the light and rotation sensor.

How can you create a navigation program using the sensors that you have? How many sensors will you need? Which sensors will be most versatile for the different paths that you need to take? How many navigation routes do you need to program? Can you navigate without any sensors at all? (Some teams do this every year. In navigation terms this is called dead reckoning. What are the advantages or disadvantages of doing at this way?)

Without question, a robust navigation mechanism is important for a well functioning robot.

3.3.3 Robotic action (how do I move the ball, open the latch, etc.?)

In this part of the problem you must either cause your robot to perform some action like opening the latch or picking up a ball. The exact action will depend on this year's challenge. The following questions relate to the 1999 challenge where the robot had to open the latch and transport balls.

How will you grab the latch? What are different ways for moving this latch? Do you need to grab it? How will you detect when to try to move it?

How will you transport the balls? How will you move the ball from your robot into the chamber? Will you grab the balls? How? Is there another way to move them without grabbing them? How will you determine when to move the ball from your robot to the chamber?

3.3.4 A few more general tips.

Look on the Internet for examples! This is not cheating, this is looking for models/examples/inspiration.

Look to the real world for examples. Look for real-life machines that do some of these things. How do they work?

How many programs will you need to implement your solution?

Will you need to add or remove parts between missions?

How will you use your three motors? Do you need them all?

If you are a technical professional, just treat this project as you would something at work, divide and conquer.

If you are a teacher, just think of each one of the three functional areas of the problem as a

different lesson plan where you are doing some inquiry-based learning.

Also in each case, remember the problem solving process: What's the problem?; Brainstorm solutions; Evaluate the solutions; Pick one and build it; Evaluate.

3.3.5 Solve the Problem a Little at a Time

The first reaction of most kids is probably to build a robot to solve everything and then try to program it. This "big bang" approach rarely works well.

A better approach is to solve each problem independently. Prototype a robotic arm (or whatever mechanism you choose) by building it and writing a simple program to test it. This is how products are designed and built. The reason is that it is much easier, and cheaper, to find the problems that inevitably occur early. In scientific terms you reduce the number of variables you are dealing with. By simply dealing with the robotic arm and the program to control it, you don't have other things interfering like navigation, etc. You can write a simple program to test what you're trying to do.

There are lots of statistics and papers in the engineering world that document the order-of-magnitude cost increase to fix an error each time you progress a step through the product life-cycle (requirements, design, build and test, manufacture, field support). Every day in the newspaper it seems, there are reports of some type of security bug or other problem with some released software by Microsoft. It is much easier to fix those problems earlier. The way to do that is to build the project in pieces and verify each piece by itself. Once you know that piece is working, you can integrate it with another piece. The same concept is true for FLL.

You can break your kids up in different ways. Maybe you could work on each area one at a time. Or maybe you could break the kids up to work on different areas.

3.4 Outcomes

Good technology development consists of good problem solving and good decision-making. Problem solving has been discussed earlier. Learning good decision-making is also important. Good decision-making means understanding what the criteria is upon which to make your decision. [More on this later]

Comments from coaches:

“Anyhow, from the competition we learned one very important lesson (no offense to anyone)- don't worry about implementing the "ideal" solution- just get the job done!”

“I think there's a trap if you let the kids try to chase some sort of

elegant solution -- they spend so much time on design and construction that they don't have the time to test and practice before the contest. Also, the kids have less input as a team if you freeze the design too early. I pushed the kids to get something (anything!) built ASAP that moved reliably under its own power. Then we worked on the missions, modifying the device to do what the missions required. This gave everyone a chance to make observations about how to make it work better, though a few leading designers did indeed emerge.”

{these need to be categorized yet”
“Gears vs Pulleys:

I think it's easier to build a robust and accurate device using gears. If the kids match the gear sizes correctly, the whole assembly snaps together tightly. When the motor turns a gear to turn a wheel, there's no risk of inaccuracies due to slippage. There's also no risk of elastics pulling the device apart when the kids put them on.

Straight Lines:

I think there are two crucial tricks behind straight lines. First, you must build the whole assembly of wheels, gears, and motors as rigidly as possible. Slack, backlash, and loose parts just won't work. Don't use pulleys because they aren't as rigid as gears, and any slippage will throw it out of line. Second, use exactly the same gears to drive both wheels. If you do these right, the robot should run straight and true when you run the built-in RCX program #1. The program runs both motors at the same speed and in the same direction. If the wheels and gears are matched and tight, then the robot runs straight.

Our team played a variant of the "hot potato" game that tested this: the kids sat in a large circle, one kid would aim the robot at a "gateway" another kid made with hands or feet, and try to run the robot through the gateway. The other kid would intercept the robot, turn it around, and send it off to another kid. This gave the kids experience with the moving robot and (important) experience dealing with parts coming loose or falling off.

Measuring Motion with the Rotational Sensor:

One of our team members, Alex, had taken several Lego Logo classes at the

Science Museum, and had soaked up one of their cardinal rules -- there should always be a gear or pulley connecting a motor to a wheel. I don't know the rationale for this, and the Constructopedia suggests that this isn't true for the RCX. Alex insisted, and the resulting robot worked well.

As a practical matter, you need to drive your wheels with gears or pulleys if you are using the rotational sensor to measure your motion. To get the best accuracy, the rotational sensor should rotate in conjunction with one of the driving wheels. Then the motion of the wheel on the ground is fed directly into the rotational sensor. It's less accurate to measure motion any other way.

My own bias is in favor of gears, since they yield tighter and more reliable devices than pulleys.

Precise, accurate turns:

Our robot made sharp 90 degree turns by pivoting around a center point located between the two powered wheels. This was the easiest and most reliable way to do it. The robot would essentially rotate in place when we ran the two motors at the same speed but in opposite directions.

We made sharp 90 degree turns by counting rotations using the rotational sensor, which was geared to one of the driving wheels. It took some trial and error to find the right number of rotations for various turns (we used specialized turns to grab the escape latch) but the sensor provided the degree of accuracy we needed.

Another trick - we used rubber tires on the wheels attached to the motor, but did *not* use rubber on the other wheels. The rubber tires ensured accurate motion by the powered wheels, and the omitted rubber prevented the non-powered wheels from causing too much drag when the robot turned.

This worked well enough to earn us fifth place at State, though it didn't get us into the final four. We noticed that two of the top scoring robots (the ones we would have had to beat) didn't make much use of 90 degree turns.

Rick.
Hastings Wizards”

3.5 Profile of 1st 5 Practices

Here is an example of the first five meetings to help you get started.

FIRST MEETING - with parents and students

- Meet with parents and students to fully explain the program & types of problems, EMPHASIZE that being on an FLL team means a commitment from both parents and team members!
- Review general rules and FLL Philosophy
- Discuss students and parents goals and expectations. Why do you want to be an FLL'er? is a good discussion starter. This is a good time to discuss the difference between 'winning' and 'succeeding'...you may find yourself repeating this at every meeting!
- Ask Everyone what it means to be on a team.
- Have team members and parents read & sign a contract
- Point out parental responsibilities and restrictions
- Ask for parent volunteers to help with transportation, course building, refreshments, or as resource people.

SECOND MEETING - (team members only from this point on)

- Discuss the pros & cons of working as a group
- Explain how all ideas must be considered and not judged
- Have team read and discuss the simplified version of the challenge
- Make a list of special skills and strengths of the team. Decide what skills must be learned to solve the various problem.
- Talk about the roles and what it means to be on a team.
- Have some kids do the Mindstorms CD-ROM tutorial.
- Have some kids build something from the Constructopedia.

THIRD MEETING - *Note...new FLL'ers might need more meetings to reach this point...or less!*

- Have some kids do the Mindstorms CD-ROM tutorial.
- Have some kids build something from the Constructopedia.
- Encourage kids to visualize their solutions & to compromise their ideas with others
- Determine when and how you will build the course.
- Try programming some of the robots you build from the Constructopedia.

FOURTH MEETING -

- Brainstorm a list of tasks necessary to complete solution
- Brainstorm ideas for your hypothesis
- Coach and team should devise a timeline for task completion
- Set up a schedule of field trips, shopping trips and necessary research to be done
- Practice programming and building.
- Make a master calendar of your timeline, tasks, and field trips, assign a team member(s) to each task

FIFTH MEETING -

- Begin accomplishing tasks
- Practice programming and building.

4 Specific Kit Components/Lessons

4.1 Engineering Basics

4.1.1 RCX

4.1.2 Software

Back in the dark ages of programming they taught programmers to use flow charts. When planning out your program, use a couple methods to help your kids visualize this. Have one of the kids pretend he/she is the robot. Have them walk through the course. What do they do each step at a time? What program step should I use to do this? Then use a simple flow chart or something to write it on the board. Kids may have hard time making the connection from the board to the computer, so you may want to go right to the computer.

It seems like a very large percentage, maybe even more than 50 percent, of the teams are using RoboLab. If you are an adult geek, you will probably find RoboLab an easier environment to use since it is more like a traditional programming language. It does have more features than RIS but there really is not any advantage that it will give for this problem (this was borne out in 1999 results where very good performing robots were done with each programming environment). The standard RIS programming environment takes a lot of knocks from adult programmers, but in reality it is a pretty sophisticated environment made to seem pretty simple. It does some funky things with multi-tasking but the kids seem to pick it up pretty quickly.

If you are using RoboLab, you'll have to go to at least inventor level three to get the same functionality as the standard RIS programming environment. What you get when you go to level four inventor is what they call "containers". Variables to the rest of us. This ability to save state, brings RoboLab to another level that the standard RIS really can't get to. For complicated problem this is a great advantage however for this problem you really can solve it easily without using this feature. There are certainly ways to work it into the program but that's a topic for another day.

4.1.3 RIS

4.1.4 RoboLab

4.1.5 RoboLab vs. RIS

A couple of differences between the environments: RoboLab grades power from 1-5, and the RIS environment from 1-8. However the table on page 42 of the RoboLab manual shows the

conversion.

You can add comments and print using the RoboLab environment, both useful features.

RoboLab uses something called jumps, rather than the loop type structures the RIS environment uses. You can do the same things so there is no advantage to either one except the RIS method is probably clearer to kids.

RoboLab allows you to use multiple threads like the RIS environment (you connect the sensor watchers on to the regular program) except you use forks/splits in RoboLab. Again no advantage either way. To a programmer the RIS environment seems odd in the way that it does it, but the kids seem to grasp it right away.

The RIS environment does not allow myprograms in myprograms. Only one level. When you get something working like a left turn, make it a my program.

The RIS environment does not allow a loop (red blocks) in a loop. Should not be a problem.

4.1.6 Constructopedia and other documents

4.2 Sensors

4.2.1 Touch

4.2.2 Light

4.2.3 Rotational

4.2.4 Temperature

4.3 Motors

4.3.1 Mounting

4.3.2 Drive Systems/Linkages

4.3.3 Gears

4.3.4 Pulleys

4.3.5 Shafts

4.4 Wheels/Tracks

4.5 Wiring

4.6 Programming

4.6.1 90-degree Turn

4.6.2 others

4.7 Mini Challenges

There are certain basic skills you can develop now, that will come in handy when you are trying to conquer the FLL Challenge. You will learn the basics and more, if you can get a robot through the following mini challenges. The effort will allow you more time for higher-level activity after the FLL Challenge is unveiled.

{Thanks to Richard Atkins for these first 2 sections }

4.7.1 Independent Lesson Plans (one for each meeting, unconnected subassembly builds)

1. Explain the 4 kinds of pegs and their uses. Build a right angle with beams. Have one team build with gray pegs and two teams build with black pegs and compare to see that the black pegs are stronger.
2. Build a motor cage.
3. Build a plate sandwich assembly held together with axles.
4. Build a gear rotation reduction.
5. Build a touch sensor assembly and activate the motor using it. Use program one of the kids has written during free build time.

4.7.2 Staged Lessons (progressively building one robot)

1. Build 2 motor cages, one around each motor. Attach them together. Add wheels. Write program to turn on both wheels.
2. Add 2 more motor cages to wheels. Attach chassis to motor cages. Attach RCX. Add plate assembly in front. Use existing program and test it forward. Add touch sensor to front plate assembly, but without the assembly around it.
3. Build touch sensor assembly around touch sensor on front plate. Add light sensor. Add rotation sensor. Test it going out, touch stop, reverse back. Test it out 50 clicks and back. Test it out to a line and back.

4.7.3 General Challenges

{Thanks to Scott Evans for these }

1. Go forward and stop over a piece of black tape 24 inches away using
 - A) The internal timer
 - B) The rotation sensor
 - C) The light sensor
2. Follow a black line
3. Go forward about 24 inches, turn around, come back, and stop at the start
4. Repeat mini challenge 3, but end up in the same exact place five times in a row
5. Go forward and reverse direction when the touch sensor is pressed
6. Find a black line and follow it
7. Climb over a physics textbook (a math text will work, but probably not an English text)
8. Go forward, grab a 2 inch ball of tissue, come back, and drop the tissue

{Thanks to Ted Cochran for these }

Engineering:

How tall a robot can you make that can drive without falling over?

Can you make a robot that can drive into a wall and not break?

How far can pieces of the robot be away from its wheels without having it tip over?

Climbing:

Investigate how steep a ramp you can climb with various gear ratios.

--what difference does the surface make?

--what about how clean the wheels are?

--what about how BIG the wheels are?

--what about how heavy the robot is.

Add a bunch of parts, so you can see whether an arm with a motor would weigh enough to make a difference.

If you have to climb a ramp, how much will you have to worry about ground clearance?

Navigating:

Go as straight as you can for as long as you can.

Now do it while,

following a line on the ground

following a wall (using a touch sensor or just a bogey wheel)

Going exactly an arbitrary distance.

Make a right angle turn.

Make a square, and stop at the origin.

Make the same size square going half as fast.

Make a square of an arbitrary size.

Can the rotation sensor be used to make the turn more precise?

Do it all again, but this time with a hexagon.

Investigate whether a steering wheel might be better than differential turning of wheels.

Investigate whether, if you use differential turning of wheels, the third support should be a pivoting wheel or a skid. In front or in back?

Investigate how fast you can drive, using the touch sensor to stop when you reach a wall, without crashing into the wall too hard.

Does it matter what control structure the program uses to recognize the touch and stop the motors?

Does it help if the sensor is rigidly mounted? What if it is not rigidly mounted (for example, if it is put on the end of a flex tube?)

Manipulating:

Make an arm that can hold something and let go of it.

Will your design work for blocks as well as fast balls?

Will it hold the object while the robot moves, turns, and backs up?

How long an arm can you make and have move without tipping the robot over?

Programming:

How many ways are there to pause for a certain length of time?

Sure, there is the watch--will it work for tenths of seconds?

What if you played musical notes instead of using the watch?

How repeatable are the different timing methods?

Are they the same if they're on a program branch?

If the motors are running?

If the batteries are tired?

4.7.4 Common Problems

4.7.4.1 Robot Won't Drive Straight

Make sure the playing field is level. Make sure the robot is tight by squeezing it, not leaning on it. Make sure the vehicle is built symmetrically, right to left. If the drive system has rubber bands, switch them around and see if the robot behaves differently. If nothing changes, make sure all axles rotate freely. If an axle does not rotate freely, make sure the holes it spins in are perfectly aligned. Make sure it's straight. Make sure it's clean. Make sure it has a little bit of side-to-side play. If there's no side-to-side play, make sure the elements on it are not pinching whatever they spin next to. If all of the axles spin freely, take the motors completely off of the robot. Put the smallest gear on each motor. Run the motors in opposite directions and mesh the gears by hand. If the motors start revolving around each other, switch each one with the third motor to seek a speed match. If no two of your motors are a speed match, call PITSCO LEGO Dacta for at least one replacement motor. You can also redesign your drive system to work on a single motor, or hook up both motors through a single linkage.