FRC Project*Libre*™[1] Tutorial

Using Project*Libre* for Your Robot Build

Craig Yankes
Western PA *FIRST* Regional Planning Committee
cyankes@msa.com

[1] Project*Libre* is a trademark of Marc O'Brien and Laurent Chrettieneau.

Contents

Introduction

General Comments

Congratulations! Reading this shows that you are interested in using project management software to help with your team's robot build. While using this type of software to help manage an 8-week project might seem excessive, I believe you will find that this will help you deal with the hectic build period by quickly showing where your team is in the build process and whether it is behind, on, or ahead of schedule. Quite importantly, if your team is behind schedule a tool like this can help identify where the team needs to concentrate its effort to achieve the goal of taking a quality robot to Regionals and, possibly, further on to Nationals.

Two comments before we get into the "meat" of the subject. First, in writing this tutorial I am presuming that you have either attended the "Project Planning" session at the 2013 SCRA[2] Workshop or have read the annotated slides from that session that are posted with this document. The slides cover the introductory material explaining why a team can benefit from using project-planning techniques, introduces some of the terminology that is used and describes some common pitfalls that hopefully you will be able to avoid by being aware of them. While the terminology will be defined in this tutorial, the "whys" and "oops" in the slides will not be recreated here. If you haven't seen them yet, I suggest looking over the slides before continuing with this tutorial.

The second comment is one of thanks. In the spring 2013 robot build, Team 3511 used an Excel[3] spreadsheet to view their progress and word of that spreadsheet led to the Project Planning session being requested at the SCRA Workshop. Jim Broker, Team 3511's mentor, has given permission for that spreadsheet to be used in this tutorial as example tasks or as templates as desired. Thank you, Jim!

Scope of this Tutorial

It is impossible in a short tutorial to introduce you to every aspect of project management software whether represented in a free tool like Project*Libre* or commercial tools like Microsoft Project[4]. Therefore, not only is this not going to be a comprehensive user's manual describing every option and feature in Project*Libre*, the scope of this tutorial is being purposely limited to the smallest set of features you'll likely need to manage the robot build. In short, the goal of this tutorial is not to impress you with the tool, but rather to help you use this tool in your build. If you are intrigued by this software, there is plenty of time after the build and competitions are over to explore it more thoroughly.

Resources for more Information or Help

There are several resources available to help you learn and use the tool. There is an online help facility in the tool accessible by clicking on the "?" in the upper right corner. There is also a large user community

[2] "SCRA" is the Steel City Robotics Alliance. If you are not familiar with it, I highly recommend that you visit www.steelcityrobotics.org. Among other events, each year they sponsor a workshop for teams to hone their skills or learn about new topics.
[3] Excel is a trademark of the Microsoft Corporation.
[4] Microsoft Project is a trademark of the Microsoft Corporation.

on the www.projectlibre.org website that might be useful for you to join. Given the short build time, though, many questions will be time-urgent and so while the author of this document is by no means an expert in every piece of project management tools, he is willing to help where possible. You can reach him by sending an email to cyankes@msa.com. To help have your email stand out from others, please put "ProjectLibre" in the subject field.

As stated earlier, this tutorial just scratches the surface of what Project*Libre* can do. Amazon has books (paper or Kindle) available on Project*Libre* including a user manual, a tutorial, and other resources. (These books will teach you areas likely beyond what you'll need for the robot build. As I suggested in the Workshop, given the size of the build teams and the length of the tasks you can probably skip the added complexity of the "resources" concept of assigning individuals to tasks.)

Getting Started

Project*Libre* was so simple to install on my machine that I will presume that you have done that and are ready to run it. When you first run it, you will see a license popup. While this is not a recommendation either way, I will presume that you have accepted the license and want to continue with the tutorial.

Creating a Project

Closing the "tip of the day" popup will bring you to another popup asking if you would like to open an existing project or create a new project. Select "Create Project."

The "New Project" box is straightforward. You can name the project anything you would like and "sample project" works well for this tutorial. (Note that if you later merely "save" the project, the filename will default to your project's name with a .pod extension. You can, of course, "save as" to give the project file a specific name.) The manager and notes fields are optional and I'll explain the "forward scheduled" option later (but for now just leave it selected). The project's "start date" field will default to today's date, but to allow your sample project to follow this tutorial exactly, change the start date to July 1, 2014. As with the other date fields, you can set this value by either just typing "7/1/14" in the box or you can click the down arrow on the right side of the box which will cause a calendar to appear where you can navigate to the desired month and just click on the desired date. Whichever way you chose, set it to July 1, 2014 and click "ok".

Normal Display

What you now see is the normal task list / Gantt chart display screen where you will spend most of your time. Since you haven't defined any tasks yet, both the task list on the left and the Gantt chart on the right are empty. Before we add any tasks, let me warn you that it is easy to get to other screens where it isn't obvious how to get back to the normal display. For example, in the upper left section you currently see various file options (save, open, etc.), print options and project options. Click on "Projects" in the Project section. (You will see one line for your sample project.) It is not obvious how to get back to the normal display, right? Even the back arrow to the right of the Project*Libre* logo does not get you back! Getting

back to the normal display is easy once you know how to do it. Click the "View" tab and on the far left you'll see the Gantt icon and word. Click on that and you are back to the (currently blank) normal display for your work. Without clicking on anything other than the toolbar tabs "File", "Task", "Resource" and "View", take a moment and look at the options that appear under the tabs. There is a lot there, but don't worry as there are only a few options that you'll need to know for the robot build.

Navigating in this Tutorial

As you looked through the tabs did you notice that the options are grouped into sections? In the previous paragraph I asked you to "click the 'View' tab and on the far left you'll see the Gantt icon and word. Click on that…" That is rather wordy. From here on I'll shorten this to tab / section / option and so moving back to the normal display is done by clicking **View | Task views | Gantt**. If the option I'm asking you to click is currently displayed (i.e. there is no need to switch to a different tab), I'll specify the option by just saying **section | option**. Therefore, from where you are now at if I wanted you to click on Resources, I'll ask you to click on **Resource views | Resources**. Go ahead and click on that. Now get back to our normal display.

Defining Your Project

Adding Tasks

While I urged teams during the Workshop presentation and in the slides not to have individual tasks longer than one week long, for this tutorial I'm going to ignore that suggestion for one simple reason: Having nine tasks is easier for learning a tool than is having perhaps 30 to 40 real tasks during your build. Here are the hypothetical tasks and their estimated duration (length of time to complete the task) we'll be using:

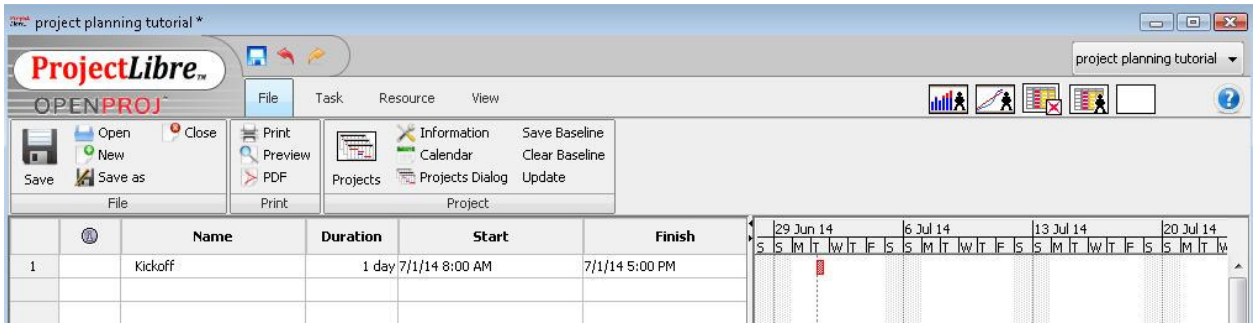
- Kickoff, 1 day duration
- Consider designs, 8 days
- Identify common design features, 1 day
- Decide final design, 1 day
- Build common features, 15 days
- Build design-specific features, 20 days
- Programming, 15 days
- Testing and rework, 10 days
- Practice, 10 days

I'm sure many of you are cringing at these tasks and the length estimates. Good! If so, you realize that these are crude and unrealistic. However, they work for the tutorial. (An actual Gantt chart used by Team 3511 in the 2013 build is included at the end of this document for an example of a real set of tasks.)

We'll now enter the first task. In the top line under the Name column, enter "Kickoff" and press <tab>. Notice how the focus has now switched to the Duration column which was automatically filled in with "1 day?" We want the Kickoff to have a 1 day duration, but since the "?" means it is an estimate let's make

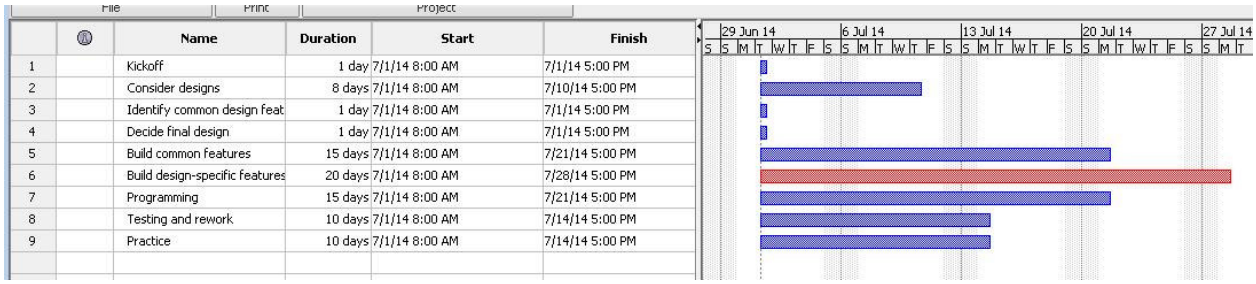
it a definite 1 day. Do this by typing "1d" into Kickoff line's duration box. (Project*Libre* lets you enter the number and the first character of the time, such as hour, day, week, etc., if you don't want to spell it all out.) Note that the Start and Finish boxes contain 7/1/14 8:00 AM and 7/1/14 5:00 PM respectively. Project*Libre* defaults to a traditional work week of 8am-5pm Monday-Friday. You will be shown later how to change this to better match your team's workday schedule.

At this point, you should see the following on the screen:



If this is not what you have, verify that you correctly did the steps in the previous paragraph and make sure that when you started this project that you specified the start date as being July 1, 2014 so that our displays will match.

On the lines below the Kickoff line (which now has a "1" in the leftmost column showing this is task number 1), enter the eight remaining hypothetical task names and durations from our example. Once done with this, your screen (skipping the header section from here on) should look like this:



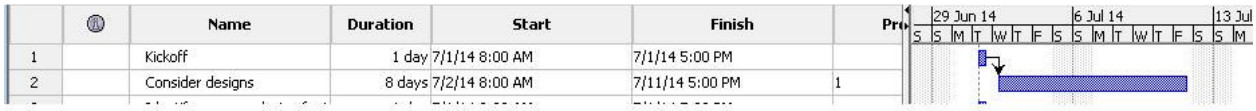
Notice that the "Build design-specific features" line is the only red line. As a reminder, the norm for Gantt charts (the right side of the display) is for the red line to represent the "critical path" meaning that this set of tasks, although just a single task in the example for now, represents the tasks that will ultimately determine when the project will end. Right now, since all of these tasks are scheduled to begin on the same day it is not hard to see that the longest running task (20 days long) will determine when the project will finish. It is not, of course, realistic for all of these to start on the same day so in the next section we'll add some dependencies between these tasks.

Adding Dependencies

In project planning, a "dependency" simply means that something must happen before something else can happen. If you are going to a store alone by driving, it is intuitive that you must complete the task "walk to the car" before you can begin the task of "drive to the store." We would then say that "drive to the

store" is dependent upon the completion of "walk to the car." As described in the Project Planning slides, task dependencies aren't just of the "start this when that finishes" form, but can also be "start this when that starts" or even "start this X number of days before or after that starts or finishes".[5]

Let's consider the dependency between our first two tasks; "kickoff" and "consider designs." Since it is at the kickoff that the teams learn about that year's competition, it is reasonable to say that the team can't start considering designs until the kickoff is over. To enter this into Project*Libre*, scroll the left half of the screen to the right so that you see the "Predecessors" column. We want to say that the second task cannot start until the first task has completed, so just enter "1" under Predecessors for line 2. Observe that the first part of the display has now changed to:



Note the changes: The Gantt chart line for the second task has shifted to the right by one day, there is now an arrow pointing from the first line to the second and that the second task's Start date has changed. This is the result of telling Project*Libre* that the second task can't begin until the first task has completed.

In this hypothetical example, the team is going to identify the common design features among all their design options so that they can start building those common elements early. Since we have 8 days for considering designs, let's say that these common design features will be identifiable three days before all the design considerations are done. Following the examples from the slides, the dependency for line 3 will then be "2fs-3" which means that this task can start three days before task 2 finishes. Now, could we say this dependency is "2ss+5" (meaning it can start 5 days after task 2 starts)? Of course. Deciding which way to specify the dependency is easy if you ask this question: "What should happen to this task (#3) if the other task (#2) doesn't finish on time?" If "this task" should start on time anyway, then the best way of expressing the dependency would be the "2ss+5" since even if task 2 is delayed, five days after it starts task 3 can start anyway. If, on the other hand, "this task" has to delay starting if the other task is delayed, then it would make sense to express the dependency as "2fs-3" so that if task 2 is delayed, task 3 will be delayed along with it so it still doesn't start until 3 days before task 2 has finished.

We'll use both forms in this tutorial so you can see where the different approaches make sense. In the case of tasks 2 and 3, though, we'll answer the "what should happen…" question by saying that even if the design consideration takes a few days more than planned, five days into the design consideration we should start identifying the common features. Therefore, this dependency will be "2ss+5" since the important thing keeping task 3 from starting is task 2 being worked on for at least 5 days. Enter the dependency into line 3.

As when we entered the dependency for task 2, the Gantt chart has shifted around to reflect the new timing of these tasks. Notice one thing different in lines 1 through 3, though: The arrow comes from the start of task 2's box instead of out the end of task 1's box. This visually shows whether the dependency is

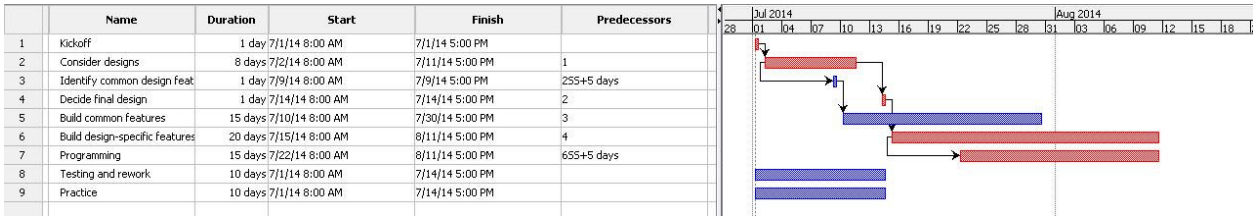
[5] Also as mentioned in the slides, tasks can be set to finish based on when other tasks start or finish, but this is an unusual way of viewing task dependencies and is likely not needed in something as straightforward as the robot build. Therefore, it won't be used in this introductory tutorial.

a finish dependency (such as from task 1 to task 2 with the arrow coming from the end of task 1's box) or whether it is a start dependency (such as from task 2 to task 3). In either case, the horizontal length of the arrow indicates how many days have to elapse before the next task can begin.

To give you some more practice at this, enter the dependencies for the following tasks:

- Decide final design: Requires "consider designs" to complete.
- Build common features: Requires "Identify common design features" to complete.
- Build design-specific features: Requires "Decide final design" to complete.
- Programming: Can be started 5 days after "Build design-specific features" has started.

Before showing you what my screen now looks like for comparison, notice that the project has grown in length and now exceeds what (likely) conveniently fits in the Gantt chart section on the right. You can zoom in and zoom out on the schedule by pressing **Views | Zoom In** and **Views | Zoom Out** respectively. Zoom out one level and see if your screen now looks like this:



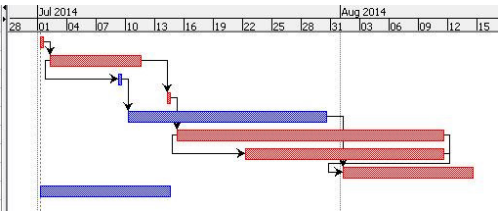
	Name	Duration	Start	Finish	Predecessors
1	Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM	
2	Consider designs	8 days	7/2/14 8:00 AM	7/11/14 5:00 PM	1
3	Identify common design feat	1 day	7/9/14 8:00 AM	7/9/14 5:00 PM	2SS+5 days
4	Decide final design	1 day	7/14/14 8:00 AM	7/14/14 5:00 PM	2
5	Build common features	15 days	7/10/14 8:00 AM	7/30/14 5:00 PM	3
6	Build design-specific features	20 days	7/15/14 8:00 AM	8/11/14 5:00 PM	4
7	Programming	15 days	7/22/14 8:00 AM	8/11/14 5:00 PM	6SS+5 days
8	Testing and rework	10 days	7/1/14 8:00 AM	7/14/14 5:00 PM	
9	Practice	10 days	7/1/14 8:00 AM	7/14/14 5:00 PM	

(Note that I've grabbed the line between the data and the Gantt chart sections and moved it to the right to show more data columns. You can do this by left clicking on the vertical line between these sections and sliding it right or left.)

The reason I had you stop at "Programming" was that this task and all the ones before it had only a single dependency. This is not always the case since sometimes multiple tasks have to reach certain points before a new task can start. Consider our next "Testing and rework" task. Imagine that the team has decided that it can start testing seven days before the basic robot build completes. The complication here is that given this overly high-level "testing and rework" task[6], its start date is controlled by three other tasks; the two build tasks and the programming task. Thankfully, it is easy to describe this kind of dependency on multiple tasks: You simply list each individual dependency and separate them by semicolons. The key here is to just take them one at a time. First, the testing task can't start until seven days before the "build common features" task is done, so this is a 5fs-7. It can't start until seven days before the "build design-specific features" task is done, so this is a 6fs-7 and, finally, it can't start until seven days before the "programming" task is done which is 7fs-7. Put them together and you get the mouthful "5fs-7;6fs-7;7fs-7". Enter this as the predecessor for task 7 and look at what happens to the Gantt chart:

[6] I did this on purpose for the tutorial. In reality, the "build", "programming" and "testing / rework" lines should be at a lower level so that individual features can be tested as they are done.

	Name	Duration	Start	Finish	Predecessors
1	Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM	
2	Consider designs	8 days	7/2/14 8:00 AM	7/11/14 5:00 PM	1
3	Identify common design feat	1 day	7/9/14 8:00 AM	7/9/14 5:00 PM	2SS+5 days
4	Decide final design	1 day	7/14/14 8:00 AM	7/14/14 5:00 PM	2
5	Build common features	15 days	7/10/14 8:00 AM	7/30/14 5:00 PM	3
6	Build design-specific features	20 days	7/15/14 8:00 AM	8/11/14 5:00 PM	4
7	Programming	15 days	7/22/14 8:00 AM	8/11/14 5:00 PM	6SS+5 days
8	Testing and rework	10 days	8/1/14 8:00 AM	8/14/14 5:00 PM	5FS-7 days;6FS-7 days;7FS-7 days
9	Practice	10 days	7/1/14 8:00 AM	7/14/14 5:00 PM	

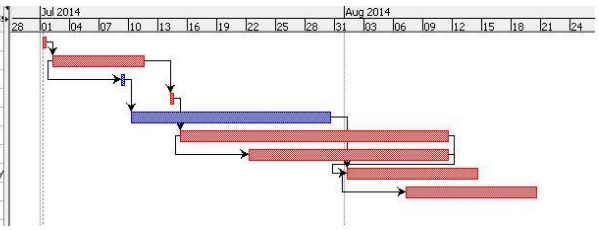


See the multiple arrows that now feed into the start of the Testing box? This shows that the timing of when this task can begin is gated by multiple tasks. Now, why go through the hassle of defining three tasks as predecessors for the testing task when we can "obviously" (sic) see that just having testing start seven days before the programming ends will give us the same date? Be careful not to fall into this trap. Just because at project planning time it is "obvious" when things end at the same time, the reality of the project can delay tasks. For example, what happens if "programming" goes well and finishes early while the "build design-specific features" task takes three days longer than expected? If you just have this generic testing task gated by the completion of the programming task, the testing could end up starting too early compared to the hardware building task. While defining all the relationships for a task like this Testing task is a few minutes longer, it lets the schedule correctly adapt to any of predecessor tasks finishing either early or late. (Project planning tools will determine the last date of each of the dependencies and will use that as the start date for the task.)

Now, did you catch that I jumped right into "fs" types of relationships here? While I didn't raise the issue in the last paragraph since it was focused on one versus multiple predecessor tasks, remember the question introduced earlier about "What should happen to this task if the other task doesn't finish on time?" We could have defined the Testing task as beginning some number of days after each of the three build / programming tasks started, but testing is a classic case where it normally makes sense to delay starting the testing if the thing(s) being tested have a schedule slip. Therefore, testing is a common task type gated by the completion of other tasks and not the beginning.

To complete the dependencies in this example, set the Practice task to begin four days after Testing has started. The final initial schedule should look like this:

	Name	Duration	Start	Finish	Predeces
1	Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM	
2	Consider designs	8 days	7/2/14 8:00 AM	7/11/14 5:00 PM	1
3	Identify common design feat	1 day	7/9/14 8:00 AM	7/9/14 5:00 PM	2SS+5 days
4	Decide final design	1 day	7/14/14 8:00 AM	7/14/14 5:00 PM	2
5	Build common features	15 days	7/10/14 8:00 AM	7/30/14 5:00 PM	3
6	Build design-specific features	20 days	7/15/14 8:00 AM	8/11/14 5:00 PM	4
7	Programming	15 days	7/22/14 8:00 AM	8/11/14 5:00 PM	6SS+5 days
8	Testing and rework	10 days	8/1/14 8:00 AM	8/14/14 5:00 PM	5FS-7 days;6FS-7 day
9	Practice	10 days	8/7/14 8:00 AM	8/20/14 5:00 PM	8SS+4 days



Setting Work Days

As mentioned earlier, most project planning tools default to the normal workweek. Let's say that your team works on the robot Monday through Thursday of each week and so we want Fridays to not be counted as a workday. Adjusting the schedule of workdays is easy. Click on **Task | Calendar** and the "Change Working Calendar" box will appear. Scroll to the July 2014 month calendar and click on the "F" in the day of week line. See that all the Fridays have been highlighted (on this and all the other

calendar pages whether shown or not). On the left side, click "Non-working time" and then "OK" at the bottom. Take a look at the ending date of the project. Now, instead of it ending at the end of the week of August 18, the project will now extend into September and if you zoom in (**Views | Zoom In**), you'll see that Friday, Saturday and Sunday are now all grayed out to show that they aren't work days.

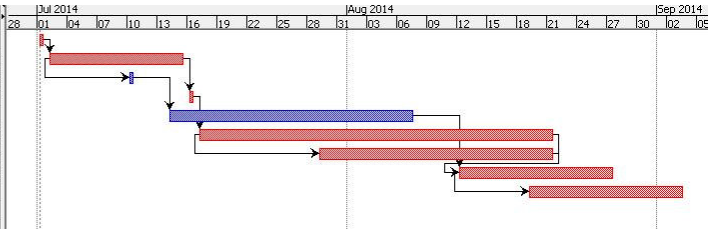
Let's now look at how to deal with specific days off. July 4th is a day that schools are closed, so will we have to adjust the schedule to show that as a non-working day? Click **Task | Calendar** again to check which day of the week July 4th 2014 is on. It is on a Friday which is already a day off for our hypothetical team, so we don't have to tell Project*Libre* to ignore that day. That's boring so let's pretend that the schools are not in session on July 21, 2014 for some reason. In the "Change Working Calendar" box, click on just July 21, 2014 and click non-working time and then OK. The Gantt chart will now show that to be a non-working day and will shift everything after it to the right one (work) day.

If you want to change a non-working day to being a working day, select the day as above and click either "use default" or "non-default working time." The difference between these is whether the day is one of your normal workdays or not. If it is a normal workday, clicking "use default" will return it back to normal. If it isn't a normal work day (a Saturday, for example), click "non-default working time". Either way, one of these will turn the non-workday from gray to white background and that is the one you want to use.

Using Critical Paths

After entering your real project's tasks and their dependencies, you might find that the project's ending date is later than when it has to complete. Pretend that the example project we've created has to be completed before September 1st. The Gantt chart "critical path" identification (the red lines) helps you to see which tasks to try to shorten. Here is what the project should look like in your display:

	Name	Duration	Start	Finish
1	Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM
2	Consider designs	8 days	7/2/14 8:00 AM	7/15/14 5:00 PM
3	Identify common design feat	1 day	7/10/14 8:00 AM	7/10/14 5:00 PM
4	Decide final design	1 day	7/16/14 8:00 AM	7/16/14 5:00 PM
5	Build common features	15 days	7/14/14 8:00 AM	8/7/14 5:00 PM
6	Build design-specific features	20 days	7/17/14 8:00 AM	8/21/14 5:00 PM
7	Programming	15 days	7/29/14 8:00 AM	8/21/14 5:00 PM
8	Testing and rework	10 days	8/12/14 8:00 AM	8/27/14 5:00 PM
9	Practice	10 days	8/19/14 8:00 AM	9/3/14 5:00 PM



See how the "build common features" task line is blue? The reason it is blue is that the task gated by its completion (the testing task) also requires other, later completing, tasks to complete. To show this, change the duration of the "build common features" from 15 days to 12 days and look at the project's end date. See how the end date didn't change at all? *While shortening tasks that are not on the critical path might free up people to work on other tasks, shortening it alone will not change the project's completion date.* (Please return "build common features" to 15 days duration now.) If you need to shorten the schedule, concentrate on the red "critical path" tasks.

While you are shortening task durations, don't be surprised by seeing the critical / non-critical (i.e. red / blue) status of the lines change. As an example of this, change the duration of "build design-specific features" (line 6) from 20 days to 18 days. Doing this shortened a critical path task yet didn't change the end date and, furthermore, the line turned blue. The reason this happened is that while it is no longer a critical task item (by gating when the testing task could start), another task also gates the testing task and

so the project's end date still didn't change. (Note that right now all of the tasks prior to the build task might be blue. I have seen a bug in Project*Libre* that occasionally when a red task becomes blue it accidentally, and incorrectly, sometimes paints the earlier tasks blue as well. If this happens, you can force it to recalculate all the colors by taking the last task, shorting it by a day and then putting it back to where it was.) At this point set the duration of line 6 back to 20 days and save your project file (**File | File | Save as**). Now that you can easily get back to where we currently are at in the tutorial (**File | File | Open**), feel free to play with changing durations of the various tasks to see how the critical path line moves around and how the example project's end date changes.

Once you are done experimenting with how duration changes affects the critical path, please Open the file you saved (**File | File | Open**) so that our displays are once again consistent.

Tracking the Project

What we have done thus far in the tutorial is create a project estimating how long each task will take. While it is important for planning to do this with the best estimates you can give the tasks, the estimates will rarely turn out to be perfect.[7] Some tasks will complete faster than anticipated while others will take longer. Dealing with this changing reality is one of the main benefits of using project planning software. In this section, we'll explore how to update the project's progress.

Task Completion Percentage

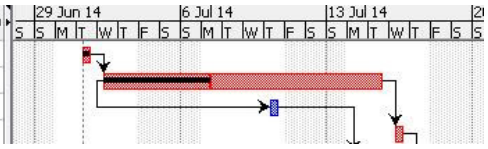
While the project is ongoing, you will gain much more insight into the project schedule if you periodically update Project*Libre* with the status of the current tasks. You can assign a completion percentage of 0% to 100% to any task and the program will then use this information to help give you a better estimate of the remaining work and the anticipated project completion date.

Let's pretend in our example that the kickoff has completed. The default column to the left of the "Name" column shows progress and if you double-click on a position, the "Task Information" box pops up for that task. Go ahead and bring up this box for the Kickoff task.

There are many things that can be set for the task as you'll see if you click through the various tabs. (Nearly all of which I suggest ignoring for your first use of the tool on a project as straightforward from a project management perspective as building the robot. I'll briefly explain "resources" later on.) Returning back to the "General" tab, enter 100 in the "Percent Complete" box and press "close." Note the green checkmark on the Kickoff line: This shows that the task on that line is complete. Now, while it is hard to see since the Kickoff task is only one day long, if you zoom in a bit on the Gantt chart you can see a black horizontal line through that task's red box. As you'll see in this next step, the black line represents the percent completion and doesn't only appear when the task is done. To see this, set the "consider designs" task to be 30% done. The black line for that task goes into Monday, July 7th as seen here:

[7] Helmuth von Moltke the Elder, the Prussian Army Chief of Staff for many years in the late 1800s, is credited with saying "no plan survives first contact with the enemy." While he was describing military war plans, project planning has the same attribute that reality rarely, if ever, goes exactly as planned so don't fear changes.

		Name	Duration	Start	Finish
1	✔	Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM
2		Consider designs	8 days	7/2/14 8:00 AM	7/15/14 5:00 PM
3		Identify common design feat	1 day	7/10/14 8:00 AM	7/10/14 5:00 PM
4		Decide final design	1 day	7/16/14 8:00 AM	7/16/14 5:00 PM



If we're not yet at July 7th, congratulations, this task is ahead of schedule but what happens if "today" is Wednesday July 9th where this task now appears to be over a day behind schedule? This is the topic of the next section. Before we move into the next topic, however, I will reinforce and explain here the message on the slides that smaller tasks are better than larger tasks. The benefit of smaller tasks is that it is easier to see if they are fully done or not. Let's use an example of opposite ends of the spectrum where one project has 10 three day tasks and another project has a single 30 day task doing the same total work. After nine days it is easy for the first project's team to know if the first three tasks are done or not since they are separately defined tasks. After nine days, how easy is it for the second team to estimate their percent completion of the one large task? In general, the larger the task the harder it is to easily and accurately track its progress and errors can be major. Smaller tasks naturally lend themselves to a more definite understanding of what is, and what isn't, yet done and errors are less significant. (Estimating the completion percentage wrong by 10% on a 30 day task is a large schedule swing when it is discovered but misestimating a 3 day task by 10% is hardly noticeable.) Do yourself a favor and try to keep your tasks small.

Adjusting Late Tasks

Adjusting the schedule to account for tasks that are running behind can be one of the hardest things to do because it requires the team to acknowledge that some part of the project is behind schedule. At the same time is one of the most important things the team's "keeper of the schedule" can do. Repeating myself since this is on the slide set's annotations, the importance of adjusting the schedule for late tasks is hard to understate since doing this is how the team can recognize what I call the "current reality path" and can make changes if that path does not point to success. The earlier you can discover that a task is running late the more time you have in which to correct it, so avoid the temptation to give the team another day or two to make it up before adjusting the schedule. Besides, the team already knows that the task is behind schedule, so not adjusting the schedule doesn't help anyone. (And if the team does make up the work, the team can celebrate and the schedule can be readjusted as will be demonstrated below.)

Sorry for what might appear to have been a soapbox, but hopefully this conveys the importance of facing schedule reality. Ok, back to the mechanics of adjusting the schedule. ☺

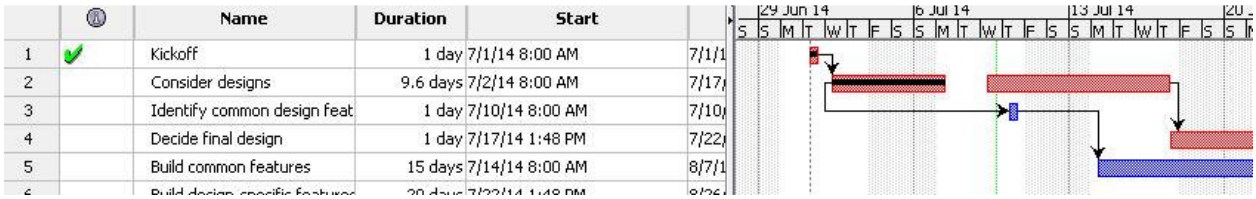
Going back to the tutorial example, "consider designs" was marked as being 30% done and is a bit over a day behind schedule since today is supposedly July 9th. What project planning tools give you is an easy of saying that any work not yet done should start on a certain day. This lets us take the remaining 70% in this task and shift it to the right (timewise) to begin today. Since this is a project-wide setting, it isn't under the "task" tab in Project*Libre*, but rather is at **File | Project | Update**. Click on this. The "Update Project" box appears. WARNING, this box has what I believe to be an unusual default, namely adjusting every task to whatever percent completion they should be at the start of tomorrow morning's work day. For projects with a small number of tasks being active simultaneously, such as the robot build, I suggest talking to the team to find out each tasks' real percentage completion instead of using this default setting.

To adjust the schedule for tasks that are late, select the "Reschedule remaining work after" button and set the appropriate date.

Now, a note about the "appropriate date": The date is likely one different from what you might expect. Read the wording of that button we selected carefully and note the "work after" part. If you put in today's date, it will reschedule the remaining work to begin tomorrow which makes sense if your team reviews the schedule at the end of the day's build activity. However, if your team reviews the schedule at the start of the day's activities, you'll have to put in *yesterday's* date to have the remaining work start today on the schedule. You'll get the hang of it quickly.

On more warning. Save your project file (**File | File | Save**) before adjusting the remaining work to start on a certain day. Sometimes, most notably if you are trying to undo a gap created by a bad date, I've seen the program apparently not display the proper task relationships. Save your work before you do the "update" task in the next paragraph. (And, of course, as with any program saving your work often is good. It is just good to do it before this function in particular.)

Going back to the tutorial example, we will presume the team reviews the schedule at the start of the day and that today is the 9th, so set the date to July 8, 2014 and press "OK". You'll now see this gap in the Gantt chart for task 2:



		Name	Duration	Start	
1	✔	Kickoff	1 day	7/1/14 8:00 AM	7/1/1
2		Consider designs	9.6 days	7/2/14 8:00 AM	7/17/
3		Identify common design feat	1 day	7/10/14 8:00 AM	7/10/
4		Decide final design	1 day	7/17/14 1:48 PM	7/22/
5		Build common features	15 days	7/14/14 8:00 AM	8/7/1

Additionally, you'll see that the project has now slipped out by a few days since task 2 was on the critical path, thus making it longer makes the entire project longer.

Oddly, the adjustment feature does not also work the other way.[8] Imagine that the team worked hard on this "consider designs" task on Wednesday the 9th and completed it. Set its completion status to 100% and then update the project for all remaining work to begin after July 9, 2014. Other than task 2 having a full horizontal black bar in it and the green checkmark appearing, nothing else happened. The easiest way of getting task 4 ("decide final design") to show as being start-able on July 10th is to remove task 4's predecessor dependency on task 2 and hit <tab>. This will cause task 4 to be moved back to the beginning of the entire project and then do the update outlined above to have all remaining work starting after today (i.e. July 9, 2014). That makes task 4's start date be July 10, 2014 and adjusts the rest of the schedule accordingly.

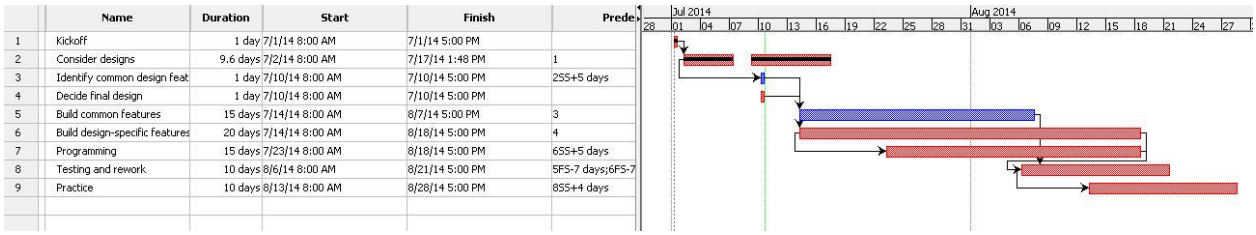
The downside of breaking these predecessors to account for tasks completed early is that you won't have them on the schedule to act as next year's template. If you double-click in the status column for the task (the column that has the green checkmarks for tasks 1 and 2 at this point), the "Task Information" box has a "Notes" tab at the far right. If you select that can you can make a note to yourself that this task used to

[8] I suspect its definition of "remaining work" means what should have been done, but hasn't yet been done, from the start of the project to today. Future items aren't yet late and thus must not count as "remaining."

have another task as a predecessor. This would make it easier to reuse this project as a template next year.

Wrap-up

At this point, your screen should look like this (zooming out a bit):



	Name	Duration	Start	Finish	Prede
1	Kickoff	1 day	7/1/14 8:00 AM	7/1/14 5:00 PM	
2	Consider designs	9.6 days	7/2/14 8:00 AM	7/17/14 1:48 PM	1
3	Identify common design feat	1 day	7/10/14 8:00 AM	7/10/14 5:00 PM	2SS+5 days
4	Decide final design	1 day	7/10/14 8:00 AM	7/10/14 5:00 PM	
5	Build common features	15 days	7/14/14 8:00 AM	8/7/14 5:00 PM	3
6	Build design-specific features	20 days	7/14/14 8:00 AM	8/18/14 5:00 PM	4
7	Programming	15 days	7/23/14 8:00 AM	8/18/14 5:00 PM	6SS+5 days
8	Testing and rework	10 days	8/6/14 8:00 AM	8/21/14 5:00 PM	5FS-7 days;6FS-7
9	Practice	10 days	8/13/14 8:00 AM	8/28/14 5:00 PM	8SS+4 days

If it does, congratulations! You probably now know all you need to know about project planning tools to apply it to a project of the size of the robot build. If your screen doesn't look like this, go back and review the steps with a particular eye to the calendar dates being entered. If you are still having problems (or just feel that I explained something in a confusing way), feel free to send me a question at cyankes@msa.com and, to help me more easily see your question amongst my emails, please put "ProjectLibre" somewhere in the subject field.

Deferred Topics

Creating a Project with Forward versus Backwards Scheduled Tasks

Back at the beginning of this tutorial when you were creating the project, I asked you to leave the "Forward scheduled" box selected and to enter July 1, 2014 in the Start Date field. I mentioned that I'd explain what this "forward scheduling" means later on. The reason I did this was it might have been confusing to describe the "forward" and "backward" options before you had some experience working with tasks.

Throughout this tutorial, we've used what is called "forward scheduling" which is another way of saying "if the project starts on this date and has these tasks, when will the project be done?" When we added tasks or adjusted durations, therefore, we weren't changing when the project was starting but rather we were changing when the project would end. "Backward scheduling" is the opposite and essentially asks the question "when do we have to start this project with these tasks if it has to be done on a certain day?" In backwards scheduling, if you add tasks to the critical path or increase their duration, the mandatory project finish date remains fixed but the start date will move earlier on the calendar.

Both of these styles have environments in which they make sense. "Forward scheduling" is quite common on projects with definite starts dates but indefinite end dates and is often used to give upper management an idea of when the project might finish (and thus an estimate of its costs). "Backwards scheduling" is often used in environments where there are contractual completion dates and the company performing the work wants to know when the various parts of the project have to start in order to meet the contract.

Oddly, the robotics build is both.

You have both a definite start date, kickoff, and a definite "stop build" date. In this sense, either forward or backwards scheduling can work for the robotics project. In either case, the quality of your schedule depends upon the team's ability to identify and estimate the duration of the tasks and in either case you have to adjust for tasks being ahead or behind schedule. Therefore, you can choose either method. One important difference between these approaches, though, is the impact of a task taking longer than expected. In backwards scheduling, tasks are delayed as long as possible and are generally started at the last possible moment. This works well if you have a solid understanding of how long something will take. (For example, while constructing a building a general contractor with experience will likely know down to the day how long it will take the pour the concrete foundation.) If you do choose to use backwards scheduling, I suggest that you start to work on tasks as early as you can anyway to allow for later schedule slips. Personally, I'd use forward scheduling for the robot build.

Resources

I mentioned several times above that there are aspects of project management tools that you are probably better off ignoring on the robot build. We've concentrated on defining and managing tasks which is one side of project management. The flip side is scheduling and managing "resources". Resources include both people and special equipment that might be needed on the project. Returning to the construction example from the previous paragraph, if the general contractor has to rent a special piece of equipment for a few days, the availability of that machine factors into when the task that uses it can be scheduled.

On most projects, though, people are the resource that the project management team has to be concerned with. Imagine a large (200-300 person) software development organization starting a year-long project with hundreds or even thousands of tasks. Among all these tasks are the following two tasks (and merging the normally separate design / build / test / rework tasks into single tasks for simplicity here):

- Design, Code and Test Function Alpha – 8 fulltime weeks.
- Design, Code and Test Function Beta – 10 fulltime weeks.

These tasks have no dependencies on each other, can both can start on Day 1 of the project and both gate the starting of other tasks. Taking a "task-only" view of the project (such as what we've done throughout this tutorial since this simplification works for the robot build), both of these would be shown as start-able on Day 1. However, what happens if both of these functions have to be written by the same programmer that has a particular skillset? Will Alpha be done in 8 weeks and Beta done two weeks later? Not likely. This is where the resource side of project management is useful: If the program knows who is to work on each task then the tool can easily identify where someone is "double-booked" (or worse) and shift the

schedules accordingly. (Or, of course, if this is a major schedule slip it can lead the project leaders to hire a contractor that also has that particular skillset to pull the schedule back in.)

Similarly, having people assigned to the tasks allows tools like Project*Libre* to show when someone is less than 100% busy which allows the project managers to know when they will have a person that can be temporarily assigned to help with other tasks.

Managing large projects, therefore, requires balancing both the task schedule and resource sides of the picture which can become time consuming. This is why I've suggested staying away from resource scheduling on the rather straightforward (from a project management perspective) robot builds. If you do have someone with a unique skill that is needed on multiple simultaneous tasks, an easier way of dealing with it is to create a dependency between the tasks so they cannot be simultaneous or lengthen the duration of each of the tasks to account for the person being split between them.

During the robot build "off season" (i.e. perhaps after Regionals/Nationals), I do invite you to make a copy of your team's real schedule and explore what happens when you assign people to the various tasks. The Project*Libre* books on Amazon are very good resources to help you understand dealing with resources.

Appendix 1: 2013 Team 3511 Task List

As mentioned above, Team 3511 used an Excel spreadsheet for the 2013 build to track their progress. Jim Broker of that team has graciously allowed the spreadsheet to be included in this Tutorial as it might help you break down the robot build into tasks. Since subsequent competitions won't be throwing Frisbees, your team's details will be different but this is a good example of a task-level that worked for a team.