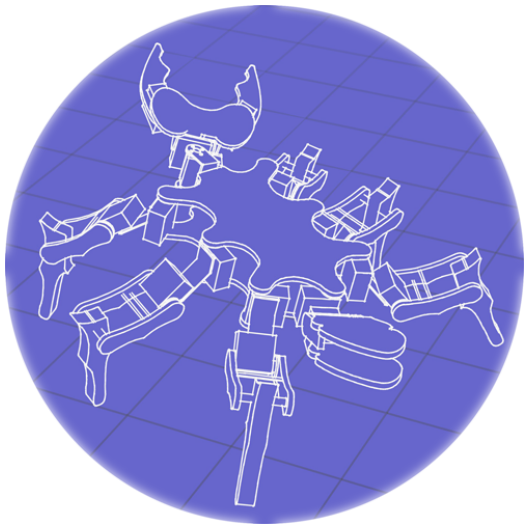
RoboCup Manual - Nationals 2015, Adelaide

Baralaba Robotics

Open Dance



John Board

September 23, 2015

Page intentionally left blank.

Abstract

This document outlines relevant information about the research, construction, and development of Baralaba Robotics' Open Dance performance for RoboCup Junior Australia Nationals, 2015 Adelaide.

It is my goal, as much as possible, to provide clear documentation in both chronological and topical formats. It is designed to be clear so that if anyone wants to recreate all or part of what I have done, they can do so with significantly less effort and time.

I plan to achieve this by providing a clear format by which I present the information through proper sectioning and indexing such that should anyone want to know anything about what I have done, they can quickly look it up.

A significant portion of the work I have done has not been done before, and as such I had to make all the mistakes to find out how to do it properly. I want to help others not make these same mistakes in the future.

NOTE: I wrote a significant amount of this documentation in a hurry, and as such it is not by any means exhaustive, nor does it contain all the information I wanted it to. Feel free to me at documentation@johnrobboard.com if you want further information!

Acknowledgements

I would like to acknowledge Brisbane Boy's College for their continual support and generosity, especially in the area of allowing me to travel with them to RoboCup Nationals, 2015 Adelaide.

I would also like to extend a special thank you to my parents for supporting me through my RoboCup endeavors through the period I have been doing it. Their continual love, support, and wisdom has not only helped me, but has served as an example to me.

Contents

Introduction

Accompanying Resources

This documentation can stand by itself, however your understanding will be enhanced by using the extra content as outlined in the following subsections.

Wiki

I've very recently started maintaining a wiki, which I hope to populate with all the knowledge that I've amassed through trial, failure, and success - not only from this year, but also from previous years of competitions.

In early 2014 I received an email from someone who I had met at RoboCup Nationals, 2012 and given my email to. I couldn't remember the particular person, but they sent me an email asking about how to build custom built rescue robots. What ensued was a some 9620 word correspondence over 5 months, in which I outlined all the lessons that I had learned in regards to motors, locomotion, sensors, electronics, manufacturing, debugging, etc.

I don't claim to have a great deal of knowledge because I've succeeded a lot - I claim to have a small fortune of knowledge because I've failed so much. I know what doesn't work - because I've often experienced failure. There are, however, a few small successes - which over the course of 3 years shaped how I created my robots. The result was a robot that was almost "perfect" in 2013 - if I had continued Rescue for another year, I believe that I would've produced an almost perfect rescue robot.

Because of the knowledge of all these failures, and how much time and money this knowledge could save if it was available to the general community

of RoboCup Rescue, I thought of releasing this information on a Wiki. At this point it's in it's very early days - but I hope to continue development of it.

I genuinely believe that a combination of my experiences, with the experiences of others culminated onto a single Wiki could greatly thrust the level of Rescue light years ahead. Imagine if a school could read a wiki which told them what sensors to buy, and what sensors not to buy - and listed all the reasons why.

But not only would it lift the level of competition within RoboCup, it would in the end produce better engineers entering university, as they've been able to cover much more ground, and experience many more technologies than previously available.

I've been unable to put a whole lot of content in the wiki before Nationals, 2015 - however I intend to put all my acquired knowledge in it over time. You can see the latest changes on the wiki on this page:

`http://linnode.johnrobboard.com/wiki/index.php/Special:RecentChanges`

GitHub

Throughout this document I make mention to my code. I frequently mention that it's located on BitBucket in a private repository due to the copyright nature of some of the content I'm using. I've made a copy of this repository, removed any sensitive information, and uploaded it to GitHub. You can find this repository here:

`https://github.com/boar401s2/RCJA2015`

YouTube

Although I dearly love the communication medium of text, I have been learning to also love video. As such, I've been creating a few videos that help illustrate what I do in robotics. Although I only have a few robotics videos on there, and at fairly low quality - I intend to increase that all the time.

You can check out my YouTube channel at:

`https://www.youtube.com/channel/UC5_HY8gw3HC2Rq7axfU144g`

Facebook

I frequently share the status of what I do, with pictures and documentation on my Facebook account. Specifically written to the reviews of this logbook, should you wish to see further information, feel free to add me:

`https://www.facebook.com/profile.php?id=100005747055401`

NOTE: I am in the process of creating a public Facebook page where I will share these posts, that will accompany my website. It will be called WhiteBoard (as my website is also called WhiteBoard).

Website

Over the last few weeks (and as documented in the Log chapter), I have been working on a personal website/portfolio. It is my intention in time that I share what I would normally share on Facebook on a blog here. I would then link to it using the Facebook page.

How to Read

I recognize that this document can be quite lengthy for the average reader, and as such I've added some features that will hopefully help you find what you are looking for quickly.

You could either read this document as a book from start to end - in which case you would get the entire view of this project. You could alternately read just the Log chapter, which would give you an "as it happens" approach to what I do on a day to day level. Finally you could treat this

book as a reference book, and search for particular problems (and/or solutions) using the index.

Although I've tried to make the index comprehensive, it won't be. Try looking for what you want using the table of contents first,

People have questioned why I included an index, and the reason is because I have personally found that having an index is very beneficial - especially in large documents. An index can cover specific information, whereas the Table of Contents only gives a broad gist.

The index should be especially useful in the log section, where I list specific problems that aren't mentioned in the table of contents.

Revisions

This document is the third revision in the series.

The first revision was for RoboCup Regionals. It was compiled on the July 23, 2015. The first revision was 29 pages long.

The first revision was for RoboCup States. It was compiled on the August 13, 2015. The first revision was 43 pages long.

The third revision is this, designed for RoboCup Nationals. This document was last compiled on the September 23, 2015. It's length is currently undetermined - check the end of the book for more information!

Contact

A large portion of this documentation was created on a hurried schedule, and as such I expect there to be at least a few mistakes. If you feel so obliged to report these problems, or if you have any questions in general, please feel free to contact me using the details below.

Alternately, if you want a digital copy of this document, feel free to send me an email at:

Email: documentation@johnrobboard.com

Part I

Metadata

For lack of a better name, I've called this chapter Metadata - or data about data. The reasoning behind this is that this part is about the team, and it's history - which is not technically what this document is about, but it does contextualize the "data".

Chapter 1

History

I had the pleasure of being invited to present at an Art Gallery Exhibit in Goombungee, *Gumna No Sheila*, in Goombungee in the month of November, 2014.

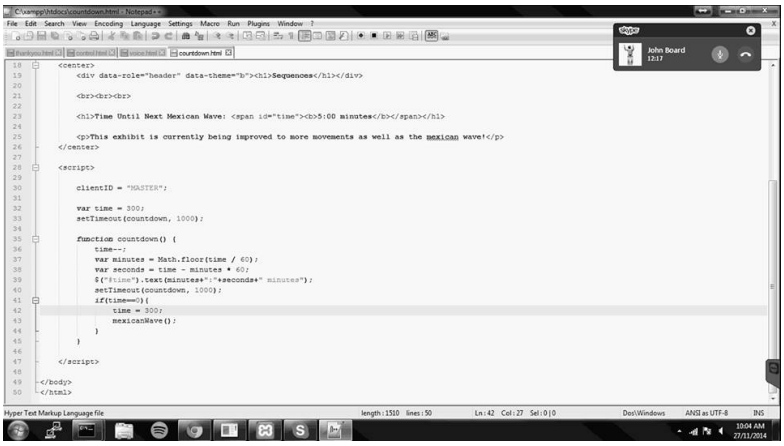
The show revolved around "things men do in their sheds". Ranging from steam machinery, wood sculptures, motor bikes, kites... and our robotics.

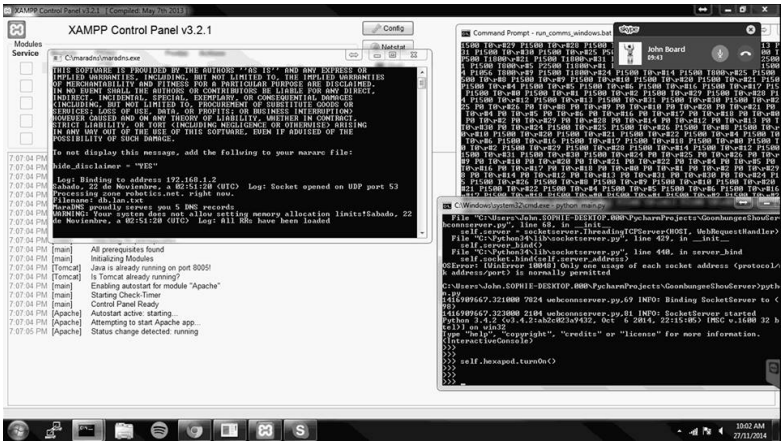
1.1 BaralabaBob

BaralabaBob (the hexapod, talked about in the Roobtics chapter had been ordered several months previously, and arrived just in time for me to prepare it for the show. I spent the month of October preparing, but specifically from the 12th-18th of October, in which time I assembled and wrote the code to control BaralabaBob.

This show is significant because in preparation for it, I was essentially preparing the building blocks for RoboCup. A lot of the code I wrote for Goombungee I use today.

The hexapod had to be left, without my human presence, in Goombungee for 30 days. To that point I had never developed a system that had to operate for more than a few minutes - this presented its own challenges in me trying to design it as reliably as possible.





Although the system I created was fairly reliable, I did have a few computer problems which I had to sort out remotely, by means of instructed a human over telephone how to reboot my systems, and connect me to the internet. These were quite challenging, but all challenges were overcome.

The result of these efforts meant I created a very reliable and well rounded codebase right from the start. Originally the code was designed to run on an external Windows 7 desktop, however I was able to easily port the python code to operate on the now Raspberry Pi.

1.2 Beaker

As well as having an exhibit set up, I visited Goombungee for a weekend to give talks, and personally present the robot. During that time I set up "shop" in the attached Library. In the library I had several interactive exhibits which people could visit and play with. I also presented a few talks, aided with a PowerPoint, to visitors.

One of the interactive robots was Beaker, a plywood robot with electric-car-seat (see it's section in the Robots chapter for more information).

Beaker was mounted with an Apple MacBook Air, which was connected to a private wireless network I maintained in the Library. Then my laptop,

with my joystick connected, would send messages from the laptop over wireless to the MacBook which was connected via serial to a Propeller Chip which would receive the signals, and then send them on to two HB-25 motor drivers.

People could control Beaker using the . I had also prototyped a system which would use the skeleton tracking capabilities of the Microsoft Kinect to control Beaker. Although people could control it using hand gestures, I noted that they generally opted to go for the tactile .

1.3 Conclusion

Goombungee was a good time, in which I many valuable skills such as designing long term and reliable systems, improved my public speaking, improved tech support communication over the telephone, and learned a bit more about how humans interact with robots - amongst many things.

Goombungee also provided a solid code base from which I could build the later systems for RoboCup from.

If you have any more questions on this topic, I'd be more than happy to answer any of them if you send them into me using my contact details.

Chapter 2

Team

Baralaba Robotics is a team that has been competing in RoboCup for several years. The only "team" member is John Board. John has attended 8 RoboCups at both regional, state, and national level.

2.1 John Board

2.1.1 History

Typically in the past Baralaba Robotics has entered the Rescue divisions with 3 generations of custom built robots.

John first heard about RoboCup on TV one evening whilst still living in Brisbane. He saw the soccer robots, and dance robots on TV. He made a commitment to himself at that point that he wanted to competed in that competition one day.

After moving to Rockhampton, John saw an advertisement for RoboCup in 2010. He decided to go along to watch. After seeing the other kids, he thought that he could compete too.

The year after in 2011 he entered a modified version of the BOE-Bot from Parallax Inc into Rescue.

After that John entered several more competitions at State, Regional, and Nationals level.

John's most memorable RoboCup was Nationals in Canberra, 2012, to which he was sponsored by RCJA QLD to attend. He enjoyed it because although his robot completely bombed out on the day, he was able to talk to a lot of like minded students, and have a lot of memorable experiences in Canberra.

2.1.2 Media

John has been no stranger to the media over the years, after being featured in several newspaper reports, two TV features, and several other TV appearances.

The most recent TV interview can be watched here:

`https://www.youtube.com/watch?v=OhAaHymBDpM`

John's first TV interview meant a lot to him, as he felt the circle had been "completed" - he was first inspired to attend RoboCup from watching kids on TV... he was now able to inspire others through appearing on TV himself.

2.1.3 This Year

Many people wondered why John switched from Rescue to Dance - as this year is the final year that John Board will be able to compete, he wanted to finish with a bang - and pour all his skills into the division to create an amazing performance.

Although typically quite reserved, John has embraced his need to let go a little with the Dance Category - and as such he hopes to provide an interesting counterpointal between himself and the robot with dance.

2.1.4 Next Year

People ask me what I would like to do next year - I am goaling to study a Bachelor of Mechatronics, majoring in Robotics and minoring in Areospace Avionics at QUT.

John has undertake significant Leadership & Teamwork training with The Boy's Brigade Queensland. This combined with many other events that John has had the opportunity to help organize has provided him with skills he would like to put use organizing events whilst at University.

Chapter 3

Sharing

I consider information sharing of the utmost importance, including the open source movement. Over the years the openness of the robotics and software community has enabled me to learn what I have. Unusual kindness and time spent on me by community members has served as an example on how I can act likewise.

The most notable instance of this took place in 2013. I quickly needed some form of ADC for my 2013 Premier Robot, Achilles. Because I didn't have the time to order an ADC, I asked the Parallax Forum community on help to design a cheap ADC-like system using simple electronics. It wouldn't be optimal - but it would be good enough.

One of the (only) Australian members offered that because we lived in the same city, I could drive to his house, and he would give me some MCP3208 SPI ADCs. After further inquiry I found that he lived only 8 minutes from me!

I promptly drove to his house - and was greeted by his great in showing me what he does, giving me a tour of his lab, and giving many, many more parts (wires, motors, and other misc electronic components for me to play around with).

I believe he also ordered some more ADCs which he sent me in the post - but not only this, he also sent a bunch of other parts, most of which I actually used on the robot!

This, along with the time put into my endeavors on the forums have inspired me to do likewise. I also find it very rewarding sharing what I've learned with others - and seeing how it improves their own robotics. Listed below are a few ways I have been sharing my knowledge:

Goombungee Exhibit: The Goombungee exhibit, as outlined earlier in this part in the History chapter, is an example of how I have shared my knowledge by setting up an exhibit and maintaining it for a month. Beyond that I also traveled out to Goombungee and over the weekend I daily held robotics shows in the local library, as well as talking to many locals and tourists about robotics.

Central Queensland Trip: After competing in RoboCup Junior Central Queensland, I traveled out to my old hometown, Baralaba. Whilst I was there I gave several talks to primary and highschool students, as well as putting on a presentation for the community in the local Ambulance Station in both Baralaba and Wowan.

Fiftysix: I have recently concluded my work writing a curriculum for the Fiftysix Tablet. In this program I teach about the inner workings of a tablet.

RoboCup Junior Australia: Where better to share my knowledge about robotics than at RoboCup! I regularly attend the RoboCup events, and whilst not competing I regularly give the other teams support. An example of this was whilst at RCJCQ this year I helped fix a Mechanical issue on one of the Lego Dance teams, enabling them to compete. I also regularly assist with programming issues.

Website, Facebook, Wiki: As I mentioned in the Resources section of the Metadata part, I have recently started pooling my experience onto a Wiki. I also frequently post what I'm doing with images on Facebook (as also explained in Facaebook.

Contact Details: I regularly hand out my contact details at competitions or other robotics events, inviting people to contact me if they want to know anything. I have been contact on two or more occasions as a result

from this.

GitHub: I upload most of the code I write to GitHub, a free code sharing service, making it available to anyone who would like to look at it. The only times I don't upload to GitHub is when I have models that potentially breaks Copyright.

UQ Robotics Club: I have had the opportunity to attend the UQ Robotics club since 2013. In such time I've been able to help some of the students there with projects that they are working on.

This is just a small swatch of what I do to give back to the community that has given the same skills.

Part II

Log

I'm the first to admit that I'm not great at keeping a chronological - as you may have guessed from the rest of this document, I tend to write the logbook in blocks, but not as it's happening. This said, I do tend to leave a digital trace of my progress through frequent commits, Facebook posts, and emails sent. I've tried to accurately compile a chronological log of major events throughout the building process of this robot based upon this information until the 29th of August, 2015 from which point I am keeping a more traditional log - filling it in "as it happens."

Chapter 4

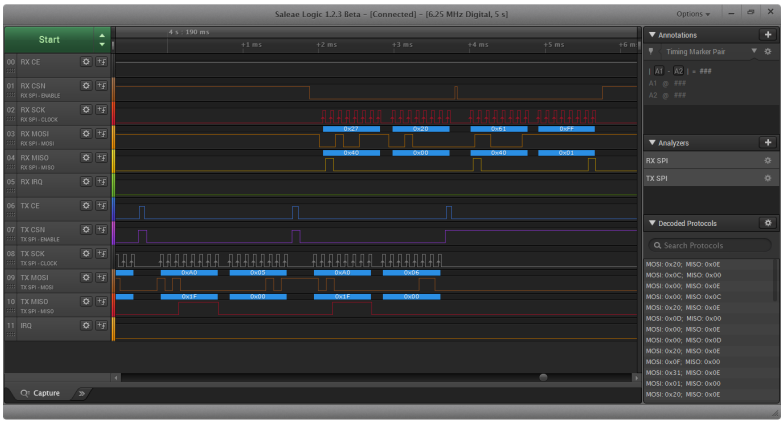
August 2015

4.1 August 28th/29th, 2015

I've been once again revisiting the nRF24L01+ modules, attempting to finish the drivers that I've been writing for the Propeller chip. On top of that I've been cram-learning C. In the future I may rewrite these drivers for the propeller chip in C using the PropGCC compiler. Random side note, I've found the concept and implementation of linked lists fascinating - and have ideas on how I can use such stuff on the propeller platform.

For those at home who've just tuned in, I have a few cheap rf modules that I'm looking at using in the performance. I have yet to reliably implement the drivers for them.

Progress has been slow over the last two days because I haven't been working on this code for a while I have to get back into the same "mindset" again. It's 11:25PM on the 29th, and I'm starting to make some positive progress.



The main difficulty has been remembering to send the correct sequences to properly initialize the nRF24L01+ modules. I've found that the following bytes work well:

Transmitter

Command	Data	Description
%0010_0000	%0000_1110	EN_CRC=1, CRC=2bytes, PWR=1

Receiver

Command	Data	Description
%0010_0000	%0000_1111	EN_CRC=1, CRC=2bytes, PWR=1, PRIM_RX=1
%0011_0001	%0000_0001	Set RX Pipe 0 packet width to 1

Previously I was having difficulties with the receiver only picking up one packet before stopping receiving - unknown reason. I reverted back to some old code from GitHub for a retry - everything working so far.

Now I appear to be having difficulty picking up new packets - the first packet (0x01) keeps on repeating for whatever reason - perhaps it's not deleting it from the FIFO buffer when I read it.

Another issue encountered was that between code changes I was only restarting the MCU, not the nRF24L01+ modules (meaning that they kept all the old data in their buffers, old settings, etc). I needed a way to switch the power of both MCUs on and off at the same time. In the past I've simply flicked both switches of both MCUs at the same time (before recording data

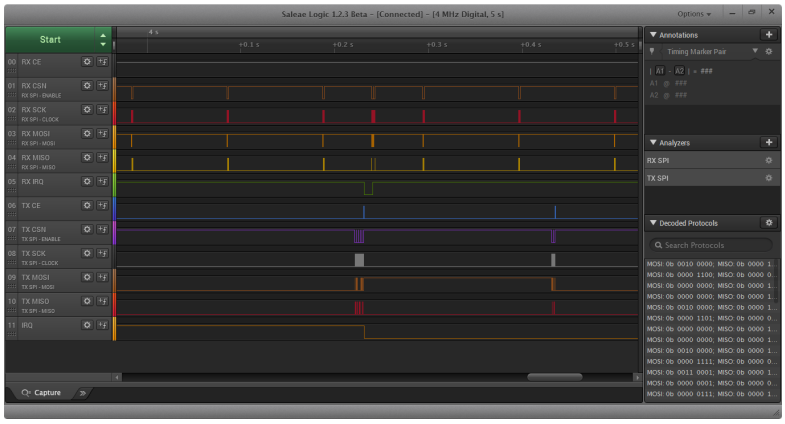
using the Logic Analyzer). Because of limited resources, this wasn't an issue.

To combat this issue I drew power from the board with the switch, and fed it into the 5V rail on the other board through a servo header that I had previously built into that board. This solves the issue allowing for me to capture all the data for both the transmitter and receiver on the logic analyzer cleanly.

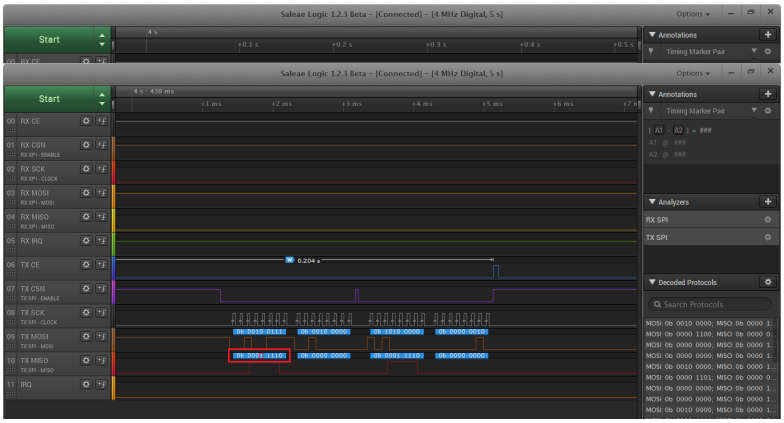
4.2 August 30th, 2015

I've been continuing work on the nRF24L01+ modules today. I have found that with both Auto Retransmit, and Auto Ack left to default on the transmitter, I get 1 byte received on the receiver, however when I disable both Auto Ack and Auto Retransmit, I do not.

I define packet reception as IRQ dropping low on the RX end. I've been doing some playing around and it would appear that when Auto Ack is left to default, and Auto Retransmit is turned off, the system works fine.



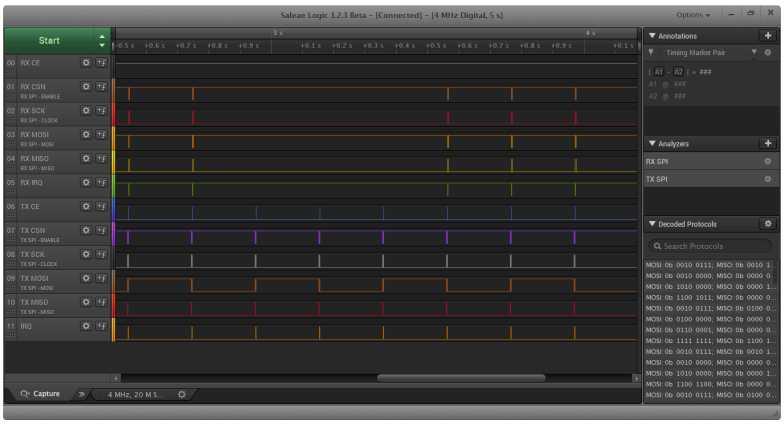
Here we can see that IRQ on the RX end drops low for one packet, however the IRQ on the TX end is low continuously after the first packet.



In this image you can see that the max retransmit bit has been set to one in the status byte. This means the transmitter has tried transmitting a indexbytebyte, but it didn't get any acknowledgment back - so it tried retransmitting. It's reached is max retransmit amount without any return signal, so it throws the MAX_RT IRQ flag.

To fix this I can either disable AUTO_ACK on the TX, or enable AUTO_ACK on the RX end.

I ended up fixing the problem by disabling AUTO_ACK on both RX and TX. This solution is not optimal as there is obvious packet drop between TX and RX - which the auto retransmit would normally fix. From now I'll be working on making this connection more robust (by enabling the autoack) and adding more features to my driver code.



4.3 August 31st, 2015

I didn't do too much today. I've been picking out some songs for the routine, but nothing major yet. I have several criteria I use when picking out songs. They either have to be well known (and loved), or have a good strong beat that people generally like. Secondly the section of the song I want to use has to be 20s long, 40s max. If I have too much beyond this, I don't have enough time in the routine for other songs.

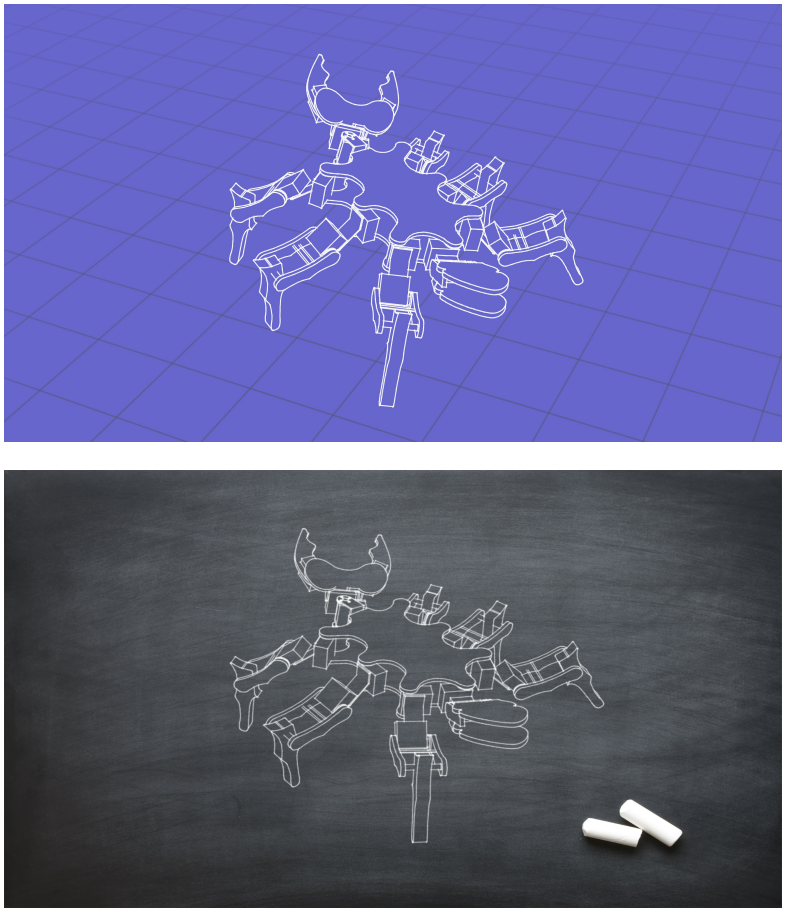
NOTE: Also read Audio Selection

Why do I use different songs? Having not entered dance previously, and thus not being affected by the general mentality to only pick one song, I thought of having a mashup of multiple songs - inspired by this Britain's Got Talent performance:

`https://www.youtube.com/watch?v=FP7wN301yjc`

At regionals I tried this out, and it was highly praised. It was commented that it showed that the robot could dance to a variety of genres. This effect was multiplied at regionals because of the variety of genre I was using (classical, pop, rock, etc). With a general mix of more contemporary songs this effect is lessened, but I feel that it provides variety to the routine.

Beyond that, I took some time to find a suitable theme for my website, which I hope to use to post information about my exploits on. I spent time experimenting with the "Freestyle" rendering option in Blender, in an attempt to make a logo/letterhead/headerimage. I was quite pleased with the results:



As I mentioned, I used the freestyle rendering option in blender. For the "blackboard" image I exported the lines generated by this function as an SVG, which I imported into Inkscape, which I then used to superimpose onto a blackboard. I also modified the blur and opacity of said lines in Inkscape to give a "chalky" effect.

Chapter 5

September 2015

5.1 September 1st, 2015

Pinch and a punch for the first of the month!

I played around with some more blender rendering today. Not much to report here. Did some more work on the website.

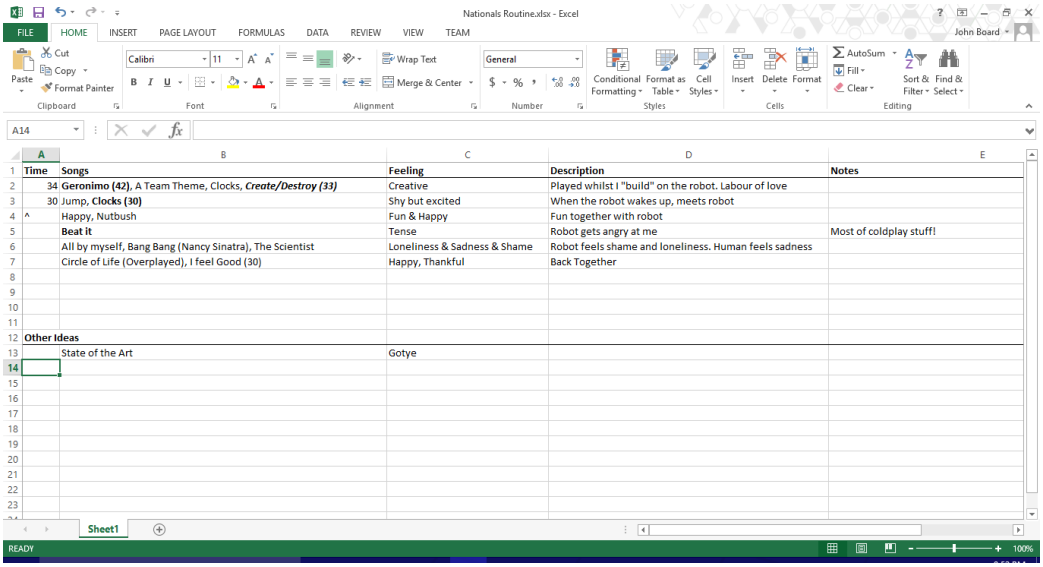
I personally felt that the regionals routine was much better than the state one. I had given a few days to developing the regionals routine, and months of thought into songs and actions. I was able to take my relative time in putting a good routine together. I also felt that the Christmas Tree prop provided a good "setting up" for the theme.

In comparison, I spent 99% of the time in preparation for States building the stage. Although I produced a very nice stage, I didn't find a good way to use it, so at 1AM on the night before the competition I was scrambling to get a routine that fit the stage usage. I made the routine fit the props, not the other way around.

With these lessons learned, I have decided to do a little bit of routine development for a long period of time (30min or more a day for a month). This will allow me to look into getting a great selection of music that flows together well in a flowing story. I have time to remove or change parts of the routine if I need to.

Also see: *Props*

After talking to people, and reviewing the footage I have noticed that using popular songs that everyone knows generally draws in the audience. As such, I have tried to continue this theme. Here is a of my brainstorming spreadsheet as it stands:



Nationals Routine.xlsx - Excel

Time	Songs	Feeling	Description	Notes
34	**Geronimo (42)**, A Team Theme, Clocks, ***Create/Destroy (33)***	Creative	Played whilst I "build" on the robot. Labour of love	
30	Jump, **Clocks (30)**	Shy but excited	When the robot wakes up, meets robot	
^	Happy, Nutbush	Fun & Happy	Fun together with robot	
	Beat it	Tense	Robot gets angry at me	Most of coldplay stuff!
	All by myself, Bang Bang (Nancy Sinatra), The Scientist	Loneliness & Sadness & Shame	Robot feels shame and loneliness. Human feels sadness	
	Circle of Life (Overplayed), I feel Good (30)	Happy, Thankful	Back Together	
Other Ideas				
	State of the Art	Gotye		

I'm a great believer in story telling. I wanted to convey a flowing emotion throughout the performance. I believe this draws the audience in, and can evoke an emotional feeling in them. For this routine I modeled the story off a Biblical Parable, "The Prodigal Son".

Also see: *Story*

I'm now fairly happy with the selection of songs - and I'm getting excited about the routine. I think it'll be the best one yet. (or maybe it's just because I'm tired, it's 12:11AM - can't get to sleep).

I've started cutting the songs together using Audacity. I've just cut together the snipping of sound from *"Create/Destroy" - Art vs Science*. When editing the songs together I have to be careful about where I cut the music of them. If you cut a track in the wrong place it can sound disjointed.

Also see: *Audio Editing*

I use a few tricks to get around this. Firstly (and the one that seems to make the most sense), I fade the tracks in and out (and sometimes into each other). The second trick, and one that I've used twice now with great success, is to cut the start and the end of a song together. The best example of this is *"I Feel Good" - James Brown*, in the regional and state routines where I took the starting section "Heeeeyyy! I feel good, nanana...", and stitched it to the end, "so good (bump bump), so good (bump bump), I got a-you....". I'm also using this trick with the *Create/Destroy* sound track as I've found that cutting the sound track part way through doesn't sound good, but if I stitch the start and end together carefully, it sounds much better as the song naturally ends.

5.2 September 2nd, 2015

Today I've been working on the website. I think it looks quite spiffy now. I'm using the wordpress theme, Enigma. The website is designed to provide a portfolio for what I do, and place which I can share information and things that I've learnt. It can be found at:

`http://johnrobboard.com/`

I ran the design by a few people on Facebook, and got a positive response, with some suggestions which I've now implemented.

I've also been finalizing the songs for my nationals routine. I've swapped out *"All by myself"* for *"Bang Bang"* - Nancy Sinatra. I came to this conclusion after consistent disgust with *"All by myself"*. I felt *"Bang Bang"* is a very good match for the routine as it seems to even narrate the story. During the song it says "he wore black, and I wore white...", as the hexapod is black, I was thinking I could wear a white lab coat. This would fit in quite nicely.

5.3 September 3rd, 2015

Today has been quite stressful as I've been organizing my QTAC applications and SAT exam dates. I only get one shot at the SAT exam, and it's only a week after Nationals. I'm going to have to fit robocup into a tight study schedule.

I would like to more or less finalize the music in the next few days, which will allow me to start animating in plenty of time, as to not stress my SAT prep. I have enough time to do everything, if I'm wise as to how I spend it.

Currently I'm looking at using:

1. Create/Destroy - Art vs. Science
2. Clocks - Coldplay
3. Beat it - Michael Jackson
4. Bang Bang - Nancy Sinatra
5. Circle of Life - The Lion King

Also see: *Song Selection*

I think there's possibly a better song than Circle of Life for the ending, but haven't found it yet. The key is fitting in all of these songs. I might cut short Create/Destroy.

I'm also looking at getting back into my army fitness training, as there are a few events next month which I'll need my fitness up for again.

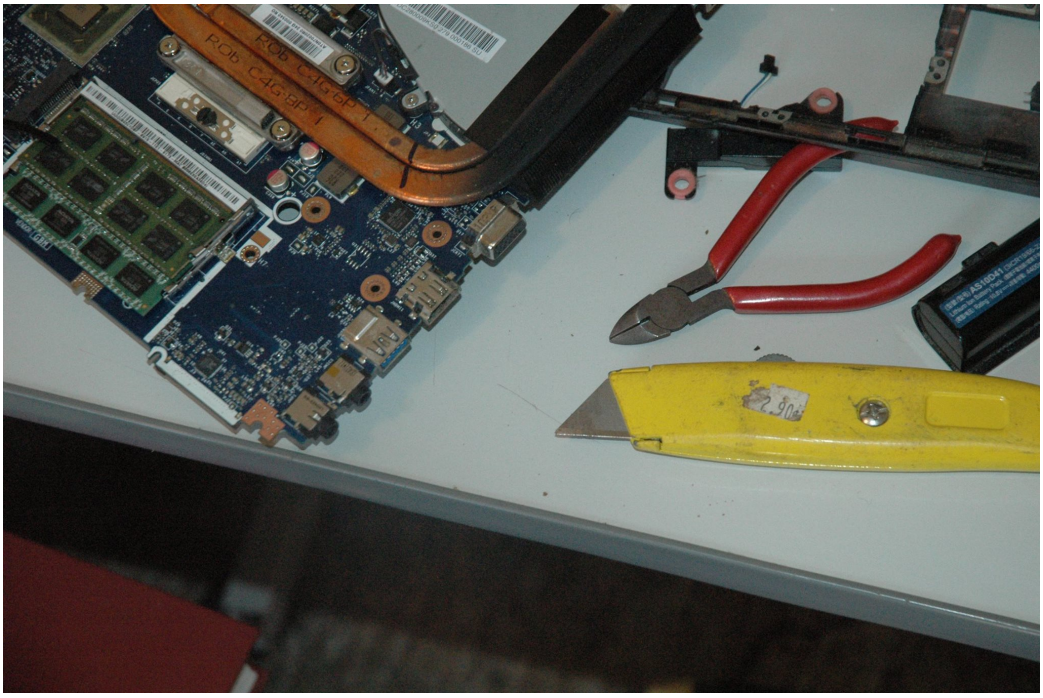
To organize my time I use a method by Dr. Stephen Covey in his book, *"The Seven Habits of Highly Effective People"*. This has helped me organize my time significantly, and I have seen great improvement because of it - and the other habits he outlines in his book.

5.4 September 11th, 2015

It's been many days since I've worked on anything important, and this is mainly due to the demise of the monitor on my old laptop. The hinges holding the monitor onto the base of the laptop had separated from the base. Although fixable in the short term, I'll be needing a working laptop for Uni next year, so I opted to purchase a new one. I ended up acquiring a Razer Blade 14".

As for the old laptop, it's electronics still work very suitably, so I opted to turn it into a "porta-desktop". I didn't want to leave the monitor on it because I felt leaving a half broken monitor had a chance of damaging the base even further. I removed it, and now use it with one of my many unused external monitors.

There were also a few minor issues with the laptop, such as a very dodgy power jack, broken audio jack, and 2 broken USB ports. Whilst I had the laptop open to take the monitor off I fixed these problems through soldering, etc. The "surgery" went well, and my sister is now using it.



After the laptop fiasco, I went back to setting up my new laptop. I save all my files for this project on BitBucket (due to free private hosting, compared to using GitHub). I tried git cloning the project onto my laptop and was annoyed to find out that it didn't work - over both HTTP and SSH cloning. It would fail at 8% (which took 15min to reach), because of some error.

I finally decided to download it directly as a zip from my account. I was surprised to find it was some 600MB in size. After attempting to download it multiple times, I was disappointed to find my download speed never went above 100KB/s. I had previously been playing around with VPS/Web hosting services to get my website a little closer to home (to allow for faster download speed and less latency). I had played around with using an Amazon EC2 instance, and still had my account set up. I remembered that I had noted that the EC2 instances had fantastic internet speeds, so I launched an instance of Ubuntu 14.04, from which I wget'ted the zip (using HTTP auth) from BitBucket. This process took less than 3min, achieving a peak speed of 7.4MB/s. I then downloaded from that host onto my computer achieving a peak speed of 4.2MB/s (obviously much faster than 100KB/s). This process was very satisfactory, and was completed in under 10min (compared to a prospective hour or more).

I downloaded the Adobe Creative Cloud onto my computer, with a view of trialling such software as Premiere, After Effects, Photoshop, Illustrator, but more specifically for RoboCup - Audition (the sound editing software). I'm using the 30 day trial, but leaving money aside should I wish to purchase a year subscription to the student creative cloud.

I also installed the student versions of Autodesk Maya, and have the intention to install others from the Adobe suite.

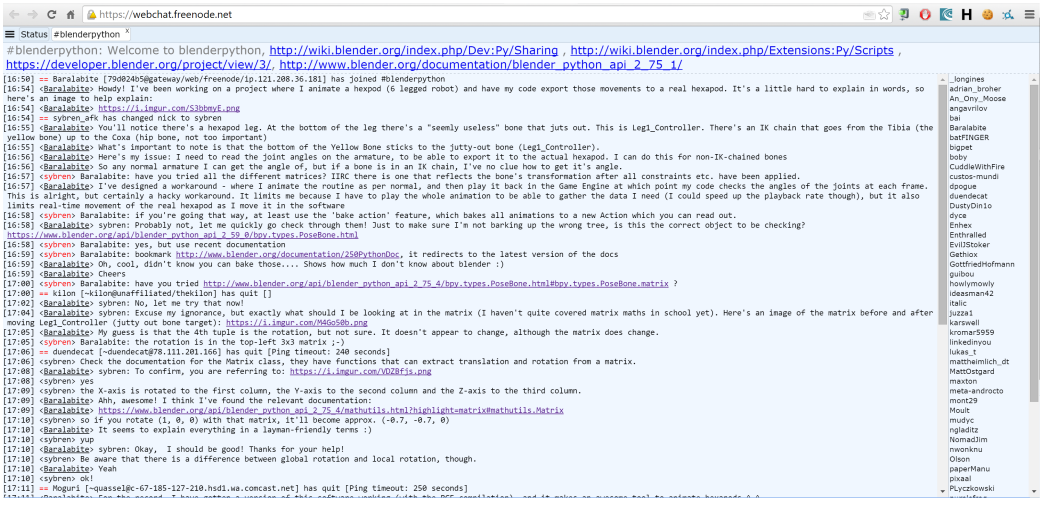
I have been pleasantly surprised by how much I've been enjoying the graphical design aspects of Blender. I've been getting an appreciation of what graphics designers do. I'm considering spending some significant time each week learning how to use the professional tools such as Photoshop, Premiere, Audition, Maya, and Blender over the next few years. I believe this will enhance my skillset and help me become a more well rounded engineer.

I've had many people suggest in the past that I should start up a YouTube

channel about the stuff I do. I have considered this proposal, and am willing to purchase a video camera as finances allow. This also contributes to my want to learn how to use After Effects and Premiere.

In an attempt to further my documentation I downloaded a program that takes screenshots every 5 seconds, and stores them. I did this with a view to creating a stop motion of me animating the robot. This will hopefully give people an idea of what goes into me programming my robot. I have run this process, and captured some 7000 frames. I will stitch these together in the appropriate software when the time permits. I will upload this to YouTube.

Currently the code that I've written for extracting movements from Blender works, but it can be improved. Namely, because I couldn't originally figure out how to extract bone rotations from the Animation UI from Blender, I had to animate the routine, and then play it back in the Blender Game Engine. This is quite suboptimal, and as such today I researched into getting the bone rotation information straight from the Animation UI. I asked around on the Blender python API IRC channel. I struck up a conversation with a very helpful gentleman...



This lead me to writing the following code:

```
>>> def getAngle(b):
...     rotation = b.matrix.decompose()[1].to_euler() #[1] is the rotation quaternion
...     x,y,z = math.degrees(round(rotation.x, 5)), math.degrees(round(rotation.y, 5)), math.degrees(round(rotation.z, 5))
...     return (x,y,z)
...
```

This code returns an Euler XYZ tuple of the rotation values in degrees, very similar to the one I got from my Game Engine code.

The advantage of getting the rotation values in this way is that I can now have the robot respond in real time to moving the robot around on screen. In other words, I can move the robot around on screen, and have it do that in real life. This is as opposed to having to compile the animation, ship it to the game engine, replay it - gather the rotation values while it's replaying, etc. Also this feature will speed up the time it takes from finishing animating the routine, to testing it on the robot in the order of magnitudes.

5.5 September 12th, 2015

Today I've been working on finalizing the routine music. I've decided not to use *"Bang Bang" - Nancy Sinatra* in lieu of going back to *All by Myself - Celine Dion*. The reasoning behind this is that I wanted distinct sections in the routine where the robot shooed me (Beat it), and more importantly a good section where we feel sorrowful for being apart, and want to come back together. Bang Bang was excellent for prompting sorrow, but I would need another song for "getting back together", whereas All by Myself has this "built in" - it has a sorrowful part, and then a getting back together part. I chose this because of time restrictions.

I've switched from using Audacity to using Adobe Audition (30 day trial). I've really enjoyed learning how to use a new tool. Audition's quality is outstanding, and I look forward to using it in the future. I've finally settled on the audio I want to use.

I'm also going to install Rasbian on a RPi tonight. Hoping to "upgrade" the old brain to this new one while I'm improving software and electrical systems. When I say upgrading, I'm going from RPi B to RPi v2.

Also see: *RPi*

On a completely separate note I've been preparing for a talk I'm giving in December at a camp. I've been thinking about what I want to talk about. Through thinking about the talk I've been trying to figure out why I procras-

tinate tasks. I'm not satisfied with the answer, but I think I procrastinate because I have fear of "the unknown". If I commit to completing a task, I don't know how it'll turn out. If I pretend the task doesn't exist, "maybe it'll go away" (which I realize is completely rubbish). I think the best way to combat this is by realizing that I can learn just as much from failure than from success - and that the time commit to a worthwhile task (no matter the consequence) will yield greater results than wasting time on Facebook, or watching movies.

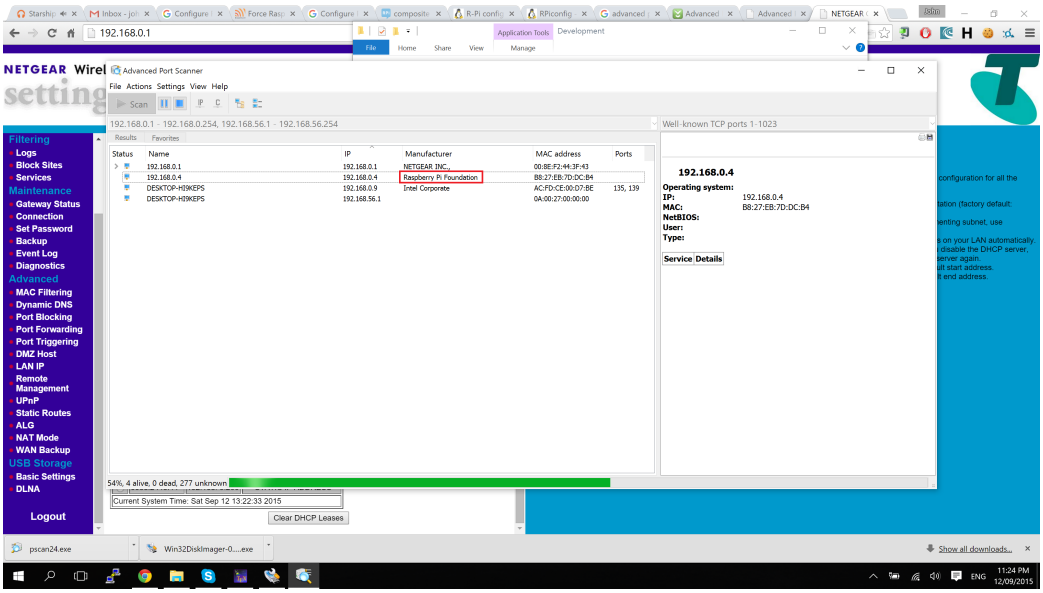
I don't think that's what I'll talk about though, I haven't had enough experience on it to talk about it. Instead, I've been looking at talking about humility. A lot of the guys coming on the camp are a little younger than myself, but are up and coming (aspiring) leaders. I'm thinking of talking about humility. Although I'm far from a humble person, I think I might have more to talk about than procrastination - I have much experience at being proud, and it's destructive properties. RoboCup has caused me to be very proud - and has humbled me as well. I'm inspired by humility in leaders, and I seek to achieve the same. Humility is the key to wisdom and knowledge.

In other news, I've downloaded the latest stable Raspbian image (05-05-2015), and in the process of writing it to the SD card via Win32DiskImager. Whilst looking at card readers(/writers) for my new laptop, I decided to go with a USB 3.0 option with the hopes of getting faster write speeds. From past experience I know that writing to the 32GB SD cards can take a while, and without researching it I was guessing that USB 2.0 may have been bottlenecking the procedure.

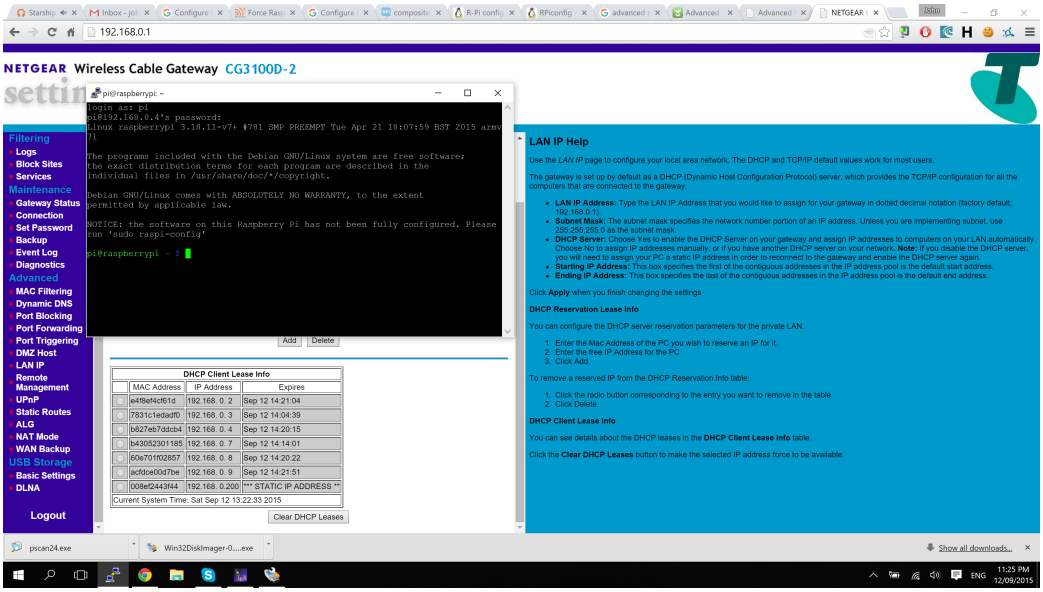
I'm going to record the speed it takes to write the Raspbian image to a 32GB disk. Time taken was 3min 47sec. Min transfer speed was 12.5MB/s, max transfer speed was 14.7MB/s, and average transfer speed was 13.5MB/s. As I didn't record speeds for a USB2.0 reader, I have no comparison. I may test this in the future.

I'm attempting to boot up the RPi v2, and having difficulty getting both HDMI and Composite output. I'm going to logging in via SSH. To find out it's IP I scan the network using *Advanced Port Scanner*. Without this software, there are two options. One is manually pinging computers, secondly logging into the network gateway/DHCP server and check recent DHCP

lease.



And we're in! Firstly I run the raspi-config command, which allows me to set some settings like SSH, passwords, but most importantly it lets me change the partition to house the entire disk.



For now I'll ditch trying to get HDMI/Composite output, I don't generally intend to have anything but the CLI running on the RPi monitor out because of the intense resource usage, and general uselessness of it.

I now need to configure the RPi install to what I need. More details about this are outlined in the Software chapter of the BaralabaBob Technical Manual.

This process typically includes me setting a static IP for the network interface, thus allowing me to use the RPi without a DHCP server (because I can just configure my network interface on my computer to the correct subnet manually.).I also install the software I use (generally just , , screen, and).

Also see: *Networking*

I got slightly distracted when I realized that I was using the old LaTeX coding convention (aka, none) in the BaralabaBob Technical Manual, so I updated to the current standard that I use.

Part III

Performance

Chapter 6

Robots

In this chapter I will list several robots used in both the Nationals performance, and other robots developed for regionals and state performances.

6.1 BaralabaBob

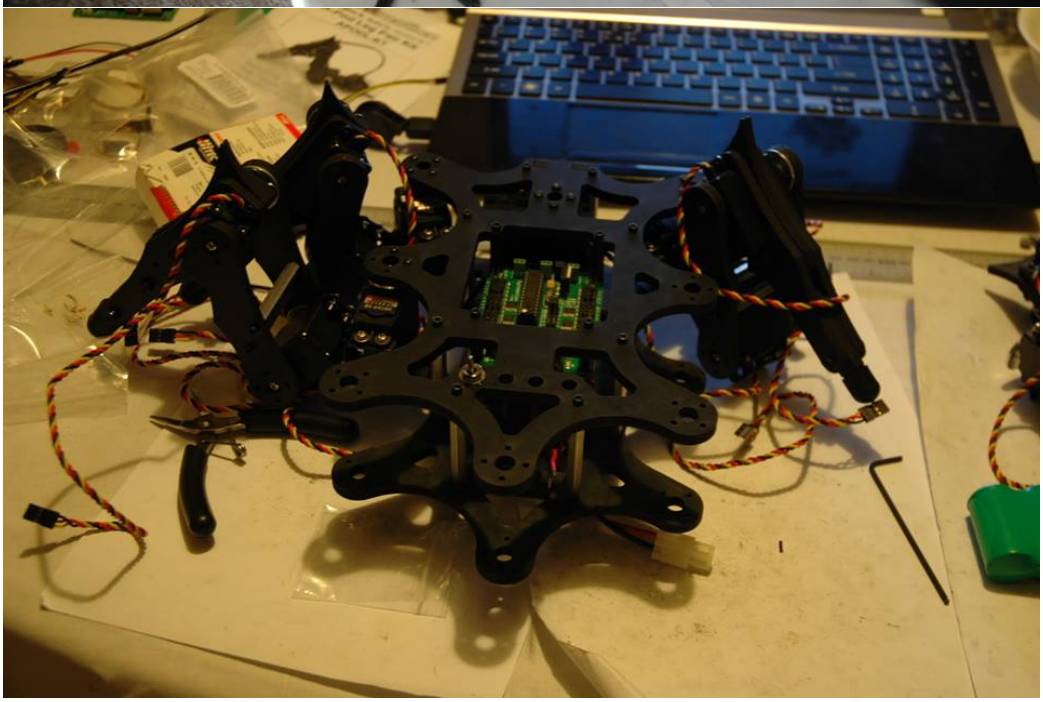
BaralabaBob is the main piece of the show, being a 6-legged Lynxmotion APod.

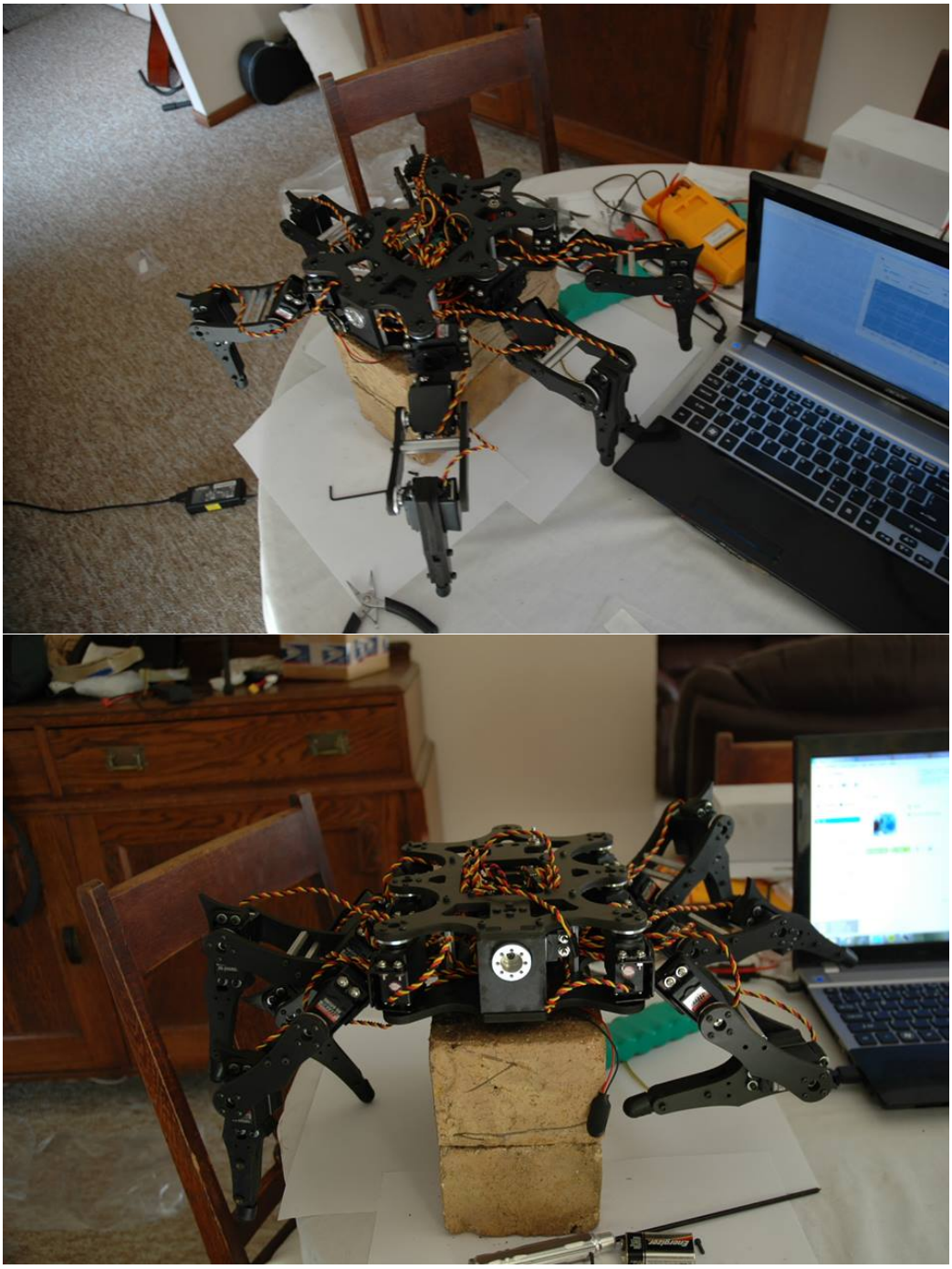
6.1.1 Technical Manual

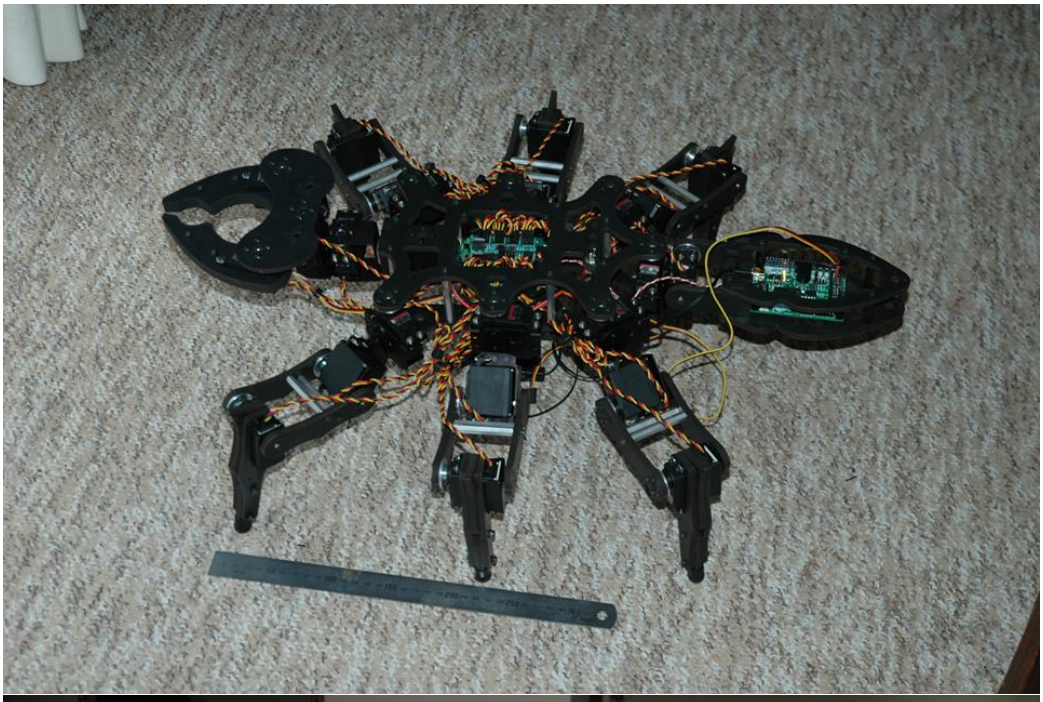
A seperate document, BaralabaBob Technical Manual, has been created, and it contains all the relevant information.

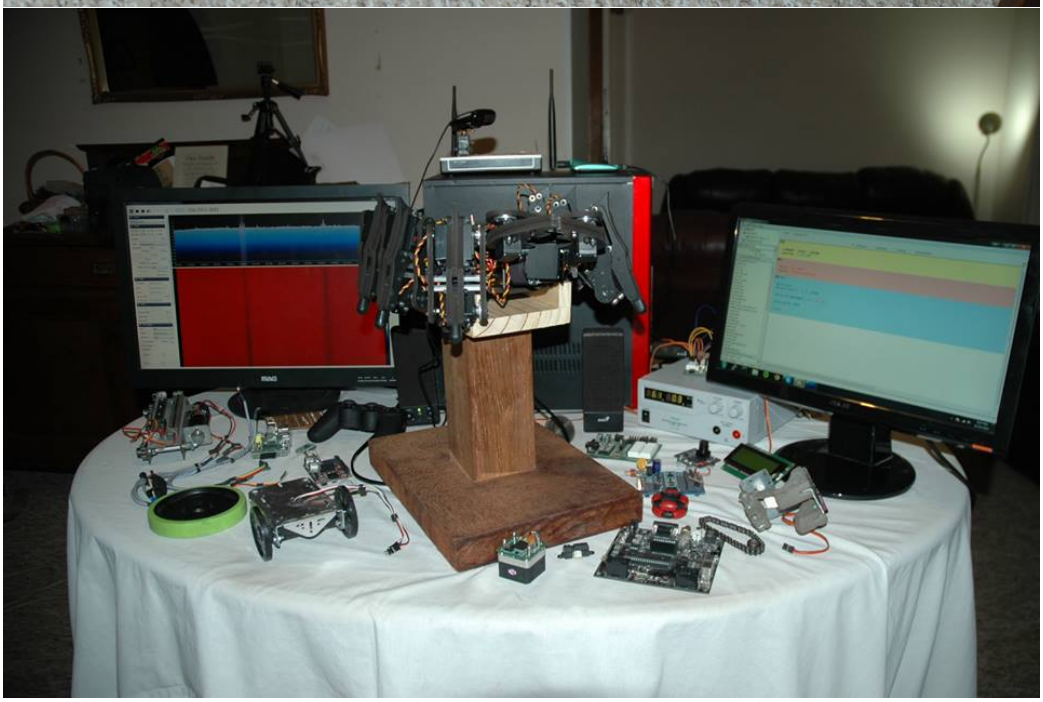
6.1.2 Construction

BaralabaBob first arrived on October 18th, 2014. Construction began, and took 6 days to assemble the mechanicals, and write the base python code, of which he still uses parts of to this day.









6.1.3 Software

Consult BaralabaBob Technical Manual for more information.

6.2 BobMobile

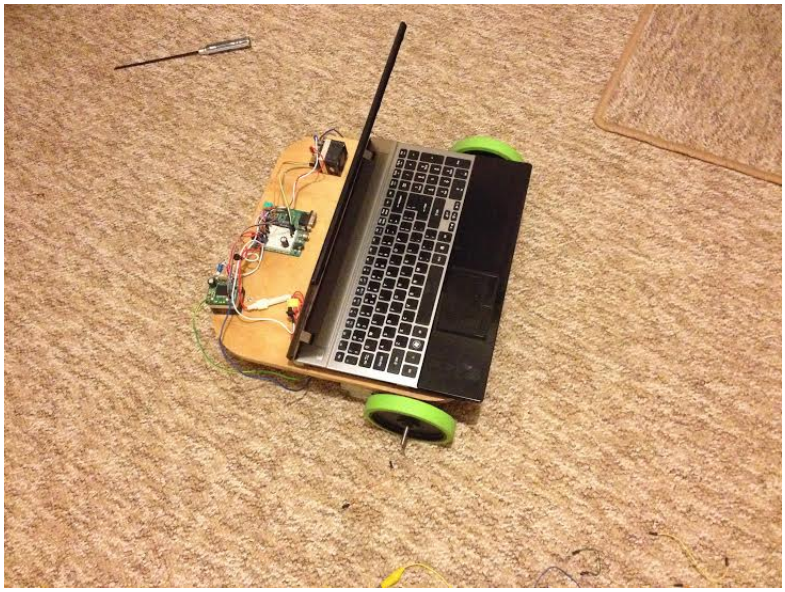
BobMobile is the robot that BaralabaBob rides out in the first part of the performance.

6.2.1 History

Whilst living in Baralaba, John became known to some locals as the "robotics guy". One of these particular people was good enough to give us some spare motors out of an electric car seat that he didn't need.

The motors were quite good as they had a good speed for a mobile base, as well as having good torque. However their axle had worm drive on it. I took it to a local engineering firm, who kindly agreed mill the teeth of the motor axles.

After having brackets built for the motors, John mounted the motors on a small wooden base. His intention was to use this base for general experimentation with bulkier systems, such as laser scanners and Computer Vision using a laptop. This robot's name was Beaker.



As I've talked about in the Goombungee Show chapter, Beaker was used as an interactive - being integrated with the HID devices of a Joystick, and the Microsoft Kinect.

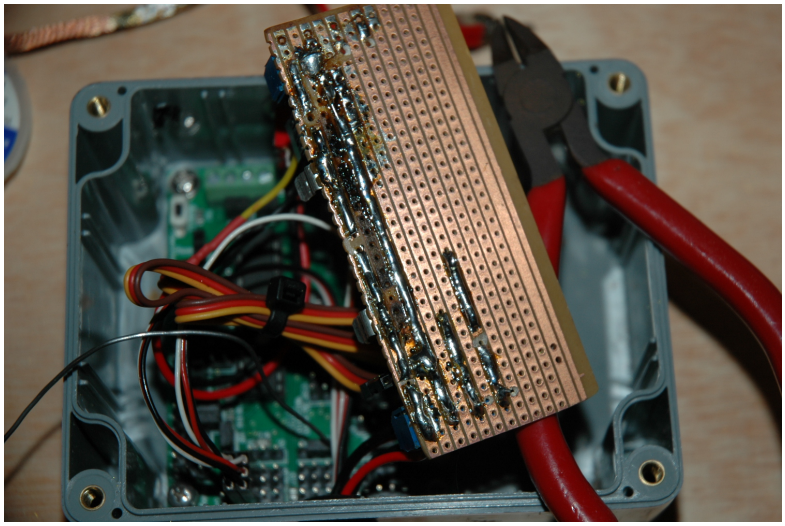
I wished to have BaralabaBob ride out on Beaker for RCJA QLD State Championships. Beaker was too small for the hexapod to fit on, so I widened the base, and added a second deck. At this point it's name changed to BobMobile.

6.2.2 Controllers

Originally BaralabaBob shipped with Lynxmotion's BotBoarduino. I soon replaced that with a RaspberryPi, but decided the BotBoarduino would be good for the BobMobile.

Also see:*Controllers*
The motor controllers are the HB-25 motor controllers from Parallax Inc.

Find below a picture of when I made a mistake when soldering the powersystem on Veroboard. The whole copper track burnt(?) and peeled away from the PCB.



6.2.3 Software

Beaker

All software for Beaker can be found here:

`https://github.com/boar401s2/Beaker`

This software includes joystick and should include Kinect code. It was mainly used for the Goombungee Exhibit. The code is written in a variety of languages including Python, Bash, C#, SPIN, and SPASM (SPIN Assembly).

BobMobile

The BobMobile code can be found here:

`https://github.com/boar401s2/BobMobile`

This is the software used for the competition. The code is written in Arduino.

6.2.4 Decks

Bottom Deck

The bottom deck is for the hexapod to stand on.

Top Deck

The top deck was made in anticipation for future expansion to using the Laser Localization System, and Helicopter Systems outlined in the Research and Development chapter.

6.3 Stage

The "stage" is the PVC frame with black covering, and red curtains used in the RCJA QLD State Championships, 2015.

6.3.1 Sub-systems

The Stage as a whole has several systems linked together - the Fog, the Lights, the Curtains, and the Button-sync systems.

See Also: *Research and Development*

The Curtains

The curtains are arguably the hardest part of the stage to implement - but perhaps the most spectacular.

The system works on a series of pulleys and string, along with a counter-weight.

I didn't have time to install an end stop microswitch to tell the system when the curtains were open and closed. Instead I used feedback from the current monitoring system on the ESCON 36/2 drivers from Maxon. The drivers send back a signal to me when the motors are straining, which gives me a good indication of when the curtains have reached the end of their limit.

6.3.2 The Fog

See more about the fog in it's section in the Research and Development Chapter.

6.3.3 The Lighting

See more about the lighting in it's section in the Research and Development chapter.

6.3.4 Button-sync

Find out more about button-syncing in the lighting section in the Research and Development chapter.

6.3.5 Design Considerations

I designed the whole stage with the concept of portability - I needed to fit it in the boot of a car, and, should the need arise, in the baggage compartment of a plane.

Because of these reasons, I designed it to be collapsible, and use common PVC tubing.

Chapter 7

Props

See:*Routine Development: Props*

Part IV

Routine Development

From very early on I recognized that a RoboCup Dance routine had to have an interesting routine. It's not all about tech, it's also about the entertainment factor.

For each level of the competition I have developed a different routine. I always had in mind that I wanted some sort of storyline to the performance to provide interest, and draw the audience in. For Regionals and States I didn't create a particular storyline, but for Nationals I've been able to implement a good storyline.

For Regionals I had a Christmas theme, using a Christmas tree and presents as props. I animated the Regionals Routine in around 2 days. I was animating the lights and Christmas tree up until the night before. I feel that the Regionals performance has been the best to date.

For States I had no particular theme. I didn't really like the States routine - it felt too much like a mashpot of random things that were disjointed and didn't fit together. I feel the reason for this is that I had created the "Stage" prop and based the routine around the stage "Stage" prop, rather than basing the prop around the routine. It felt rather disjointed, and didn't flow. Rather than the "Stage" prop adding to the performance, I felt it subtracted from it. Along with this I had only allowed myself a day to animate the robot, which was not enough to produce a quality performance. You may be wondering why 1-2 days was enough for Regionals, but not for States - and it's because for Regionals I had the benefit of months to think about what I was going to animate, before animating it. For states I had to come up with what I wanted to animate, and animate it within a day.

For Nationals I have a very specific storyline. In essence the Nationals routine is the essence of what I wanted my routine to always be like. The performance tells a story of a robot and it's creator - a story of rejection and reconciliation. After the states.

7.1 Parts of a Routine

There are many parts that go into a RoboCup Junior Dance routine, each are incredibly significant - and whilst by themselves they may not seem particularly significant, as a whole they make up a polished routine.

These parts include Audio, Robot Performance/Animation, Human Performance, Props, Story, Stage and Presence/Audience Interactivity.

It's important to keep all these factors in mind whilst developing a routine - to gain the most optimal result.

Chapter 8

Audio

8.1 Audio Uses

Audio is obviously one of the more important parts of the routine. It can convey significant emotion that the robot by itself cannot. Many times I show people my robot doing a RoboCup performance - but the performance is rather dry in of itself, because the Audio really enhances and provides an explanation to what the robot is doing.

The real advantage with the way I constructed the robot's development environment is that I can very accurately animate the robot to the music. This synchronization between the robot and the music can be used with great effect together.

I've also been able to leverage the sync advantage by adding in sound effects to the track - such as a roar, and then having the robot roar in sync to the effect. I've also been able to use other effects, such as having the robot sigh, and use "robot motor sounds" to exaggerate it's movement effect.

Obviously audio can also be used to tell a story, as I'm doing in Nationals. I was inspired by using short snippets of music to tell a story by the following *Britain's Got Talent* act:

`https://www.youtube.com/watch?v=FP7wN301yjc`

I didn't want to be restricted to one song - or even one genre in my performances, and that's why I opt to use a mix of songs. Especially at Regional level people commented that using different genre of music in the performance, in a short period of time really showed off the strengths of the robot - in it's timelessness. Another advantage of using different genres of music is that if one particular song doesn't appeal to some members of the audience, the next genre might.

8.2 Audio Selection

For every 10 people choosing music for a RoboCup Routine, you'll get 11 different opinions. Because of this I have determined that whilst it's wise to listen to all opinions, ultimately I need to make the decision based upon what I like. Often if I like the selection of audio, in the end, other people do too.

On occasions I have selected music or sound effects that didn't particularly compliment the performance, and more than often other people warned me of this. Typically these are small things though, and overall the selection of audio has been complimented. For final decision making, I take other peoples views, consider them, and then make the final decision based upon what I like.

I have found that well known music with a beat are typically the favorites. Examples of these are *Gangnam Style - Psy*, *I Feel Good - James Brown*, and *Stayin' Alive - Beegees*. On the other hand, audio such as *Pink Panther Theme*, and *Baby - Justin Bieber* weren't nearly as popular. If it's something you can tap your feet to, it's something that'll work.

This is what has caused me the most worry about my Nationals music selection. I've picked a mixture of music that has a good beat, popular, and wel known - but my entire routine is not comprised of it. For example, my opening song for Nationals, *Create/Destroy - Art vs. Science*, is not well known, however has a good beat (and I really like it). *All By Myself - Celion Dion* is well known, but doesn't have as much of a beat. I'm mainly worried because I have none of the songs that have proven their worth previously

(Gangnam Style, I Feel Good, Stayin' Alive, etc).

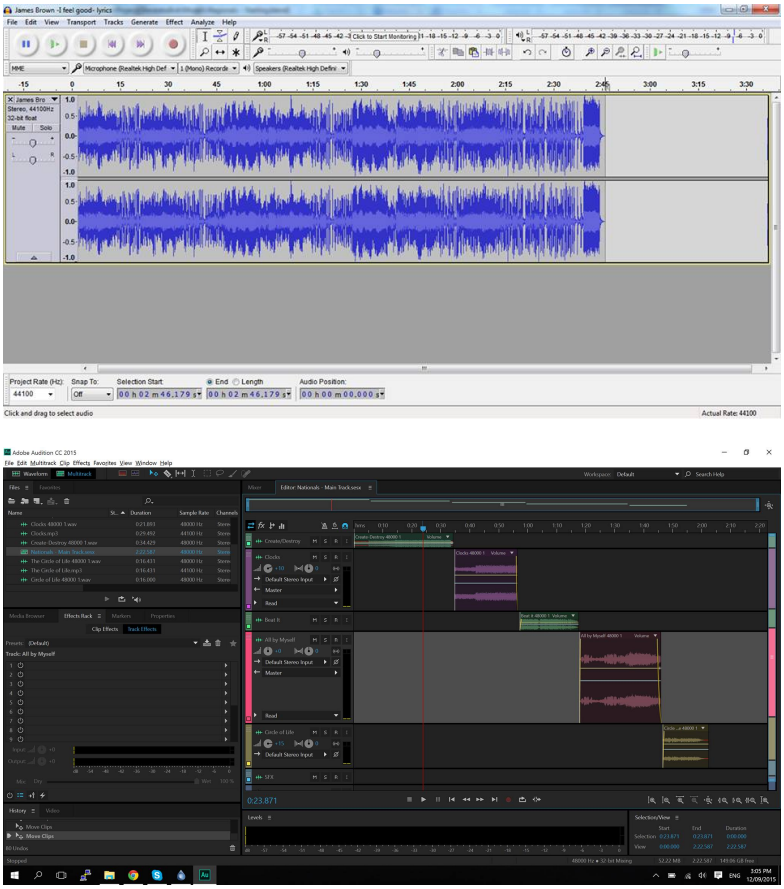
This being said, the purpose of this routine isn't to get the audience bobbing their heads - but rather to convey a story, which I feel the selection of music does amply. In short, I'm taking a risk by experimenting - but I think it'll pay off. In the past at Regionals and States my experimentation has typically paid off in my routine being quite unique.

That being said, here's the final selection of music for Nationals (in order):

1. Create/Destroy - Art vs. Science
2. Clocks - Coldplay
3. Beat it - Michael Jackson
4. All by Myself - Celion Dion
5. Circle of Life - Lion King Theme

8.3 Audio Editing

If I'm to have multiple songs in the routine, then I need some way to cut them all together. For Regionals and States I used Audacity. For Nationals I moved to using Adobe Audition. Both these are fantastic tools, and a pleasure to learn. Although I'm not particularly skilled in Audio editing, I have did a short course on a camp once, the knowledge of which I was able to employ.



The biggest problem in editing songs together is figuring out when and how to cut a song. Obviously I can't string 5 full length songs together, so I need to cut out the parts I want. Typically I like to keep each individual song to 20-30 seconds each. I try to find a place which I can cut it in that vicinity.

I employ two major techniques to gain a cleaner cut if where I'm cutting the song is not clean in of itself:

8.3.1 Fade In/Fade Out

This one is somewhat self explanatory, fading in and out is a great way to soften unclean cuts.

8.3.2 Cutting Parts of the Song Together

Sometimes you're able to cut the beginning of a song to the end. The advantage of this is that you get a clean start (as you are using the start of the song), and a clean end (because you are using the actual end of the song). If you are very careful, you are able to get these two cuts and combine them with no noticeable cut.

An example of when I used this was in the *I Feel Good* audio. I cut the start of the song, straight to the end - cutting out the middle lyrics and chorus. I cannot distinguish where I have cut the music together - even when I'm looking for it, so the audience doesn't suspect a thing.

Chapter 9

Robot Performance/Animation

The way that I designed the development environment for the hexapod, I am able to animate the routine to the music using the open-source 3D modeling and animation software, Blender.

9.1 Animation

To give you the best overall idea of what the process of animating the robot looks like, I've compiled a stopmotion video of me animating a movement. This particular movement was for States, and I was animating the robot stepping off a raised platform, onto the ground. This took around 2 hours for me to animate.

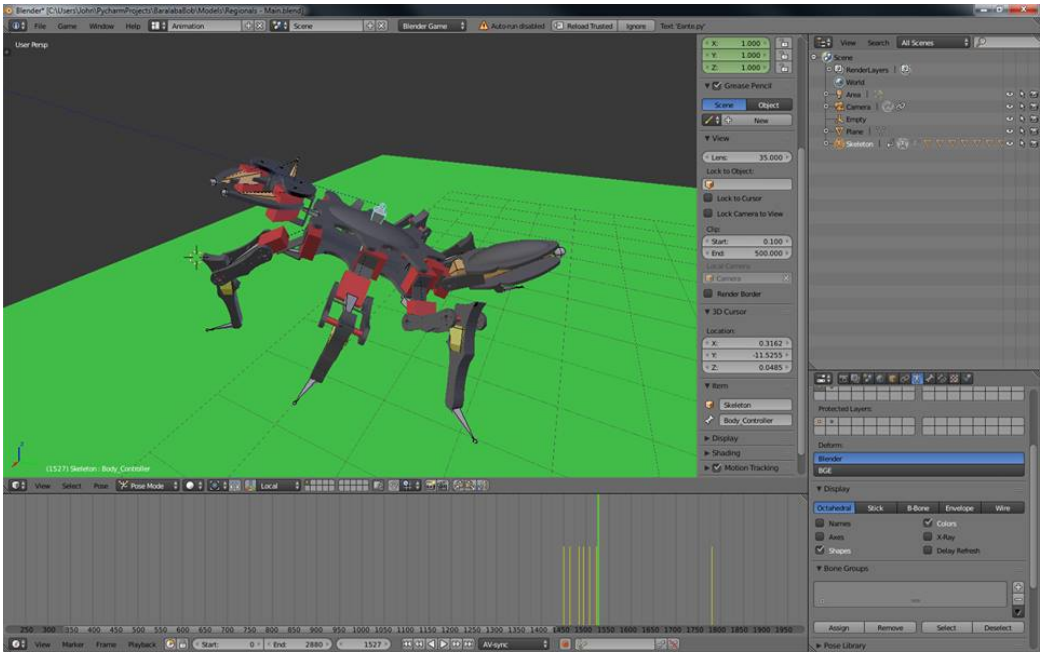
`https://www.youtube.com/watch?v=o0-15CS6urg`

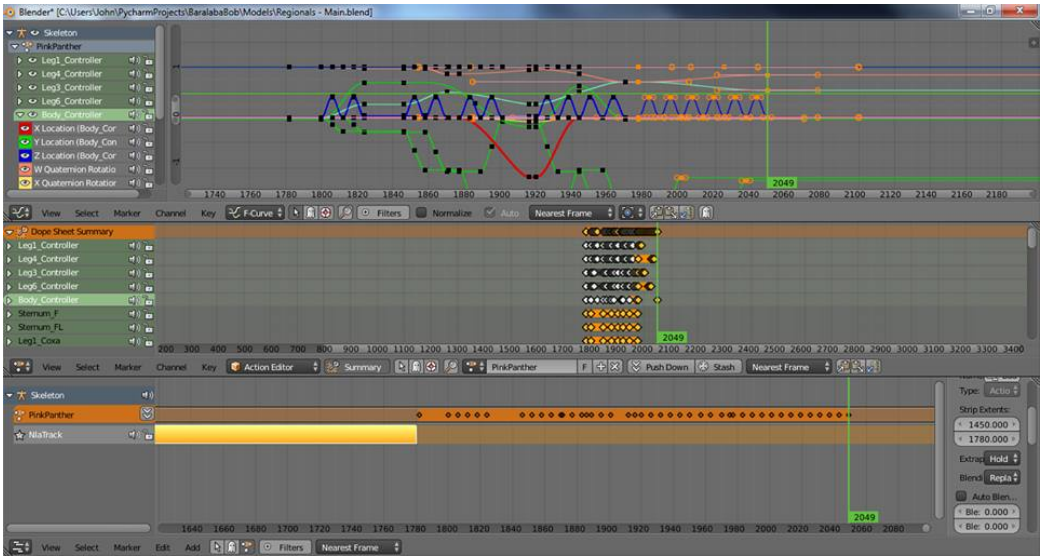
All things considered animation isn't the most time consuming activity, but I might venture to say that it is the most irritating. Often you are listening to the same music over and over, and doing very repetitive tasks. Often you have to delete half an hours worth of work at a time. It's not uncommon to throw away several seconds of content (which could have taken hours to create).

You can do many things when animating that won't work in real life - you can make the robot jump in Blender, but that doesn't mean the robot

will jump in real life. You need to have a good sense of how the robot works, physically, to be able to animate it. This is not all bad though - if you are familiar with what the robot will do in regards to how you animate it in Blender, you can achieve some pretty amazing results that you wouldn't normally be able to achieve if you animated the robot conventionally.

The soundtrack has to be fully created before starting animation, as I animate to the music. To do this I add the soundtrack to the timeline in Blender, allowing me to hear the audio whilst I'm animating, allowing me to animate to the music.





9.2 Performance

Before animating the routine for Regionals, I had thought of a lot of movements which I could put to music - allowing me to develop them quickly. For States I hadn't thought of movements - I was animating and making up movements on the fly.

Some considerations in the development of the robot's routine is making sure I use as much of the floor space as possible, having lots of unique movements to maintain interest, making sure the movements fit the audio, and ensuring that the movements aren't straining the robot (causing motor failure).

Also see: *Motor Failure*

It's very difficult to walk the robot around the entire floor space, as it takes a while to have the robot walk anywhere. Part of this is my not wanting to strain the robot by having it walk fast, but as time goes on I get more and more adventurous with how far I push the robot. The problem of movement speed still presents itself.

One of the sub-difficulties is that turning the robot is very slow. It takes

a minimum of 3-4 individual turn sequences on the robot to turn a full 90 degrees, each turn taking 0.5-2 seconds. Multiply 4 by 2 seconds, and you quickly run into a very time consuming movement, at 8 seconds - and that's just to turn 90 degrees. I prefer to have the robot walk sideways, than turning sideways and then walking in that direction. The advantage of a legged base is that you don't have to face the direction you walk, so you can do this.

Having unique movements to provide interest isn't particularly difficult to create, but it is often difficult to animate and time consuming to integrate into the routines. It often takes many hours of continuous effort to animate a single 10-15 second segment.

Chapter 10

Human Performance

From the outset I realized that a good performance will include good human interaction. As expressed in the Dance Performance score sheet, the interaction should not detract from the robot itself - but I believe a performance can be rounded off with a good human performance.

I'm naturally not a person who defaults to dancing or acting on stage - however because of requirement to do so, I recognized that I needed to put aside my illogical fears, and perform. The result of this is that I made an agreement with myself that I would put aside any fears or doubts about performing on stage, and that I was just do what I had to do, and put my all into it.

I believe that because I made this agreement with myself, I was able to give a half decent performance, which compliments the robot. As I've mentioned previously, I don't want to detract from robot's performance, but at the same time I try to create a human-robot interaction throughout the performance in the setting of a story to increase the entertainment factor.

Often my performance comes from what I want the robot to do. Often the comical factor of the robot is it showing attitude to me. It's quite simple to express emotion in reaction to the robot. For example, I can look surprised when the robot roars at me. Just having the robot perform is only one side of a conversation - having a robot respond to the robot adds the second side of the conversation.

10.1 Human Interaction Incidence

There was one incident at State titles which caused a problem I hadn't anticipated. The issue was that how my routine was developed, I would stand to the left of the robot for the duration of the routine - as the robot would face me at times throughout the routine (ex. roaring at me).

On the day, I didn't have a good way of solving this problem (except for standing on the other side of the performance, and hope for the best), which is what I did.

For Nationals I'm conscious of this problem, but as of yet haven't found a suitable solution. The best solution is to create two instances of the performance - one for right side, and one for left. Unfortunately this effectively doubles my work, which I may not have time to do. I have a theory that I *may* be able to "mirror" the actions in blender, and export the mirror, but I haven't been able to test this yet.

For Nationals I will advise the judges previously to the nature of the positioning of the routine, and also I'll try to keep out of their view as much as possible.

10.2 Stage Presence/Audience Interactivity

Having good stage presence can also really round out a routine. Having stage presence means that you are able to command and hold the attention of the audience, simply by your demeanor and confidence. This can be achieved by confidently introducing your routine, and talking directly to the audience.

Audience Interactivity can include asking the audience to clap along, sing, or maybe even dance! This helps draw the audience in, and creates a fun and vibrant atmosphere. I employed asking the audience to clap along for Regionals and States. Because I'm doing a different kind of routine for Nationals, I'm not sure if I am able to include this kind of interactivity.

Chapter 11

Props

Props are to enhance a routine. This is a fatal mistake I made in States.

For Regionals in Rockhampton, I designed the entire Christmas routine, including audio, animating, etc. All the while I was thinking about what props I could use to enhance the routine. This worked really well, and I got a lot of really good props I used to great effect.

However, for States I got caught up in trying to create excellent props. The result was that I got a really good "stage" prop - but I put all my time into designing the props, and not enough into routine development. The result was that I created the routine around the props, and not around the music or story.

This meant I couldn't animate the routine as well as I wanted, and the story/routine was disjointed and not flowing. I learned my lesson from this, and for Nationals I have put my efforts into creating the routine.

I've found that as I create the routine, I naturally think of funny and engaging props that will enhance the show. When it comes time to adding props, I don't have to put too much effort in, as I already know what I want.

For Nationals I had created the storyline that I wanted and as I was working on animating I realized that having a lab-coat, safety glasses, misc tools scattered around, and some technical panel with dials, buttons, and lights would enhance the routine. Most of these things I have lying around

at home, and I can easily put them on a plane.

The only thing I'm thinking of working on is the panel with lights and stuff. I put a few hours aside for creating something like that, and programming it.

Chapter 12

Story

I'm a big believer in adding a story to most anything. I think stories engage the audience, and draw them in. It creates an emotional attachment to the performance, and to the actors. I like to employ stories and storytelling in things I teach, and performances I put on.

The story of this performance is loosely modeled after the story of the Prodigal Son in Luke 15:11-12 (The Bible) - and typically the creation/salvation story outlined in the Bible.

In the story, I'm the creator/father figure, and I create my son/child/robot. We are at first having fun, and enjoying each other's company - but as time goes by the robot gets annoyed, and pushes me away. The robot walks off, but through which time the father wishes that his son was with him all the while. The father doesn't care that his son did him wrong - his love transcends that sin.

After a while, the robot realizes that he was better off, and happier with his creator. They both come back together again, and lived happily ever after!

Part V

Research and Development

Work on the first systems of the Regionals performance started in October, 2014 when the Lynxmotion APod arrived in the post. Since then a steady stream of technological ideas have been formulated, researched, and in some cases - created!

Each of the following chapters outlines a major line of researched I followed in the formation of the routine.

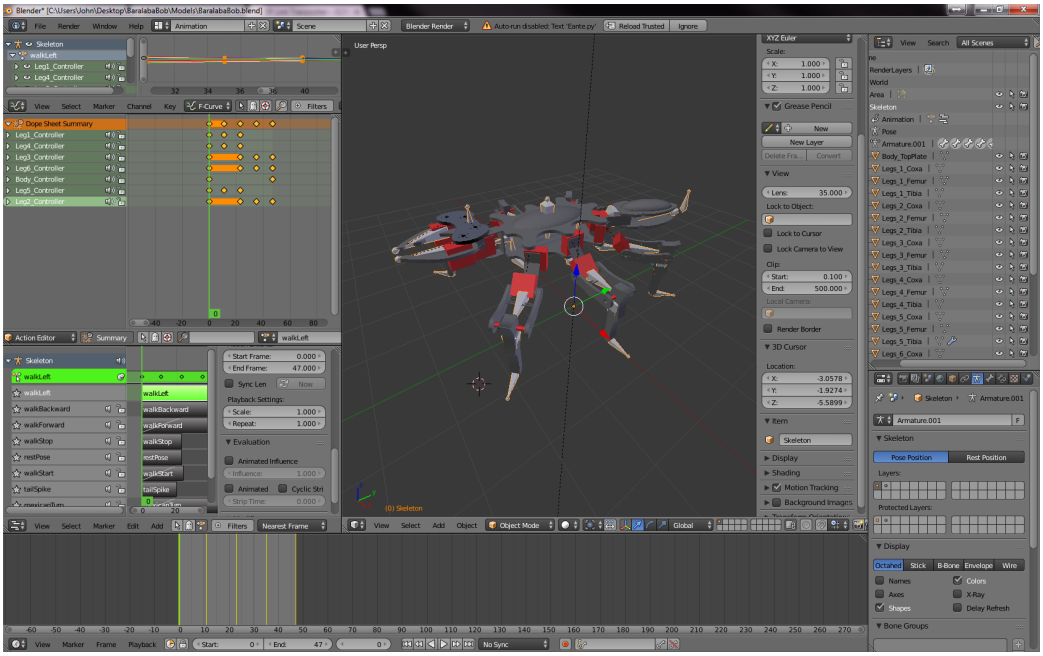
Chapter 13

Blender Animation

Before even acquiring the hexapod, I had formulated the idea that to easily create any form of dancing hexapod, you'd need some sort of 3D animation software.

Also See: *BaralabaBob Technical Manual: Blender*

After acquiring the hexapod, and started programming it - I once again reaffirmed the need for this software.



13.1 Base Software Choice

Writing 3D animation software from scratch would be reinventing the wheel - so I looked for an existing 3D modeling and animation program that I could use as the "base software" for this idea.

The two best options where Maya and Blender. Both of which support python scripting, and can be downloaded freely as a student.

I ended up choosing Blender for many reasons, a few of which listed are listed in the following: it being much smaller in file size, open source nature, and being written in python.

13.2 Model

I needed a scale model of the hexapod in a digital form to animate in Blender. A model was created in TurboCAD 2011 to scale, and exported in the 3DS format to Blender.

After being colored appropriately, I added an armature to the model that correctly mimics the joints in the actual robot. The model is attached to this armature - as I move and animate the armature, the model also follows.

13.3 Scripting

After animating the model, I needed to export that data to a form that the hexapod can playback. This is where blender's scriptability comes in.

I first attempted to write a script that would extract the armature data when told in the standard blender interface. Unfortunately for some reason I couldn't achieve this (I forget why...). Instead the script, named Eante (pronounced Yaantay), cycles through the selected action and records all the important keyframes.

Eante then saves this information in compiler_config.dat, ready for the next script, Dumper.py.

I found that I was able to easily read the armature information in Blender's Game Engine mode. As such I wrote a script that would open the compiler_config.dat, playback the selected animation, and then record any changed joint positions at the specified keyframes.

Dumper saves this information in a .act file. I then manually copy the contents of this information to the actions folder on the hexapod. In the future this will happen automatically.

13.4 Linear & Bezier Modes

Outlined above (where the system only captures bone positions on relevant keyframes) is considered Linear mode. When playing back on the robot, the movements aren't very smooth - there is a constant speed on each movement, and no sort of ramping.

When creating keyframes, blender automatically interpolates the position of keyframed objects between frames with a cubic bezier curve speed modifier.

So in the animation you would see very nice smooth ramping, whereas on the robot the movements would look very, well, robotic and stiff.

At this point I decided to create "bezier mode". When Eante compiled the keyframes to check, it would present the option for bezier mode. If you selected bezier mode it would write said setting to compiler_config.dat. When Dumper reads that bezier mode is enabled, it ignores all keyframe data, and simply saves the armature angles every 3 frames (this is configurable, of course).

The idea of this was that the robot would have smoother movements. The concept was sound, but execution had a few issues. Firstly I hadn't properly implemented timing.

If you think about it, when a .act files specifies the angle of a joint at a certain frame - the joint has to be at that position by the time that frame has been reached. Instead, I told the joints to reach their positions by the next frame - wreaking havoc within the bezier system.

I created some systems to counter this, the result was some very smooth movements, but timing accuracy suffered.

Since then I have fixed the timing implementation, and the bezier mode works perfectly now, creating incredibly smooth movements. An alternative was considered where I implement the bezier curve calculations within the processor on the fly. That way, you save the amount of file each action takes up, by only storing keyframe data, and interpolating positions to a function of a bezier curve manually. This is on the TODO list, however due to fixing the timing issue, I may not deem it worth my while to implement it.

Chapter 14

Laser Localization System

This was one of the unfortunate systems that were cut in liue of lack of time. I have a large wheeled base which I was going to incorporate into the routine - however unfortunately it doesn't have the best dead-reckoning ability (one wheel turns significantly faster than the other).

Whilst this can be fixed using speed ratios in code, I didn't feel it would be accurate enough for the purposes.

Of course, another solution would be to use rotary encoders, however the resources and opportunity for me to install them didn't present themselves.

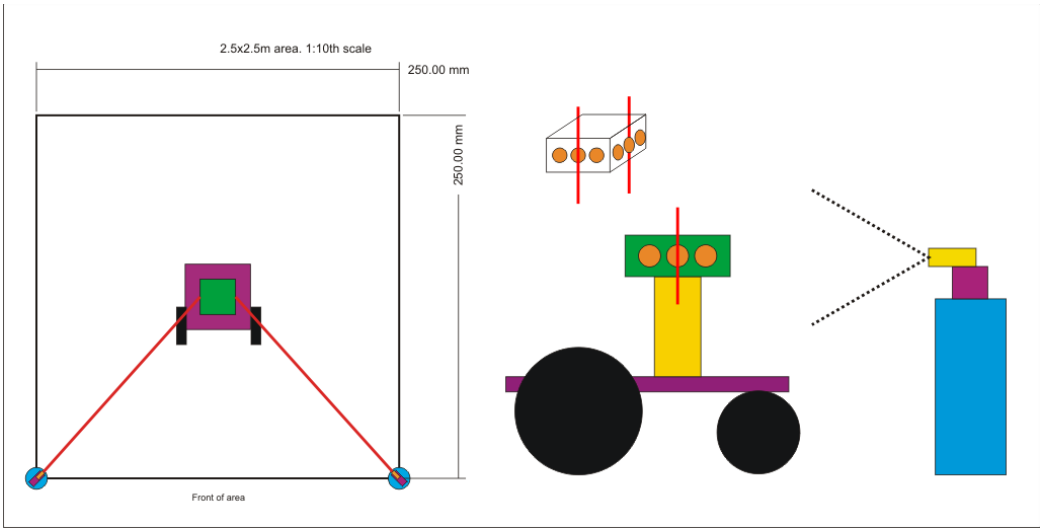
As I was intending on using the system in Regionals, I developed this cheap, quick, easy, and accurate way of localizing a robot in the dance area using equipment and sensors that I already had, along with ¡$50 of parts.

I was ready to build this system - I had the parts, the maths, parts of the code, and the concept ready to go - I just lacked the time to actually put it together!

I originally *posted a question* asking the feasibility of said system on the Parallax Forums, and got lots of interesting thoughts to help with this idea.

It's possibly also worth mentioning that I had designed a completely different system based on IR, but decided not to use it. For the sake of brevity, I'll exclude going into detail about it.

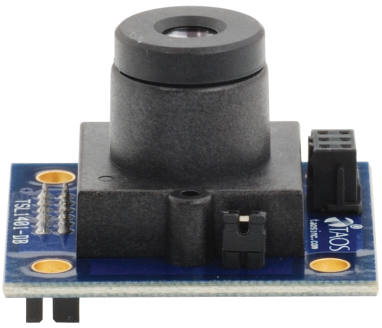
14.1 System Concepts



In the figure above, the outer 2.5m area is the bounds of where the robot is allowed to go. The system will not have to track outside this area. The big purple square is the robotic base I'm trying to track, there are two black wheels either side of it. Now focus your attention to the circles on the bottom left and bottom right of the 2.5m area. The purple squares at 45d are servo motors. The gold squares mounted on top of the servos are line lasers (with the lines set vertically).

Now focus your attention to the diagram to the right of the square area, you can see a side on view of the robot base. The yellow block on the robot is simply a wooden spacer. The green box on top of that is a "cube" of circuit boards, with 3 LDRs/Phototransistors/Photodiodes on each side (denoted by orange circles). (See figure above the side on robot view for a 3D representation). To the right of the side on view of the robot is the "beacon/laser scanner". The blue denotes a wooden block, purple denotes servo, gold denotes line laser.

NOTE: At this time I wasn't aware I had some TAOS linescan sensors in my possession, so I was still working with the concept of using LDRs. Since then I updated the idea to work using said TAOS linescan sensors. The advantages of the Linescan sensors were clear - faster response times and wider sensing areas.



14.2 Laser Tracking

When the system first turns on, the servo (and thus the laser) scans throughout it's range. When the middle LDR on the robot detects the laser, it sends a message over an RF link to the beacon. The laser should be focused on the centre LDR. (as the diagrams denote). Please remember that I've updated this idea to work using TAOS Linescan sensors instead of LDRs.

The second laser will do the same, although will be focusing it's laser on a different side of the "LDR cube". (since updated this to, the laser will be focused at a different section of the linescan sensor's sensing area). At this point in time the system knows the position of each beacon, along with the angle from each beacon to the robot. It will then bilaterate the position of the robot.

14.3 Issues with the LDR System

As I've amply mentioned, I resigned the system to use TAOS linescan sensors instead of the LDRs for a few reasons, mainly because of limited sensing area, and speed.

Here's an excerpt of the forum post outlining my concerns:

I'm wanting the robot to move 0.5m-1m / second. That equates to 0.5mm-1mm / millisecond. The LDRs I'm wanting to use have a size of 8mm$\hat{2}$. That means there's a distance of 12mm from the centre of the middle LDR to either outside edge (where it looses detection). That means, in perfect conditions, you'd have 8ms between the time that the robot starts moving at 1m/s, and when the laser moves off the edge LDR. Thus, the loop between the laser hitting the outer LDR, sending the signal to the beacon, the beacon processing that and moving has to be under 8ms (at the very most). For reference I'm currently using the nRF24L01+ transceiver modules, which had a response time of ¡2ms when I tested them (in unoptimized conditions).

My current testing with the TAOS sensors have been promising, but I haven't tested the system in it's entirety.

14.4 Rule Legality & Safety Considerations

Understandably there are some concerns with using lasers where there are lots of humans around, and the hazards associated with it shining in their eyes.

I approached Mr. Jason Bell, and Dr. Damien Kee with my concerns, seeking clarity. The answer I received was that I should be able to, provided the lasers meet Australian Standards.

As such, I set out to seek whether the lasers I was intending to use meet such criteria. Here's an email I composed to Mr. Jason Bell, and Dr. Damien Kee outlining my research and conclusions on said topics.

Hi Dr. Kee,

Following our recent discussion about using lasers in RCJA Dance, I can offer the following comments about the lasers I'm wanting to use:

The Weapons Act defines a Laser Pointer as a hand-held battery-operated device, with a power output of more than 1 milliwatt, that is designed to emit a laser beam and may be used for aiming, pointing or targeting.

My laser uses 5mW, but does not concentrate that energy to a single point, but rather spreads it out over a line, reducing the concentration of the beam. I rang up the Weapons Department of the Government to seek clarity in this matter, and ensure that it'd be safe enough for the competition. They have indicated that this type of laser fell outside the reasonable definition of a "Laser Pointer" under the Weapons Act.

Finally, all due care and attention will be given to using the lasers in a safe and responsible way, and the issues will be dealt with in the following ways: The line laser will be pulsed on for the minimum required time to take a reading, which is around 50ms (reducing it's overall power). The line laser is mounted within a protective covering, reducing stray laser light, and facing away from the audience at all times.

I am satisfied that if used safely, this type of line laser falls within the guidelines of the competition, but I would like your final consideration before proceeding. I don't imagine I'll have enough time before the competition to actually assemble this system, but in the off chance I do I would appreciate your go-ahead.

Kind Regards,
John Board

The response to the letter was positive, and the final ruling was that such system was allowed at a Regional level.

Chapter 15

RF Networking

Early on in the development of the system I recognized the advantage to having each MCU in the system linked. The primary option was to use RF.

This RF system would be used to start the routine, sync the robots, and aid in the laser localization.

15.1 433MHz

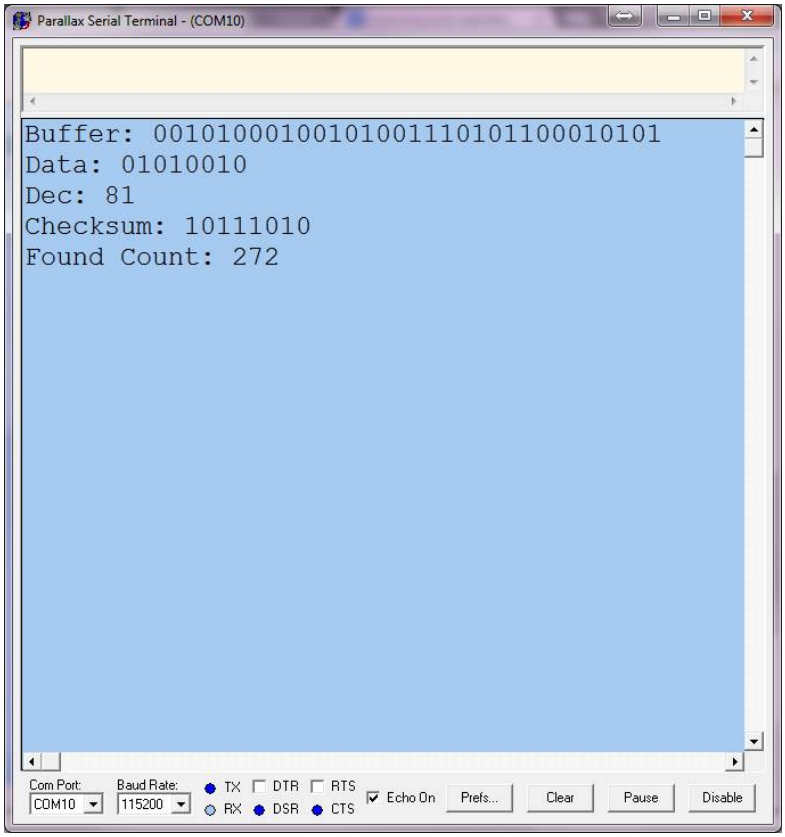
The first RF solution I experimented with was cheap 433MHz modules that you see riddled all throughout eBay. I happened to buy my transmitter *Transmitter* and *Reciever* from Jaycar.

After playing around for a little while with no success, I posed my problems on the Parallax Forums.

`http://forums.parallax.com/discussion/160707/433mhz-rx-noise`

I received incredibly valuable advice, especially from Dr_Acula. From this advice, I wrote a simple driver for the Propeller Chip, the code of which can be found here:

`https://github.com/boar401s2/JaycarTXRX`



15.2 2.4GHz - nRF24L01+

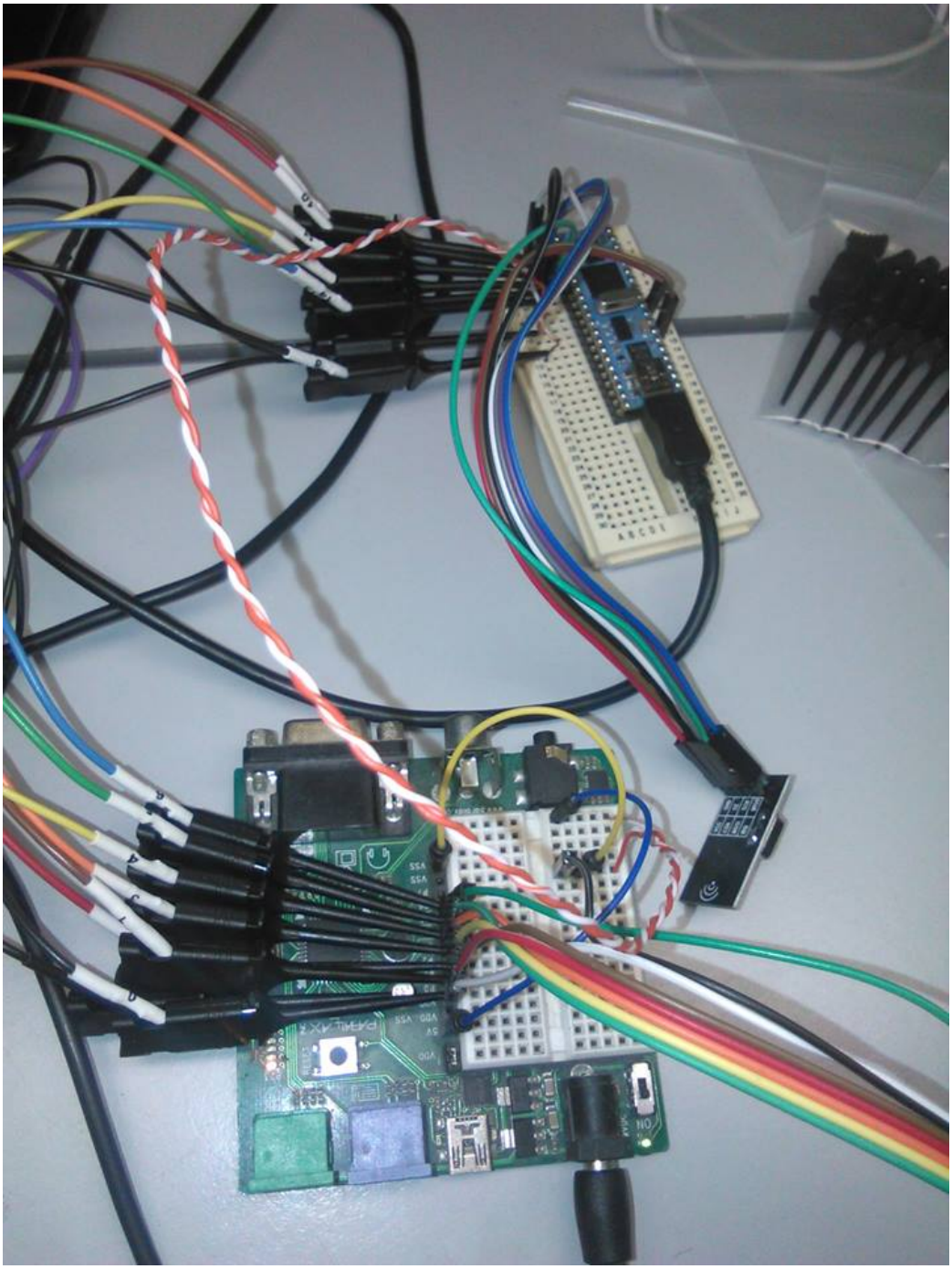
Whilst I was working on the 433MHz transmitters, I had also decided to order some nRF24L01+ modules, to also experiment with - and see which were better.

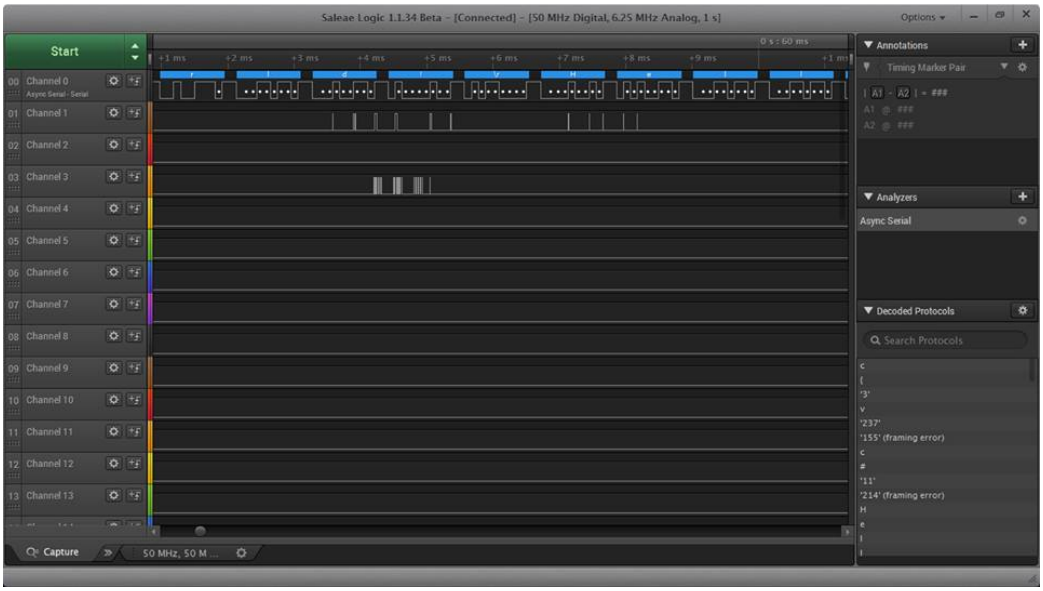
I spent a few days researching the nRF24L01+ modules, and then I started to write a driver for them in SPIN. Whilst the driver in a MVP state, there's still work that can be done on it. The code can be found here:

`https://github.com/boar401s2/nRF24L01-Driver`

Whilst writing the driver software for the 433MHz and nRF24L01+ modules, I had a Saleae Logic Pro 16 logic analyzer arrive in the post, which

significantly helped speed up the development process, and I've also been learning how to use logic analyzers.





The nature of an event such as RCJA involves lots of laptops/robots using Bluetooth and/or WiFi - which both reside on the RF ISM Band (2.4GHz). By attempting to implement a network on the 2.4GHz band, foreseeable there could be a lot of interference to contend with.

15.2.1 2.4GHz interference Solution

Two solutions I came up with was: Use the 433MHz modules, or modulate data signals into the lasers in the Laser Localization System (Personally my favourite solution, although it has many shortcomings!).

There is a third option - and that is using IR to communicate - of course this could potentially break the rules as it may interfere with other robots (such as the soccer robots).

Chapter 16

Fog

NOTE: This is the entry about Fog that I entered into the Logbook for RCJA Rockhamption Regionals, in July 2015

Experimentation was done in the realm of trying to have fog on stage for added effect. This was never intended to be entered into regionals, but rather for States and Nationals.

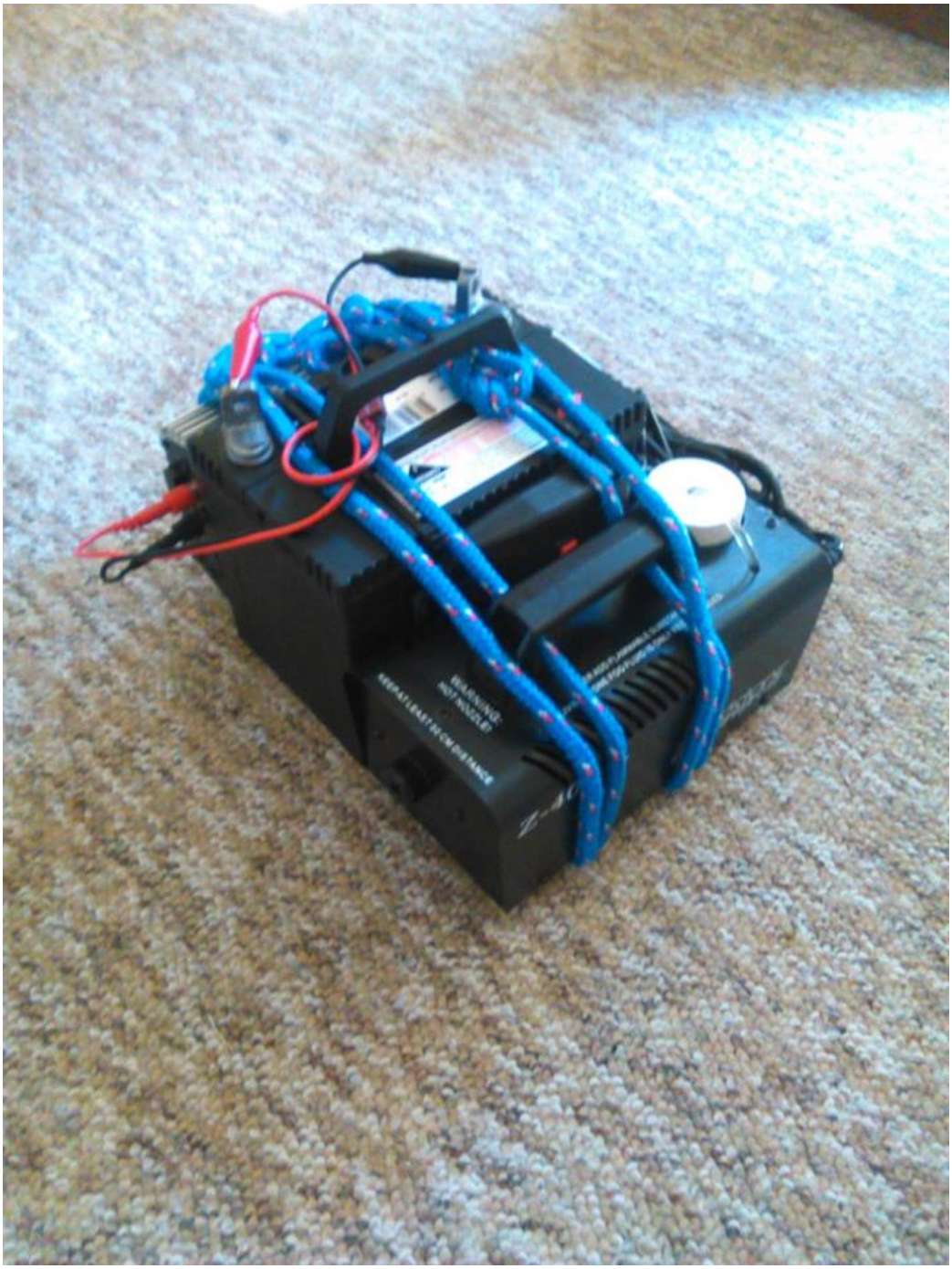
The first and most obvious is getting a fog machine, however in the rules it states that no mains power will be provided to the dance area. The most obvious solution is to use an inverter on a SLA battery, which is probably the option I'll look at implementing.

The other option is by making your own. Generally involves heating up a solution of glycerene and water (fog mixture). I experimented with mixing glycerene and water, and heated it - but if anything less fog was produced than just plain water itself. No more work was done on this concept.

NOTE: I have now obviously created this Fog, and here's some information about how I did it:

I wanted to implement a Fog Machine to add effect for whatever performance I was doing. In the paragraphs above I noted some early attempts on how I wished to achieve this. In the end I settled with borrowing a fog machine from a friend of mine, as well as a 400W inverter.

I attached the fog machine and inverter to a 12V SLA boat battery. I am able to carry this around, set it down behind my stage, and fill the stage with fog.



One problem that I overcame is that the fog machine I borrowed is 400W, and the inverter is also 400W. When turned on the fog machine slightly overloads the inverter - thus shutting the inverter down. If you are unfamiliar with how a fog machine works, it has a heater and a pump. Once the heater has heated, the pump pumps fog "juice" onto the heater, which then evaporates it.

The heater draws too much energy for the inverter, but the pump can run off the inverter. What I decided to do was to heat the fog machine up using mains power, and once the heater has heated, I switch it over to inverter power. I am then able to pump fog juice onto the heater, without the heater requiring any energy.

After fogging for a set period of time the heater needs to be heated once again - at this point the inverter stalls. I have sufficient time to pump out all the fog I need before the heater has cooled down though.

I ran a few experiments testing how long I could use the fog machine before it needed mains power again, and I found that after 2 minutes off mains power (and not being used) the fog machine is able to fog for 26.5 seconds. I feel that 2 minutes is sufficient time for me to unplug the machine, take it to the stage, set up, and start fogging before the heater cools.

16.1 Projection Mapping

A while ago I had come up with the concept of projecting images/movies onto moving dancers. This system involved some form of camera tracking - which then (using some OpenCV code) would isolate where the humans are, and then project masked images onto them.

This way you could select what images to project onto what person.

Due to lack of proper equipment I had to halt development, however some promising first tests were conducted. Since then I have acquired a camera more suitable to the task, and may take up the project again at some time in the future - but not for RoboCup.

Chapter 17

Helicopter

I'm very excited at the prospects of this system - using cheap ($30) RC helicopters and hacking their remotes to allow me to control them using MCUs.

The system basically involves two "camera beacons" which track the location of said helicopter, feed this information back to the microcontroller which then send signals to the hacked original remote control (which in then turn sends the signals to the helicopter).

A very promising, and exciting system. I would like help to figure out how to track multiple helicopters at once though.

I would be keen for this system to make an appearance at States or Nationals.

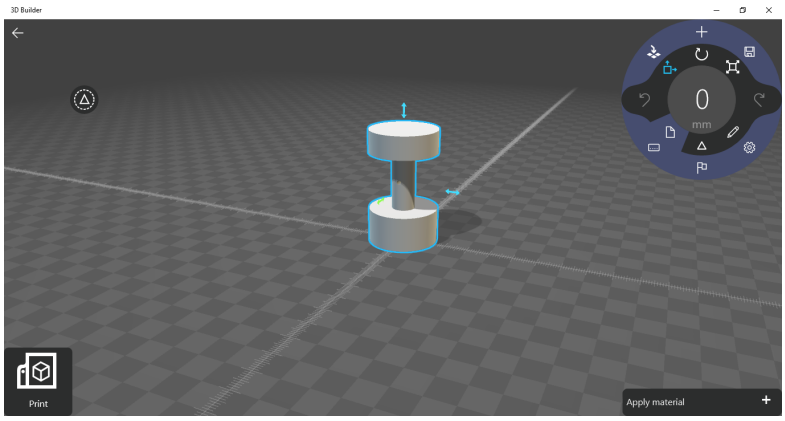
Chapter 18

3D Printing

As part of the stage setup there is a winch system - a "reel" and a motor. As the motor turns the string winds up over the reel, and pulls the curtains open.

During development I had to rapid prototype this reel. Two options presented themselves - a bolt with nut arrangement, and a 3D printed option.

I quickly developed a CAD model for the reel, and sent it off to a friend for printing. I designed the model in Autodesk 123D Design.



Whilst this was printing I looked around in the box of old robotics parts, and found a wheel hub that I created many years ago. It was a large bolt, with a hole drilled perpendicularly to the shaft into the head of the bolt. It also had a hole drilled down the shaft. The first hole is tapped. This allows the axle of a wheel to be inserted down the shaft, and then a grub inserted

into the first hole to hold the axle in place.

This also worked, but needed a ridge at the other end of the bolt to stop the string sliding off that way - the most logical option (and the option I took) was to attach a nut to that end. This solution worked great.

The 3D model was the first I had made for printing, and I had made one design mistake - that was I made the hole (for the motor axle to fit in) the same size - meaning the axle wouldn't fit in. I've learned my lesson this time though, and know what to do next time. Because of this mistake, I am using the bolt option.

Chapter 19

Inverse Kinematics

I have spent significant time working on the Inverse Kinematics implementation into the base operating system.

The general goal of the completed inverse kinematics system is that I can give the robot a location I want it to walk to, and it calculates down to every last joint angle how to get there - how many steps to take, when, how far to step, what the joint angles should be, when it should move the legs, etc.

Results are promising - I have individual leg IK completed, and now working on Body IK. Once these have been successfully merged, I can then move onto step scheduling.

I've got an demo of moving with this type of system according to the input of a joystick. It works, but barely.

Chapter 20

LaTeX

As some readers may have noticed, this document was written using LaTeX, a typesetting programming language. This is the first time that I've used LaTeXfor a complete technical document, so I consider this to be a research and development process in of itself.

I had a friend that told me I should start looking into LaTeX- I wasn't sure at first, but I picked it up and starting using it. Because of my previous programming background, I quickly took to LaTeX, and is now my preferred method of writing technical documents.

Chapter 21

Lighting

As time went by I realized that having some sort of lighting within the stage to light the fog could add a great deal of atmosphere.

Originally the plan was to use some large and expensive LED strip from Jaycar. I realized that with the time I had available, this would not be a valid solution. Whilst thinking about this problem I came up with the solution of using a powerful headlamp that I own, and then putting the correct colored cellophane in front of it to change the light's color.

I attached the headlamp to the stage and tested it with fog to great success. The idea was that the light could pulse in time to the music - or at least the first song, *Carminia Burana, O Fortuna*.

To drive the torch, I used a motor driver. This idea originally came to me as I was preparing for RoboCup Rockhamption Regionals. I needed to drive some Christmas lights - and whilst I could have created some fancy transistor amplifier, I realized that a motor driver could do the same job.

I carried this idea over and settled on using some spare ESCON 36/2 motor drivers to drive the lights.

As for getting the lights in sync with the music, I developed code to allow me to "prerecord via buttonpress" the right timing. For example, I play the song through a first time, and press a button on the microcontroller every time I want the lights to come on. The MCU records this, and then can play

it back.

I implemented this solution after Regionals in Rockhampton where I had to manually type in the timing of the lights. To test them I had to listen to 90 seconds of music before I actually got to the part where the lights were - which was obviously time consuming. This system is much simpler, and very easy to implement.

Part VI

BaralabaBob Technical Manual

Chapter 22

Introduction

Originally this document was a separate one entitled "BaralabaBob Technical Manual", however in this edition of the manual I have combined both the logbook and the technical manual as often I make reference to and from both.

NOTE: Please understand that a majority of this documentation was written up to 6 months ago, in which time a lot of things may have changed. All of the sections have been reviewed recently (within 2 weeks), however there could be some outdated content still lurking around.

22.1 History

It was purchased to compete in RoboCup, in the dance category and as a platform for further training.

Soon after purchasing I was asked to enter a robotics exhibit into *Gumna No Sheila* in Goombungee. It had to be operational for 30 days.

22.2 Uniqueness

This robot is unique in the sense that I'm completely redesigning the control system that what is provided by Lynxmotion. Rather than the control sys-

tem using traditional textual pre-programmed movements, or IK controlled movements based upon real-time control it uses an all new system where the animator uses the open source 3D modeling and animation program, Blender, to program the robot's movements.

Chapter 23

Electronics

This chapter provides an overview of the electronic hardware located on, and related to BaralabaBob.

23.1 Control Boards

There are two main control boards on this circuit, the SCC-32 Servo Controller, and the Raspberry Pi Model B.

23.1.1 SCC-32 Servo Controller

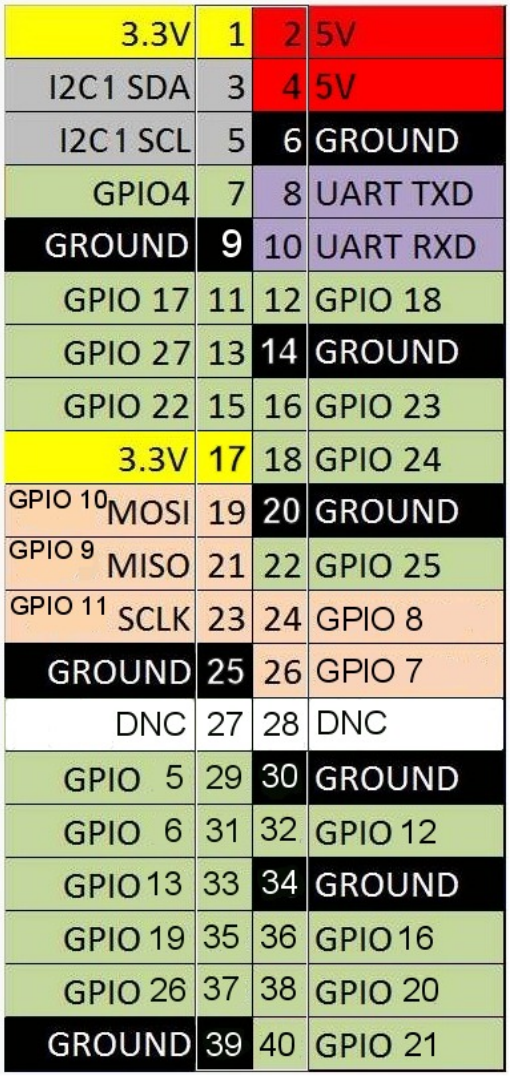
This board came with the original purchase of the APod. It can control up to 32 servos and can be interfaced with using serial.

Links:
Product Page
Documentation

23.1.2 Raspberry Pi Model B

This is the central brains of the robot. Originally BaralabaBob was powered by a BotBoarduino board, however I quickly realized that it would be a good idea to swap that out with an RPi for ease of programming, and the immense power having an embedded computer on it brings.

I currently have two RPi V2s in stock, however I haven't had a chance to fully create their OS image to use on the robot. Because the current RPi is doing the job sufficiently, I haven't needed to upgrade to its quad-core alternatives.



Storage

On BaralabaBob we are running a SanDisk Ultra (Class 10) 32GB. Previous to this I used all kinds of cheapo SD cards and had nothing but frustration with corruption and the like.

One mistake in purchasing this was, I should have bought the micro version (which costs the same). With the advent of the release of RPi v2, I'm

unable to use this SD card with it, but if I had purchased the micro version, I wouldn't have had any issues.

Previous Controllers

Previously BaralabaBob had the BotBoarduino installed, however I replaced this with the RPi Model B. This RPi was fried from some unknown reason, most likely due to power spike or short circuit. We then ordered two more RPis from Pi Australia (from what I gather, associated to Little Bird Electronics)

23.2 Power Supply

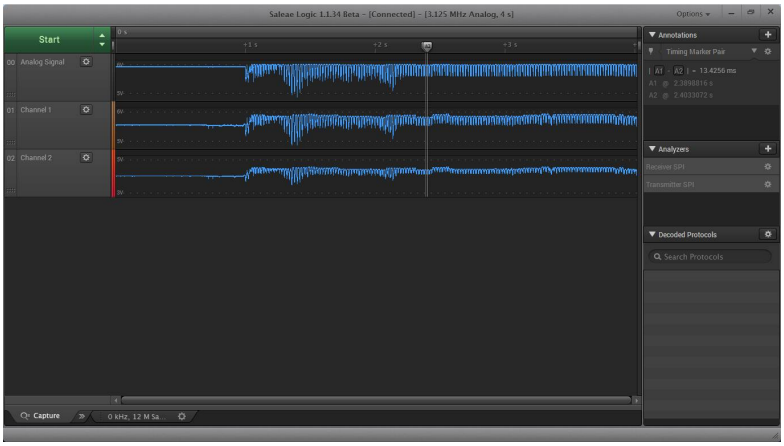
The suggested power supply was a rechargeable 6v NiMH battery. Because of the Goombungee show, and it's long term power requirements we decided to purchase a Lab Power Supply.

23.2.1 Lab Power Supply

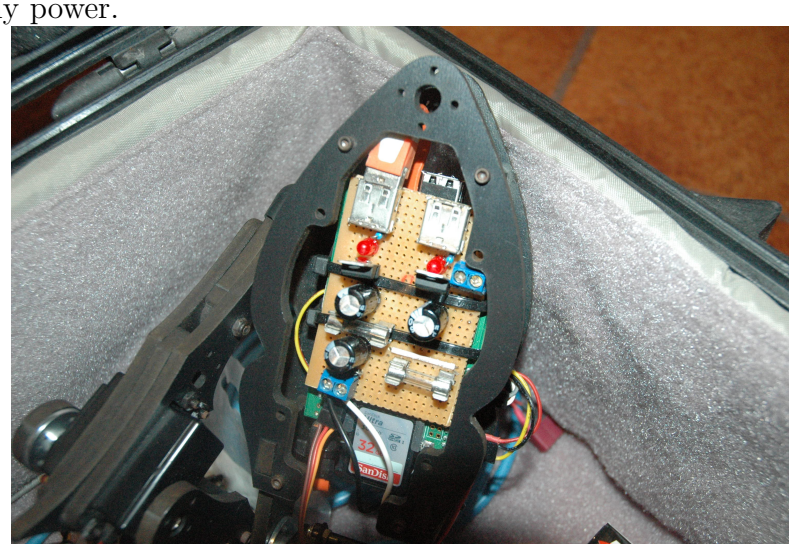
We purchased a HCS - 3402 Manson Power Supply from Altronics to supply the power requirements of the Goombungee Show.

23.2.2 Battery

There are two batteries on-board the robot. The reason behind this is, when 25 servo motors start up on the main power supply, it causes many ripples and drops the voltage way below the brownout threshold of the microcontroller.



I tried designing a power circuit (as seen on the back of the robot) to try and counter this, but it was unsuccessful, as the brownout was too severe for too long a time. The circuit consists of capacitor/diode pairs (whilst power is good the caps power up, then when power goes bad the MCU is powered by the cap, and the diode prevents backflow of current back into the motors). The board features two output USB ports. The ports obviously don't carry data, only power.

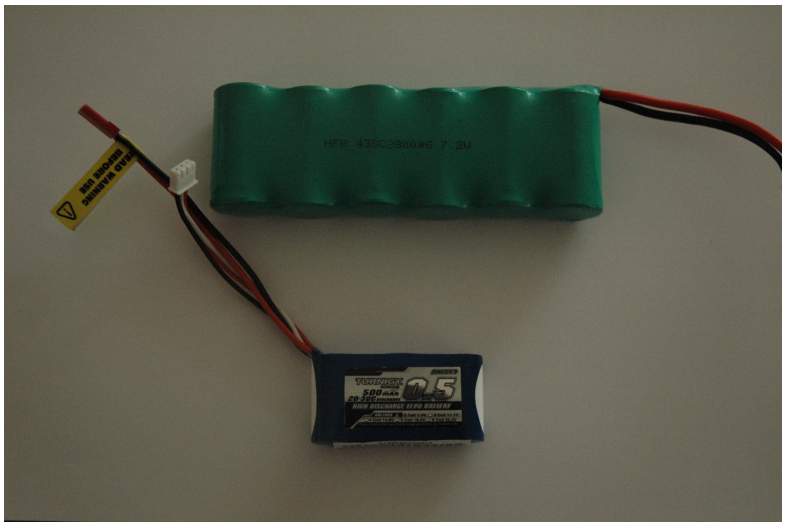


Although the board doesn't solve the problem of brownouts, it is a very functional and flexible power distribution board. It takes in power via either header (for direct connection to small batteries) or screw terminal. It then

sends this power through two identical circuits which fuse protect, regulate the voltage to 5A, give a visual indicator using a red LED, and then distribute power via USB or screw terminal.

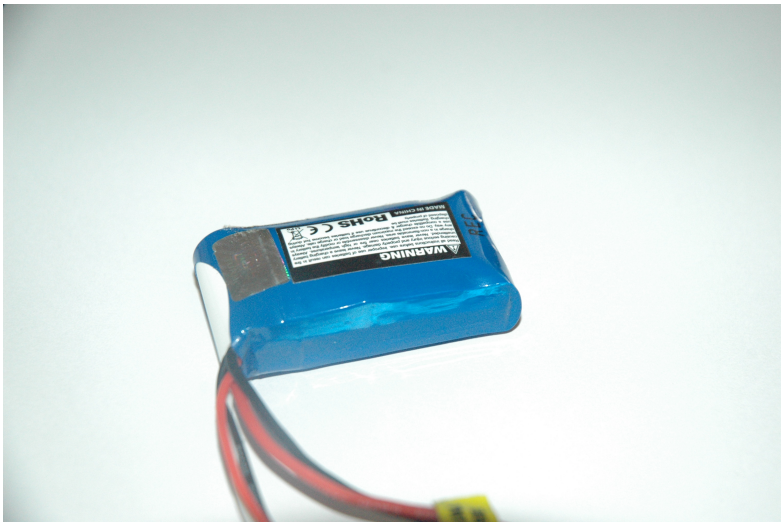
I chose USB as the power distribution connector because the RPI takes power in via a USB B-micro port. I also have the second USB power because I can use it with a y-cable to supply power to any power-hungry peripherals that I might want plugged into the RPi, such as a camera, Alfa AWUS036NHV WiFi adapter, or external hard drive.

As for the batteries, I use one 7.2V NiMH racing car battery, and one 7.4V 500mA LiPo.

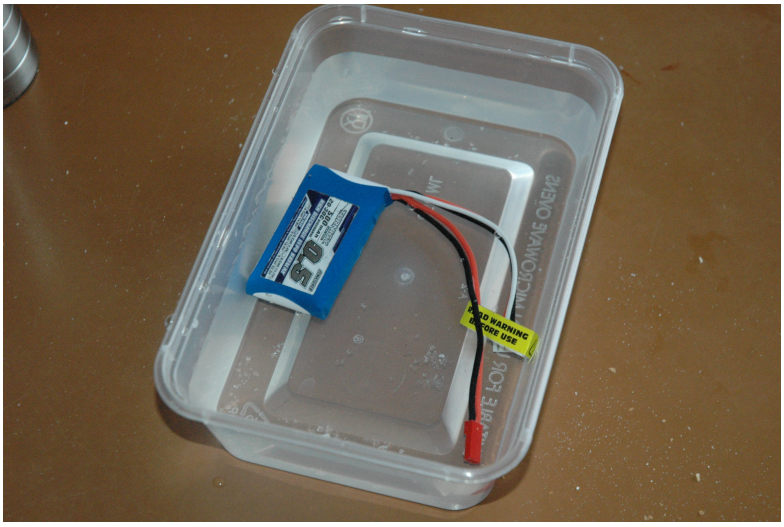


LiPo Disposal

As some of you may know, LiPo batteries have a tendency to catch fire when you mistreat them. I accidentally over discharged one of my RPi batteries, and a few weeks later after storage you can see the result, being the great amount of swelling.



To dispose of the LiPo, you soak it in a salty water solution for around 2 weeks.



Chapter 24

Software

This chapter provides information about the software located on BaralabaBob's controller, and required to run and program it.

24.1 Operating System

I'm currently running Raspbian. Raspbian is a distribution of Debian linux compiled specifically for the ARM based instruction set of the Raspberry Pi. Experimentation has been done with Arch Linux to no end.

24.1.1 Image Download

Currently I have no pre-made image ready for download. This will hopefully change in time. I could simply read the SD card and save the ISO to my website - but pratically the ISO will be 32GB. A better option is to set up some system which reads the currently used disk and saves that to an ISO on the actual RPi - but I haven't set that up yet.

This image can be restored to the SD card using a program such as Win32DiskImager. I decided to go down the path of imaging the SD card because I was frequently having to rebuild the entire OS when something corrupted (when I was using cheap SD cards). Although I am no longer using cheap SD cards, I consider it good practice to keep a good base image of the OS ready to go if anything happens.

24.1.2 Building Image

- Raspbian image should be downloaded from `http://www.raspberrypi.org/downloads/`. Torrenting is a good option for downloading for speed.
- Write the image to an *MicroSD Card* of a size no smaller than 8GB using Win32DiskImager.
- Edit cmdline.txt to remove all references to a serial console. (The reason for this is because I'm using the UART line on the RPi to communicate with the SCC-32 servo controller, and whilst booting the serial console sends invalid command signals to the SCC-32.)
- Plug in ethernet cable to the RPi and a DHCP enabled switch/router/server.
- Use Advanced Port Scanner (From Radmin) to scan the network for new IPs/IPs with the manufacturer ID of Raspberry Pi Foundation.

- Log in using SSH (use PuTTY on Windows, ssh command from terminal on *nix systems). Default username and password are pi and raspberry respectively.
- Run *"rm -rf python_games"* (This cleans up the home directory from unnecessary clutter).
- Run *"sudo raspi-config"*
- Select "expand filesystem".
- Reboot.
- Relog into SSH.
- Run *"sudo apt-get upgrade"*
- Run *"wget –user=[MY BITBUCKET USERNAME] –password=[MY BITBUCKET PASSWORD] https://goo.gl/QlJj7C"* (This downloads a script which will download the required packages.)
- Run *"sh setup_pi.sh"*. Fill in required information as required! The git clone from GitHub will fail as the repository is no longer located there. Will fix up this issue some time.
- In the future there will be instructions on how to install the RPi drivers for the new AWUS036NHV WiFi module.
- No need to git clone any repositories onto the device, as repository is installed on my computer, and uploaded using the remote deployment feature in Pycharm, which uploads all files to the RPi.

24.2 Networking

The primary mode of communication with the RPi to program it and monitor it's state is through the network interface(s). This section outlines the configuration of those interfaces.

24.2.1 Ethernet

The RPi has a static IP configured for eth0 on 192.168.2.100. Configure your computer's Ethernet interface to 192.168.2.1, and proceed to login over SSH.

24.2.2 WiFi

In the future the RPi will create an ad-hoc network, and enable DHCP, allowing people to login using specified credentials.

24.2.3 Serial

A network interface can be configured to use the serial terminal, however this has not been enabled on the BaralabaBob as the serial line is being used to control the SCC-32 Servo Controller.

24.2.4 Current Configuration

Network configuration is located in */etc/network/interfaces* in Raspbian.

```
auto lo
iface lo inet loopback

iface eth0 inet static
address 192.168.2.100
netmask 255.255.255.0
gateway 192.168.2.1

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.0.8
netmask 255.255.255.0
gateway 192.168.0.1

wpa-ssid "BigPond3BF6"
wpa-psk  "<BigPond3BF6 Password>"
```

24.3 Access

There are three options for accessing the RPi's terminal. The most common, and the one we will be using throughout this manual is SSH over a network interface.

When logging in, use the username "pi" and the specified password.

24.3.1 Terminal

Terminal access gives access to the linux terminal.

SSH

Using an SSH program (Putty on Windows and ssh in the terminal of Mac and Linux machines), you can connect to the RPi over one of its network interfaces. For reasonably fool-proof testing, Ethernet is the most reliable.

Physical

This is the most fool-proof way to access the terminal. It involves plugging in a monitor using the RCA or HDMI ports, and plugging a keyboard over USB. One gotcha is if the monitor is not properly configured. If this is the case, remove the SD card from the RPi, plug it into a computer, and edit the config for your particular monitor option.

Serial

A default Raspbian install provides by default a serial on the terminal port. This, however, has been disabled to allow communication over that serial port to the SCC-32 Servo Controller. Having the serial terminal enabled while the SCC-32 Servo Controller is powered will cause the servos to go haywire.

24.3.2 File

File access provides the ability to transfers to and from the controller.

SFTP

File access is available over SFTP (using SSH). FileZilla (for Windows) has the ability to connect to it. Make sure to use Port 22 (the SSH port) when connecting in FileZilla.

FTP

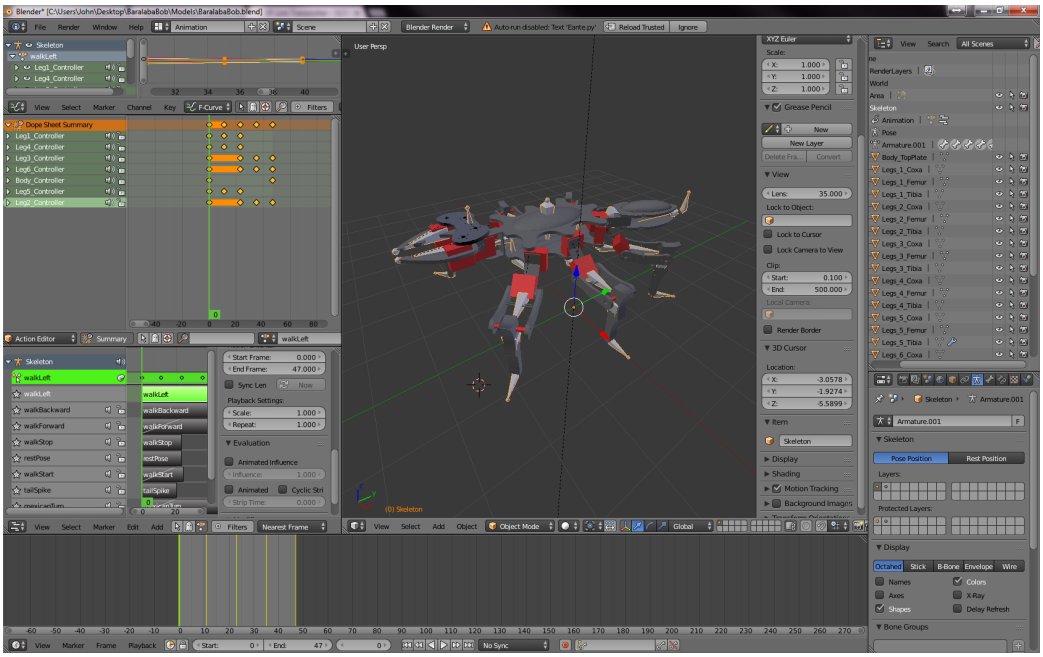
VSFTPD has been installed in the past, however has been replaced with SFTP for the sake of removing unrequired software. Consequence of this is that the pi is no longer accessible directly from Windows Explorer, and needs external software, such as FileZilla when connecting over SFTP (see above).

24.4 Software Used

This is a list of the software used in the development of this robot. All software is aimed for someone using a windows machine. All software mentioned below is available for download directly off the Pi. See Embedded Downloadable Resources for more.

24.4.1 Blender

Blender is open source 3D animation, modelling, game making, etc software. It is written in python and contains an excellent API and scripting interface allowing me to code addons allowing easy interfacing between the robot and Blender. Using Autodesk Maya was considered in place of Blender, however due to its cost, size, availability, and closed source nature, I opted for Blender. More on this later ;D



24.4.2 PuTTY

Putty is a lightweight program that I use for accessing the SSH terminal on the RPi.

24.4.3 PyCharm

PyCharm is my preferred python editor which I use for programming the python code on the robot. There is a freely available community version. Alternately you can access a student version freely for a year if you have a student email address. I've done the latter, and as such can access a full version of PyCharm until March 2016.

The main advantage of the full version is that it is able to remotely edit code using SFTP, and remotely execute code using SSH. This is really advantageous for this purpose.

24.4.4 Python 3.4.3

The python runtime is required for the running of the code. We are using Python 3.4.3 as the standard version across the board if at all possible.

24.4.5 FileZilla

Filezilla is a program for accessing a FTP/SFTP server. It is used for accessing the SFTP server on BaralabaBob.

24.5 Blender

BaralabaBob uses a custom version of Blender. The differences are listed below:

- RPyC Included
- **TODO: Script/ADDON Included**

At the time of writing, the version was 0.1.

24.5.1 Building Blender Image

BaralabaBob uses a "custom" version of blender. Custom in the sense that I've added python libraries to a .zip distribution. This version of blender is also in a portable form - no install is required. It's based off a 32-bit built of Blender 1.73a. Downloaded 24/03/2015.

- Downloaded 32-bit zip for windows from *http://blender.org/download/*
- Extract the .zip
- Download .zip source of RPyC *https://pypi.python.org/pypi/rpyc*
- Extract that .zip
- Navigate to blender_folder\2.73\python\lib\site-packages\
- Install the RPyC source folder into site-packages
- **NOTE: Install scripts coming...**
- Upload to website, under robotics folder.

24.6 Embedded Content

The pi hosts a website accessible from the local network which contains the files for all programs required.

Location: `http://<pi_network_location>/files/`

For example, `http://192.168.2.100/files`

Chapter 25

Programming

The BaralabaBob project as a whole has had significant time put into the programming of it.

There are two main divisions of the code - Frontend, and Backend.

Backend software generally is directly to do with actually moving the robot around in it's enviroment. Amelior, Shalom, and GoombungeeShowClient all would be conisidered backend projects.

Frontend is software that allows human interaction with the robot. This software includes Eante.py and Dumper.py, GoombungeeShowServer, IK Experiment, and JoystickController. Eante and Dumper are directly related to the animation cycle of the robot, whereas JoystickController is to do with real-time remote control of the robot.

25.1 Backend

25.1.1 Amelior

***Ameliorant**, verb* - to make or become better.

Amelior comprises of what used to be the BaralabaBob module in the Goombungee Exhibition days. Amelior contains the base, "object oriented" model for the robot, including event system, and blender animation playback functions. Experimental inverse kinematics mode has been added, with promising results.

Model

There are two forms of "Model" referred to in the Amelior project. One is purely mathematical, in the form dimensions and positions, outlined in configuration files. The other "model" is an object-based one, being composed of python classes.

Mathematical - This comprises of several configuration files outlining the demensions and positions of relevant mechanical systems on BaralabaBob. These configuration files were designed to be easily "swap-outable", allowing this codebase to work on several different hexapod platforms, with the corresponding configuration files.

These files can be found under *BaralabaBob/Codebase/Amelior/config/-model/BaralabaBob/*.

Objective - this comprises of several pythonical objects forming a model. Through modifying variables and calling functions this model can be manipulated. The top object is considered the "Hexapod" object, giving system-wide commands like "turnOn" and "turnOff". Hexapod has sub-objects of legs, mandibles, and tail. Legs has sub objects of 6 instances of the leg object.

At the mandibles, tail, and leg object levels there are several "joint" objects. Each joint represents a physical motor in reality. For the legs there are 3 joint objects, the coxa (hip), femur (upper-leg), and tibia (lower leg).

Scripting

In the early days of BaralabaBob I needed a way to program routines into the robot. I decided that writing python "scripts" was the best way to go.

The scripts can be found in the *BaralabaBob/Codebase/Amelior/config/-model/BaralabaBob/scripts/* directory.

The "script executor" (actually named sequence executor) is located at *BaralabaBob/Codebase/Amelior/util/scriptexecutor.py*.

The script executor code could either import a local script, or even download one over HTTP, and then execute it. The script executor was controversially named "sequenceexecutor" - which later clashed with the Blender Playback code, which was then named script-executor. This problem has yet to be rectified.

Blender Playback

As outlined in the Blender section, I developed code that would plug into blender and would export animation data, for later playback on the robot.

Both the scriptexecutor.py and action.py located in the *BaralabaBob/-Codebase/Amelior/util* folder open and can playback any seqeunces located in the *BaralabaBob/Codebase/Amelior/config/model/BaralabaBob/actions/* folder.

The average .act file includes information including the name of the sequence, the number of frames, and keyframe information.

The keyframe section of an .act file includes information for each keyframe in the animation, along with the where the joints have to be at that keyframe.

Keep in mind that the joints have to be at that position by the time that keyframe gets reached. For example, if I'm at Frame 0, and Leg1.Tibia needs to be at angle 30 by frame 72, I'll move it 10 degrees per second for 3 seconds, when the framerate is 24 fps.

When I issue a command for an action to be played back, action.py starts a while loop. Inside that while loop there is a "time checker". The "time checker" checks the time since the start of the animation, and then tells certain frames to play when their time has arrived.

Because of the way this code is structured, if the process is running slow, it may skip a frame. Although unlikely that it would skip a frame, it does so to ensure timing with the music is maintained (in the case of a RoboCup Dance performance).

Also see: Blender

Inverse Kinematics

Inverse Kinematic tells a system at what angles the joints should be for the end of an arm/leg to be at a certain position in a multi-jointed arm/leg.

In the case of this hexapod I need to be able to tell the robot to move the foot to certain position, and it figure out in real time what joints to move, and where, to achieve that. This feature has been implemented, and is working solidly.

Another feature of the Inverse Kinematics should be that I can move the base of the body, whilst the legs stay in place. Because this is modifying the positions of the "leg chain" in the opposite direction to normal, it's a little more difficult to implement. This code works, but is in experimental status.

A fully fledged IK system would allow me to tell a robot to move (for example) to 20cm, 20cm (X, Y), and it would figure out all the steps it needs to take to get there, and then executes that. This again is in experimental status.

It can be found under */BaralabaBob/Codebase/Amelior/* in the repository.

25.1.2 Shalom

***Shalom**, interjection* - Peace, to be at peace. To be harmonious.

Shalom is my first attempt at combining Inverse Kinematics with a hexapod model. I envision a much smoother and harmonious coexistance between the code and the machine with repository as it's calculating the moves on the fly, rather than following some preprogrammed routine.

Development of this project has somewhat died due to the time it would take to rewrite a fairly decent system, that is the Amelior project. This project was meant to contain the Inverse Kinematics code, however I ended up implementing a trial version of it in the Amelior code.

It can be found under */BaralabaBob/Codebase/Shalom/* in the repository.

25.1.3 BaralabaBob

BaralabaBob was the original "Operating System" code for the Hexapod. It contained a "object-oriented" model of the hexapod. By manipulating the objects in the model, it would send an event call through the system which would in turn send data to the servo controllers.

This code was designed to be specific to BaralabaBob, and could not be used on other platforms. It was also designed to specifically link in with the GoombungeeShowServer/Client software, allowing the robot to be controlled by a smartphone as part of an exhibit.

BaralabaBob was superceeded by Amelior. According to the online repository this merge/shift happened on the 12th of March, 2015. The commit code is 441d578. Around this time the repository was shifted from GitHub to BitBucket. The repository titled BaralabaBob originally housed only the Operating System code, the code was renamed to Amelior, and other projects were also put in the repository.

25.1.4 GoombungeShowServer

This code was written to connect the Javascript frontend (as seen in GoombungeeShowClient) with the python backend of Amelior (or what was BaralabaBob)

It can be found under */BaralabaBob/Codebase/GoombungeeShowServer/* in the repository.

25.1.5 GoombungeeShowClient

This code was written to provide a way for people to control the robot using their phone by connecting to a website (contained in this code).

NOTE: My IDE of choice for editing the JavaScript was KomodoEdit

It can be found under */BaralabaBob/Codebase/GoombungeeShowClient/* in the repository.

25.1.6 IKExperiment

This code I wrote when I was first playing around with the IK algorithms. It features a pygame frontend to allow you to see visually the results in real time.

It can be found under */BaralabaBob/Codebase/IKExperiment/* in the repository.

25.1.7 JoystickController

This code is used to control the robot using a joystick.

It can be found under */BaralabaBob/Codebase/JoystickController/* in the repository.

25.2 Blender

Blender is the primary method of programming the robot. Michael Board created a model of the Lynxmotion A-Pod in TurboCad 11, and exported it as 3DS format. It was then imported into Blender and rigged by John Board.

In the early days of BaralabaBob, I realized assumed that I didn't have the nouse to write the inverse kinematics code, and I also knew that creating any kind of good hexapod routine would be difficult with textual commands - so I seeked an alternative.

Several years previously I had developed a theoredical idea for a software which would allow you to animate a robot in 3D animation software, such as Blender or Autodesk Maya, export that animation data, and then play it back on the robot.

I whittled my options down to Blender or Maya because of both of their abilities to have scripts written in python for them, making it easily compatible with the rest of the system. (Also my affinity with python may have biased my decision slightly...)

25.2.1 Location

Under the git repository, the models loc Thd: */root/Models/Boilerplate.* The Boilerplate template is a file which does not change, and can be considered a solid foundation for future experimentation and routines.

25.2.2 Blender Design Standards

The leg joints all conform to three basic joints:

- When Coxa joints move towards the front of the robot, their angles become more positive.
- When Femur joints move upwards, their angles become more positive.
- When Tibia joints move outwards, their angles become more positive.

25.3 Amelior

25.3.1 Model Configuration

All model configuration (dimensions, layouts, etc) for BaralabaBob lies within lies within *Amelior/config/model/BaralabaBob* directory.

Because Amelior is designed to be a generic hexapod driver, in the future there will be support for directories with other hexapod configurations in them other than *model/BaralabaBob*.

hexapod.cfg

This specifies generic properties of the hexapod, such as servo offsets, and the configuration files for other limbs.

25.4 Git

I maintain a private repository on git.johnrobboard.com (hosted on BitBucket.org). I chose BitBucket over GitHub because of it's free private repositories. The BaralabaBob repository is private due to the potentially copyright breaking nature of some of the content (namely the 3D model of the A-Pod).

25.4.1 Git Clone Instructions

Navigate to the directory in which you want to download the repository and follow the instructions over leaf for either SSH or HTTPS...

HTTPS Clone

This is an easier method, and requires no previous setup, but requires a username/password authentication each clone/push.

Listing 25.1: HTTPS Clone

```
git clone https://boar401s2@bitbucket.org/boar401s2/baralababob.git
```

SSH Clone

Follow the instructions located *here* to authenticate your key with BitBucket.
After that clone in the Git Bash using:

Listing 25.2: SSH Clone

```
git clone git@bitbucket.org:boar401s2/baralababob.git
```

Chapter 26

Mechanical

This chapter outlines mechanical information about the Lynxmotion A-Pod. This chapter is most likely the least filled in. It needs a lot of work.

26.1 Dimensions

When standing, the top of the top body plate is 185mm from the ground on which it stands.

Each femur joint should be 120mm from the ground.

The femur joint is 50mm away from the coxa on the local X axis.

All this joint information was used to generate the 3D mathematical model used in the inverse kinematics.

26.2 Kit Information

We ordered the Lynxmotion A-Pod as a kit with the mechanical components and servos.

`http://www.lynxmotion.com/c-154-a-pod.aspx`

In the same order we also purchased the required batteries, PS2 controller, BotBoarduino, and SCC-32 Servo Controller.

The servos used are HiTec HS-645.

26.3 Maintenance

Not a lot of maintenance has to be done, however the screws and nuts on the servo brackets have a tenancy to come undone. I have looked at a few solutions such as spring washers, loctite, etc - however for now I'm simply regularly making sure that the screws are tight.

26.4 Motor Burnout

I have learned through bad experience that if the servo motor is told to move to a position that is physically obstructed, and left there for a while it will burn out.

The telltale signs of burnout is typically that the motor is warm or hot, potentially a bad small, and the motor seems stiffer than usual when turned off, and the joint moved by hand.

Prevention is better than cure - make sure that a servo is not strained for too long. If this does occur, the only fix is to replace the motor. The motor brand is:

Hitech MS-645MG

26.5 Screw Sizes

This section is incomplete. More information is required on the screw and nut sizes of the robot.

Part VII

Appendix

Chapter 27

Test & Tag Inventory

All equipment used at RoboCup Nationals 2015 has been tested and tagged according to regulations. Here is an inventory of tags that you can find associated with any of our electrical equipment.

ELECTRICAL EQUIPMENT TEST REPORT – TEST RESULTS

Customer: JOHN BOARD **Ph:** 07 3381 0522 Email: johnrobboard@gmail.com
Address: 15 ACACIA PLACE CAMIRA QLD **Date:** 13.08.15
Tester: Verl Rusovan (Rusty) **O/N:** 217
Licence No: 77689 **Tested in Accordance to AS/NZS 3760:2010**
Page 1

Express Test and Tag

	ITEM	ID #	TIME	Next Test Date	VISUAL TEST	EARTH TEST (Ω)	INSULATION TEST (MΩ)	TOUCH LEAKAGE (mA)	POLARITY RESULT	DIFF LEAKAGE (mA)	PRCD2 (ms)	PRCD2 (ms)
1	LAPTOP IEC LEAD	1	12	13-8-2016	PASS	.20	>200		PASS			
2	LAPTOP ADAPTOR	2	12	13-8-2016	PASS	D.I	>200					
3	SOLDERING STATION IEC LEAD	3	12	13-8-2016	PASS	.05	>200		PASS			
4	SOLDERING STATION	4	12	13-8-2016	PASS	.43	>200					
5	FOG MACHINE IEC LEAD	5	12	13-8-2016	PASS	.09	>200		PASS			
6	FOG MACHINE	6	12	13-8-2016	PASS	.63	>200					
7	PLUG BOSS SURGEBOARD	7	12	13-8-2016	PASS	.19	~~>200~~ 18040		PASS			
8	LAB POWER IEC LEAD	8	12	13-8-2016	PASS	.06	>200		PASS			
9	LAB POWER SUPPLY	9	12	13-8-2016	PASS	.08	>200					
10	HEAT GUN	10	12	13-8-2016	PASS	D.I	>200					
11	A/C ADAPTOR	11	12	13-8-2016	PASS	D.I	>200	.06				
12	MONITOR IEC LEAD	12	12	13-8-2016	PASS	.07	>200		PASS			
13	MONITOR	13	12	13-8-2016	PASS	.13	>200					
14		14			PASS		>200					
15		15			PASS		>200					
16		16			PASS		>200					
17		17			PASS		>200					
18		18			PASS		>200					
19		19			PASS		>200					
20		20			PASS		>200					
21		21			PASS		>200					
22		22			PASS		>200					
23		23			PASS		>200					
24		24			PASS		>200					
25		25			PASS		>200					
26		26			PASS		>200					
27		27			PASS		>200					
28		28			PASS		>200					
29		29			PASS		>200					
30		30			PASS		>200					
COMMENTS:												

Chapter 28

Bill of Materials

If I have time, I will fill out this section with the BOM details that I still have access to.

Part VIII

Index

Index