Extended Project Diary

Template

- What you did
  - Ideas and thoughts
  - Successes and problems
    - What you are gaining from overcoming problems
- Progress – what's next?
- Any changes to project topic, title or timing

6/12/2016

Today I met with George Tuli[1], who built a 6-wheeled rover for his EP. I was impressed by the professional nature of his documentation and by the pictures of the final version of the rover. Most of the specifications for his project were set by him from the start, with minor changes along the way. I should try to set detailed specifications so I can show how I managed (or didn't manage) to achieve them.

Seeing George's project suggested to me that I was on the right track as it is similar to my plans in its overall structure and process.

I should probably set a budget for my project. I might base this on George's budget as the components of the robot will be fairly similar. I should be reasonably generous too; George overran his budget several times.

8/12/2016

In the EP session today we discussed primary research. I was initially doubtful that I could do any, but I discovered that my meeting with George Tuli[1] counts as I was directly obtaining information. I could potentially do similar interview-style research in the future, alongside secondary research. Testing prototypes and/or alternate configurations of a robot also counts as primary research.

Separately, I decided that I need to make an initial digital model of how I'd like my robot to look. As I have no experience in mechanical CAD (computer aided design) software, I will research which one best fits my needs. As I am a student I qualify for various educational discounts and certain software is free! I will probably get something that is free as professional software can be prohibitively expensive.

5/1/2017

In today's EP session we evaluated projects by previous students. I was quite surprised that a project that at first seemed competent received a grade D. This made me rethink how I would show my development of skills and time management, and I have resolved to emulate the most successful project we looked at by recording more details on these in my project diary. I also noticed that the timeline for the best project was significantly more detailed than mine, so I will redo mine with more tasks.

Next I will continue looking into CAD software for my project. I am split between using simple, user friendly software like SketchUp and complex software like AutoCAD. I should probably use industry standard software (AutoCAD) as this will help me in engineering/ physics at university. It is also more difficult so it will be easier to show progression in my skills.

6/1/2017

Yesterday I posted on the EEVBlog online forum[2] asking which CAD package was recommended. This forum has electrical engineers and hobbyists who know lots about this software. Autodesk Fusion 360 was the most recommended. It looks promising: professional but easy to use, and free for 3 years since I am a student.

19/1/2017

Today I wrote a design specification for my robot, so that I will have clear goals to work towards. I also finished off the initial timeline. Next I will learn to use Autodesk Fusion 360 so that I can make a 3D model of my design.

25/1/2017

Today I began to go through a tutorial[6] on YouTube for Autodesk Fusion 360. It's difficult to use but I'll continue to work towards my 3D model.

26/1/2017

Today I added my current progress to the initial timeline. I might have to do another (more realistic) timeline at some point, I will review the situation on March 1st, by which point I should have a prototype and be testing it.

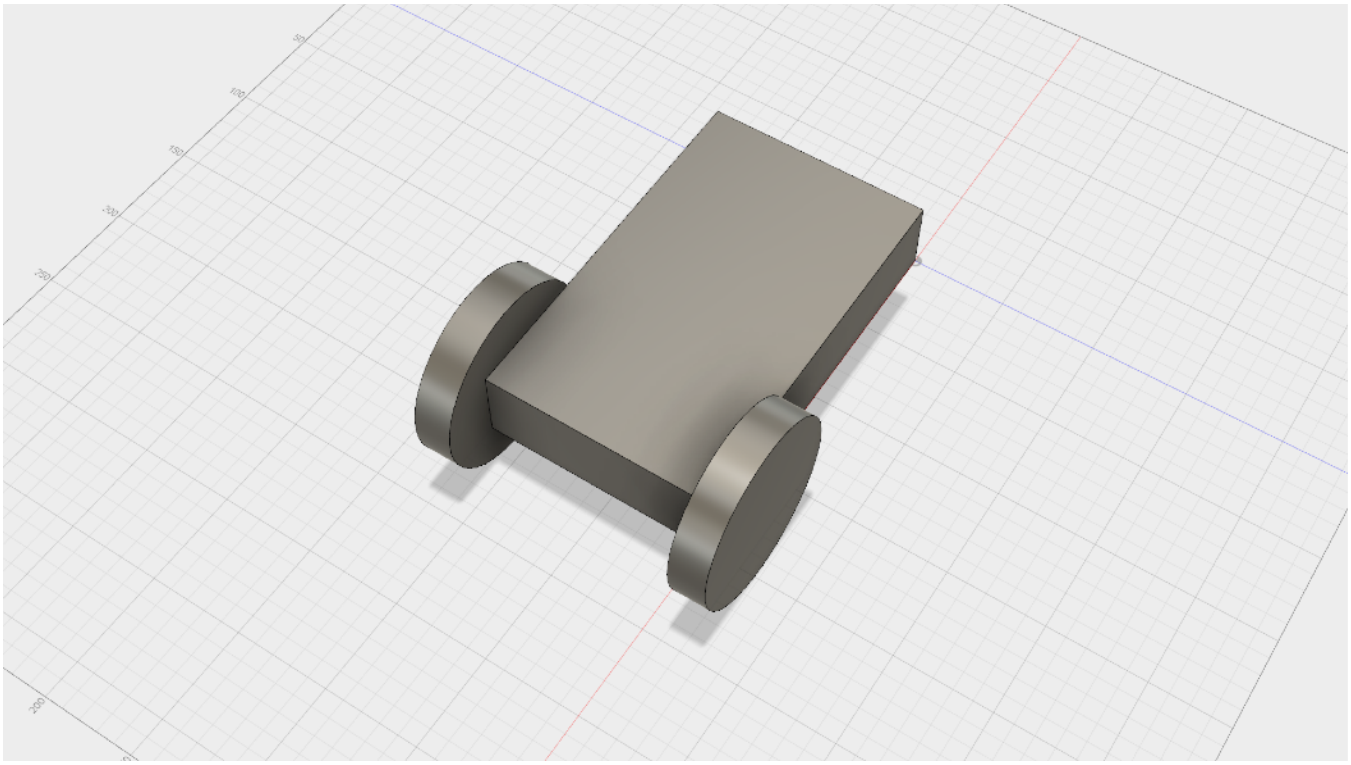
23/2/2017

I completed the initial 3D design for my robot today. It doesn't look as sophisticated as I'd like but I decided to go for a basic design as the 3D software is unintuitive. I used source 7 to help me as I found it to be much easier to follow than some other tutorials, which were aimed at people who were already experts in the software.

Next I need to make a parts list so that I can build a prototype. I have decided to try to make the prototype have all the same basic functions of the final product, so it will need to be self balancing. The differences between the prototype and the final product will be in aesthetics and reliability, rather than in functionality. This means that my prototype might take longer than I expected. If this is the case, I will adjust for it in a future timeline.

25/2/2017

I decided today that I want my robot to be open source so that others can construct a similar one easily. I will publish the designs and files under the Creative Commons CC0 (public domain) license. Next I need to continue researching which parts to get for my prototype.



2/3/2017

Today I researched which control boards I could use for my robot. These are in the 'Parts comparison' spreadsheet. I found it easy to find information but I didn't write down all my sources so I need to go back and add sources for all my points. I also need to do research on other parts that I can use.

I completed my mid project review today. I have made progress but not as quickly as I had hoped; I need to make a new timeline to reflect this. The new timeline includes additional time for research and building of my prototype.

9/3/2017

Today I continued to collect sources to determine which control board to use. When reviewing my source list I noticed that I had forgotten about the control board used in source 9, so I added it to my parts comparison spreadsheet. I also decided to use the Arduino Pro Mini clone as my microcontroller. Next, I need to decide which motors I want to use.

10/3/2017

I added a copy of the 3D model to my Design Specification today, as part of the reason that I made the model in the first place was to do so.

16/3/2017 – Employability

Interpersonal Skills

I interviewed George Tuli about his EP. This was a new experience for me and definitely improved my interview skills.

Initiative and Self Motivation
I have been consistently working on my EP every week. As I have not had an EP lesson for 3 weeks, this shows that I am capable of working independently and staying motivated.

Organisation
I have clearly organised my EP documents, and have been keeping to a regular schedule of working on my EP during my study periods on Thursday and Friday. This has developed my organisational skills; at the beginning of my time at Hills Road I did not do any work during study periods but now they are the time when I am most productive.

Problem Solving
I did not know how to use the 3D design software, so I followed several YouTube tutorial videos. Since I still found using the software difficult, I reduced the complexity of my model to allow me to get on with the rest of my project. This shows that I can solve problems through compromise as well as finding a better solution.

Working Under Pressure and Deadlines
I have stuck to my deadlines as as far as possible, but creating a 3D model took longer than expected, and I underestimated how long it would take to research which parts to use as I didn't realise how many decisions I would have to make. This has led me to make a new timeline that I will work with.

Ability to Learn and Adapt
My EP has definitely forced me to be more focussed during my free time, especially in study periods. There are many distractions that can reduce my concentration at home, but in school there are fewer. Combined with the fact that I need to spend more time than is timetabled per week on my EP, this has led me to rethink my attitude towards free time in school: I would now rather get work done than waste time during school.

23/3/2017
I added additional sources to my parts comparison list this week. Though it is not difficult, I am finding that I underestimated how many parts and decisions need to be made, so I am running a little behind schedule. To reflect this, I have re-done my timetable, and in the process I discovered I had put the Easter holidays forwards by a month. I have now corrected this.

30/3/2017
I decided that I would use the same motors as source 9's, as I know that they work for their intended purpose. After searching the websites of several distributors[29,30], I found the motor I was looking for in source 28.

I will also use the same wheels as source 9's, both because I know that they work, and because soft wheels enhance balancing performance[11] (and these are made from soft rubber). Though these wheels are expensive, they seem to be somewhat unique as I had trouble finding similar ones from source 27. I eventually found them on eBay[31].

Finding the parts I need has been quite difficult, but doing so has improved my ability to use information from one source to find similar information in another. Next I need to investigate which motor controller, battery and remote controller to use.

6/4/2017

Today I filled in some missing details on my source tracker spreadsheet. I also decided that I would aim to spend no more than £250 on this project, and added this to the Design Specification. I need to continue researching which parts to buy.

20/4/2017

I need to do the following to finish my project:

- Research remaining parts (motor controller, RC receiver, gyroscope/accelerometer) and buy them (~2 weeks including delivery time)
- Build prototype (~2 weeks)
- Program prototype (~2 weeks)
- Decide on changes for final version (1 week)
- Design chassis for final version (2 weeks)
- Build / buy parts for final version (1 week)
- Build final version (1 week)
- Program final version (1 week)

This should give me enough time to finish my project in time for the early deadline (13th July).

I bought the motors today from TME.eu. Initially I found the motors on RobotShop, where they would have been cheaper (£50.48 vs £60.41 excluding shipping), but when I went to checkout there was only one motor in stock (see Screenshot 35.png); I needed 2. I searched for 'Pololu 34:1 metal gearmotor' and found that there was a documentation for the motor I needed on TME.eu (see Screenshot 36.png). I then searched TME.eu and found the listing for the motor I needed, so I bought two of them.

Having found the motors on TME.eu, I then thought to check for the wheels as well, since they are also distributed by Pololu. I found them for £15.89 for a pair, which is much cheaper than the £42.30 that they would have costed from eBay. I then bought them, and discovered that since I had already finished my previous order I would have to pay for shipping (£5.80+VAT) again! This was irritating but I expect I will still remain below my price target of £250 max.

4/5/2017

My exams are quickly approaching, and I no longer think it is practical to expect to have finished by the early deadline. I have not made much progress since my last entry as revision has been taking priority, and I don't see this changing for at least 4 weeks (most of my exams are concentrated in the first 2 weeks of study leave).

I have been researching which motor controller to buy. I need to buy a motor driver board in order to make the motors move using the relatively weak outputs of the Arduino Pro Mini clone. From my search I discovered that boards based on the L298 chip are by far the most

common, and after researching some alternatives I decided to use a board based on the L298. I then bought an L298 board from Amazon (see Invoices folder).

5/5/2017

Today I bought an RC receiver. As I already have a Turnigy 9X transmitter, this was an easy choice for me; I bought the Turnigy 9X receiver. This cost £9.22 (see Invoices folder)

The final electronic part that I need to buy before building my prototype is an inertial measurement unit (IMU). This contains an accelerometer and a gyroscope (used for balancing the robot).

22/6/2017

Exams are finally over (Mechanics was last – 20/6/2017) so I can focus more on EP. Today we did a peer review of our projects. I reviewed Lauren's; there isn't very much in common between our projects but I realised the greater need for evidence of skill improvement. I will make sure to take lots of photos of my construction of the robot, and give lots of examples of how I troubleshooted coding problems, including work in progress versions of the code and videos of problems if applicable.

Jay looked at my project and wrote that my source tracking and evaluation was 'detailed, specific and skillfully [executed]' (pictures in Images folder). Dr McNaughton pointed out that these images are not a source, which made me think that I need to check through my source list and make sure that I haven't got other entries that aren't real sources.

22/7/2017

Yesterday I visited Cambridge Mechatronics to talk to engineers about university courses. I plan to study electronic engineering at university, so I asked my mum to get in touch with her friend who works in HR there. Though I was not expecting this visit to be relevant to my EP, I ended up talking to David Richards (Technical Director, https://www.cambridgemechatronics.com/team) over lunch.

He asked me what I was doing for my EP and I replied that I was making an inverted pendulum robot. When I told him that I was using an IMU (Inertial Measurement Unit – the module containing a gyroscope and accelerometer), he suggested that it would be best to place it at the top of the inverted pendulum as this would give it the most gain; a rotation of 1° translates to a larger movement at the top of the body than at the bottom.

David also suggested that rather than aiming for the robot to be completely still, I should have it oscillating forwards and backwards at around 5Hz. This is better as it allows me to see if the robot is unbalanced and moving faster in one direction than another.

Finally, he pointed out that the accelerometer and gyroscope combination take careful calibration and filtering, and that the accelerometer will inevitably detect a large gravitational force of g (=9.81ms^{-2}).

I am aware that I said I would work on my extended project about a month ago, and that little has been recorded here. I have been re-organising my workbench so that I have space

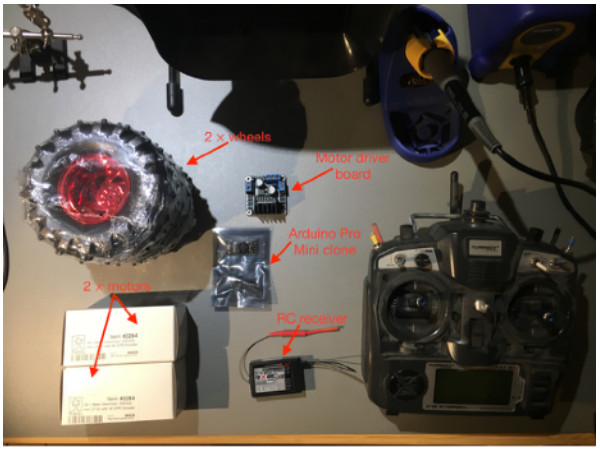
to work on my EP. I have 3 weeks of work experience starting on Monday 24th July, which is 10am–5pm, so I won't be at home until around 6pm in the evenings. I plan to work for about an hour per evening in assembling and testing my robot, and will record my progress here (taking lots of photos along the way!).

Presently my plan is to connect the parts together one-by-one, testing to make sure they work and that I can interface with them with the microcontroller. Once I have ensured they work, I plan to purchase some MDF board and attach the parts to it as a prototype. MDF is ideal as it is cheap and easy to cut, so I can screw or glue parts on to it. Once this is complete, I will look at the pros and cons of the various ways of constructing the final casing. My preliminary thoughts are below. These are from my own knowledge so they do not have sources.

Name	Pros	Cons
3D printing	• Fairly cheap and can order online • Robust • Multiple colours • Can design complex shapes (e.g., include mounting brackets for parts)	• Very difficult to design for! (I discovered this earlier when I attempted to make a 3D model of my robot) • Very expensive / difficult to get access to a 3D printer
Laser-cut acrylic	• Very cheap and can order online • Easy to design (as can only make flat shapes)	• Limited possibilities; can only cut flat shapes • Potentially less aesthetically pleasing
Wood	• Very cheap • Easy to obtain • Easy to shape/work with	• Heavy (balancing more difficult) • Less aesthetically pleasing

28/7/2017

Today I began working on the robot hardware. First I laid out all the parts I was going to use and labelled them (1 annotated.jpg). Some parts that made it onto the final robot are not included because I hadn't bought them yet.

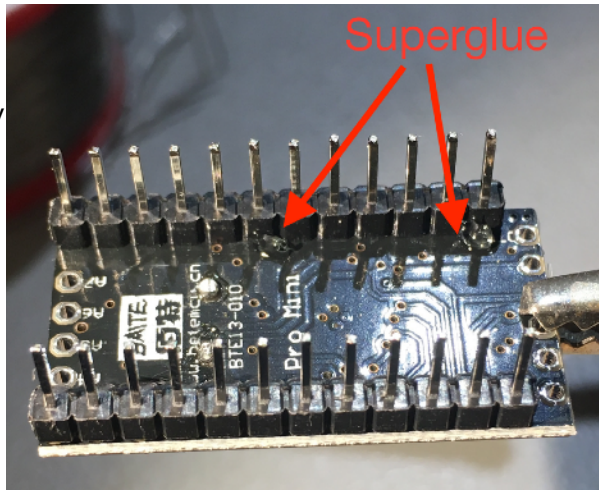


1 annotated.jpg

I attached the wheels to the motors using the included kit. I then checked if the motors worked. I unpacked them and checked the connections[24], and connected them to my power supply. See Evidence/15.m4v for a video of this.

I opened the Arduino Pro Mini clone (which is also

called a Baite Pro Mini) and thought about how to solder it. The issue was that the solder joints should be on the bottom of the board, so the header pins would fall out if I didn't attach them somehow. Initially I used superglue (7 annotated.jpg). This did not work very well as the superglue was slow to dry. When I turned the board over to solder it, the pins fell out (Evidence/8 annotated.jpg). I decided to try again later.



7 annotated.jpg

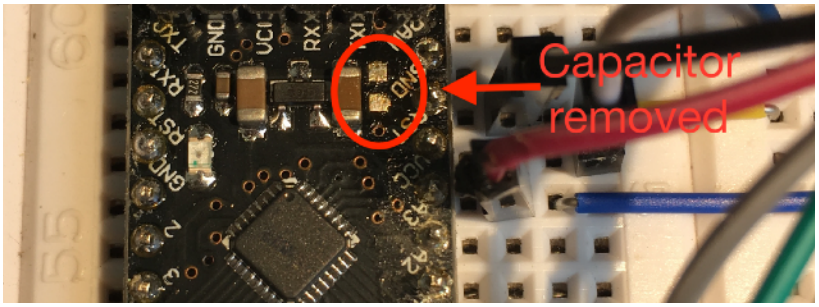
I watched a YouTube video from EEVBlog[86] that gave me a much better way to solder the header pins called tacking. This involves doing one solder joint while the board is upside down, and that holds the header pins in when the board is turned around. Evidence/9 annotated.jpg shows a tacked pin in the top right corner. See Evidence/10.jpg and Evidence/16.jpg for pictures of the soldered board.

Now that I had the board soldered, I was ready to program it. I attached my Atmel ICE to my laptop and opened a Windows 10 Virtual Machine (I had to do this as my computer runs macOS and the programming software was Windows only). When I did this, the software told me that a firmware update was available and wouldn't let me use the tool (Screenshot 2.png). I tried to complete the firmware update but it didn't work (Screenshot 3.png). I thought this was probably because I was working in a virtual machine (which is less stable than a normal PC). To solve this, I updated the firmware on my mum's Windows laptop (I don't have a photo of this).

My work today improved my soldering skills and my knowledge of virtual machines and their flaws.

30/7/2017

Having updated the firmware, the next objective was to program the Baite Pro Mini. I connected my programmer to the Baite Pro Mini using sources 53 and 54, and attempted to program a simple LED blink program (from source 55), but I got an error when I tried to upload the program (Evidence/Screenshot 4.png).



20 annotated.jpg (cropped)

I tried programming an ATtiny85 that I had lying around, and it worked. After some research I found source 52, where the first answer states 'the capacitor on the reset pin is not necessary'. I therefore removed the reset line capacitor on the Baite Pro Mini (Evidence/20 annotated.jpg). Unfortunately this did not work and I got the same error as I did with the

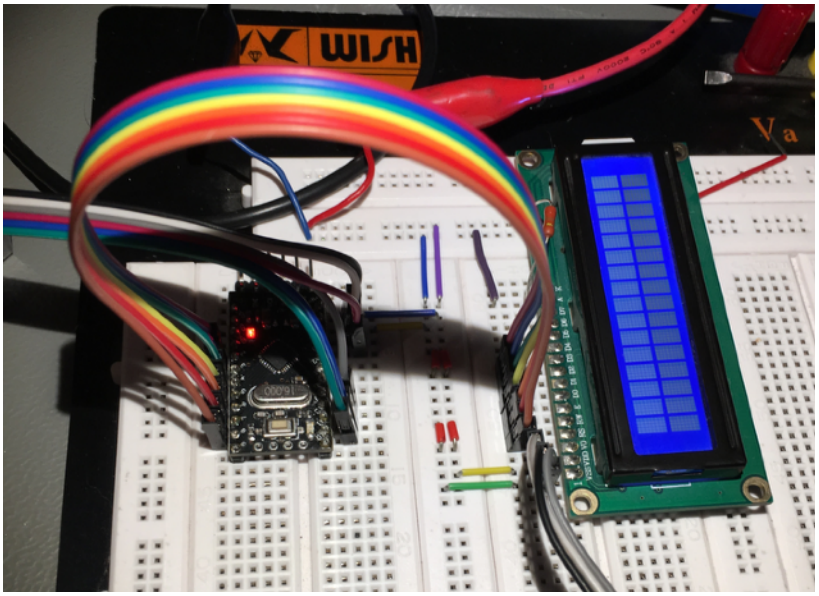
capacitor still attached. Finally I tried switching the MOSI and MISO pins the other way around to source 52. When I did this, the programmer worked.

I was now able to program the Baite Pro Mini. I changed the time between toggling the LED on/off to 500 milliseconds, and uploaded the program. This can be seen in Evidence/22.m4v .

The unexpected problems I faced today made me realise that my project won't always go as smoothly as I would like, but that solutions are usually possible. I also improved my technical knowledge of microcontroller programmer behaviour.

31/7/2017

I googled the part number of my LCD module and found source 56. I then connected the module to my Baite Pro Mini using sources 56 and 57 for reference.



23.jpg (cropped)

I realised I didn't know the clock speed of my Baite Pro Mini's processor, so I looked up how to set it. I recalled that you could set F_CPU, and thought this might solve my problem. However, I then found source 58 (a forum thread), and the first post told me that setting F_CPU doesn't set the speed directly, but rather tells the C++ compiler what speed you set via the microcontroller fuses (the two parts can't communicate each other directly). I then looked at source 59 to determine which bits needed to be set, before realising that the clock speed could be set in Atmel Studio more easily.

1/8/2017

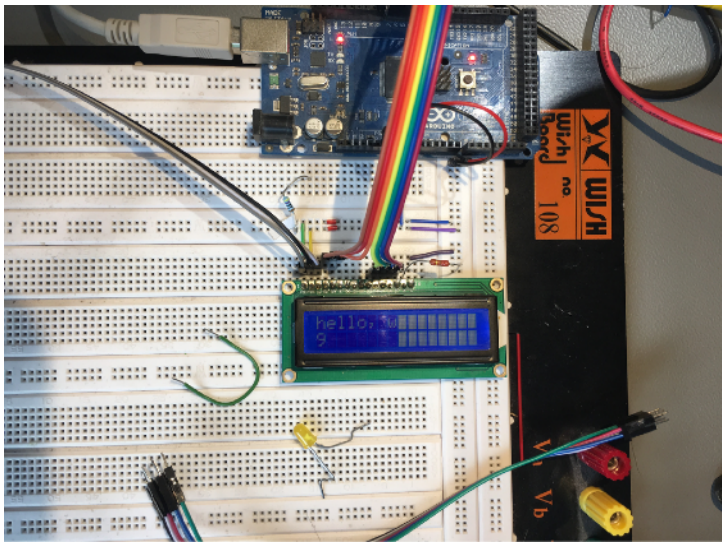
I tried to try to print text to the LCD module. I used source 57 for the code. First I realised that I connected pins D0-3 instead of D4-7, but correcting this did not solve the problem. I then tried importing the library with quote marks ("name") rather than angle brackets (<name>), but this didn't work either.

Eventually I decided that using Atmel Studio with the Baite Pro Mini would be too difficult. I knew that Arduino software made this sort of task much easier, so I decided to switch to an Arduino Mega (I already had one).

I improved my problem solving and my awareness of when to try a different approach (e.g., if the initial one is too complex).

5/8/2017

I installed the Arduino software inside my virtual machine and got the LCD working quickly using source 87. I found that the LCD was not working on half of the display, so I replaced it (I happened to have another module) and it worked. You can see the partially-working LCD on the right.



27.jpg
Partially working LCD

I then soldered header pins onto my IMU (inertial measurement unit) board, which had an MPU9250 chip on it. Before powering it up I checked source 60, which said it was a 3.3V device. I therefore ordered some level shifters from Amazon to allow my 5V Arduino to communicate with the IMU without damaging it.

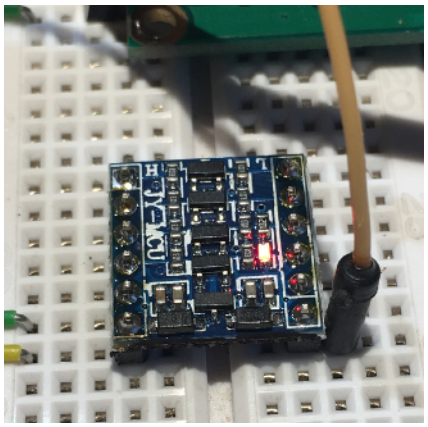
6/8/2017

I made a small board to interface the motors with the Arduino. This can be seen in images 29–39. I first planned where I would place each component, and then cut the board and drilled holes for the large screw terminals.

I decided to try to implement code to control the motors while I was waiting for the level converters to arrive. I used the sketch in source 61 to do this. I was quickly successful.

7/8/2017

The level converter arrived and I soldered on its header pins. This can be seen in images 40–42.



42.jpg
Level converter in-circuit

The L298 board stopped working during testing, and I thought I had damaged it by applying 12V to one of the inputs. I ordered another board.

8/8/2017

Upon further testing, I realised that my original L298 board was working fine. I think it was a loose connection that caused the input to stop working.

I also realised that there was no need to continue working in a Windows virtual machine, as the Arduino software is available on most platforms. I therefore copied the code over to macOS.

9/8/2017

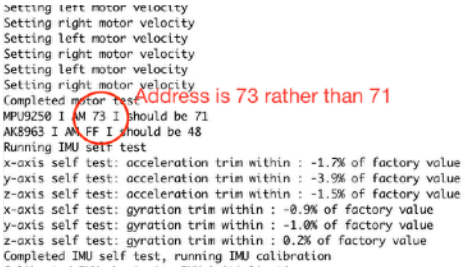
I used source 63 to find the pin configuration of my Arduino Mega, and attached the IMU's SDA and SCL pins.

12/8/2017 – Timing

I finished my 3 weeks of work experience on Friday (11/8/2017). This means that I now have a week of time at home to focus on my EP. After this, I have a family holiday, then another week of time at home. I need to finish my EP in these two weeks of time. I think this is doable, however I will not be able to design an aesthetically pleasing case/exterior for my robot, but rather will have to make a 'prototype'-like design.

I have put my robot's code on GitHub. The link is at https://github.com/microbug/ep-robot. This will allow me to track changes to my code very accurately. I learned to use GitHub (and git, the software behind it) on my work experience. This should be very useful for providing evidence of the code that I write.

Using code from source 64 I managed to get the MPU9250 working with the Arduino (screenshots 9 and 10). I was getting valid data but the ID of the MPU9250 kept being reported as 73 rather than 71. I was confused by this and checked the connections and the code, but nothing appeared wrong. I googled the problem and found source 65, which suggested that an ID of 73 meant that the chip was actually an MPU9255. This chip is almost identical to the MPU9250 and all my code seemed to work with it, so it shouldn't be a problem.



Screenshot 10.png (cropped)
Arduino successfully reading data from the MPU9250 board

13/8/2017

I couldn't get the AK8963, the magnetometer in the MPU9255, to work. It should have been working by default as Kris Winer's code[68] included setup procedures for it, but I kept getting an error. This can be seen in Screenshot 10.png, where the AK8963 address is FF when it should be 48. This indicates a hardware error as FF is the maximum possible value; it isn't like with the MPU9250/5 where the value is related to the model.

I tried adding pull up resistors for the I2C bus using source 88 but this did not fix the problem. I also wondered if the MPU9255 does not include the AK8963. I decided not to pursue the issue since I don't need to use the magnetometer anyway. Reading source 88 improved my knowledge of I2C.

I found source 67, which backs up what I found yesterday; it suggests that my 'MPU9250' board has an MPU9255 on it.

15/8/2017

As I had the IMU working and giving accelerometer and gyroscope readings, I was ready to implement some sort of filtering to get an angle reading.

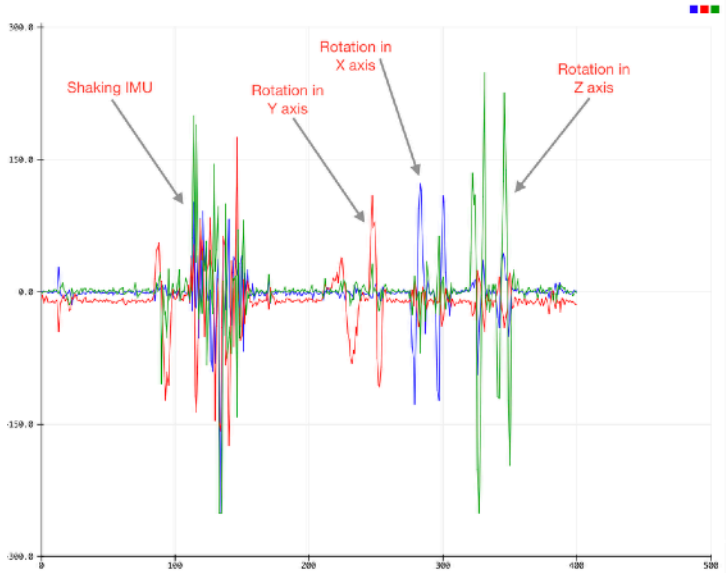
I reviewed source 46 to learn about complementary filters and made the following notes:

- Trigonometric functions use a lot of CPU time (slowing down the processor)

  - The small angle approximation can be used to get around this, since the most important angles are those close to 0
- Motor output (velocity) = (Kp × Angle) + (Kd × Angular velocity)
  - This produces better results than just going off the angle alone, as when the inverted pendulum is swinging back towards vertical the motors slow in advance to prevent overshoot.
- Complementary filter is probably the best bet, reasonably accurate but not over complex (and easy enough for me to understand)

I also reread source 45, which describes how to implement a Kalman filter in C. I decided to use a complementary filter as the maths for a Kalman filter is too complicated for me to understand.

I decided to find a way to plot the accelerometer and gyroscope readings from my IMU on a graph, to check that the readings were roughly accurate. I used source 71 to add the plotting code and the Arduino software to make the plot. I then moved the IMU in each axis separately, to check that each was working. The GitHub commit for this code is called 'Improve plotting code' its first 7 characters are 78d593f. The plots can be seen in screenshots 13 and 14.



Screenshot 13 annotated.png (cropped) Shows gyroscope reading against time in three axes

I decided to order some wood for the back of my robot. I chose MDF (medium density fibreboard) as it is cheap and easy to work with. I bought a pack of 4 A4 sheets of MDF and 100 zip ties to attach the motors to it. Invoices are in the Invoices folder.

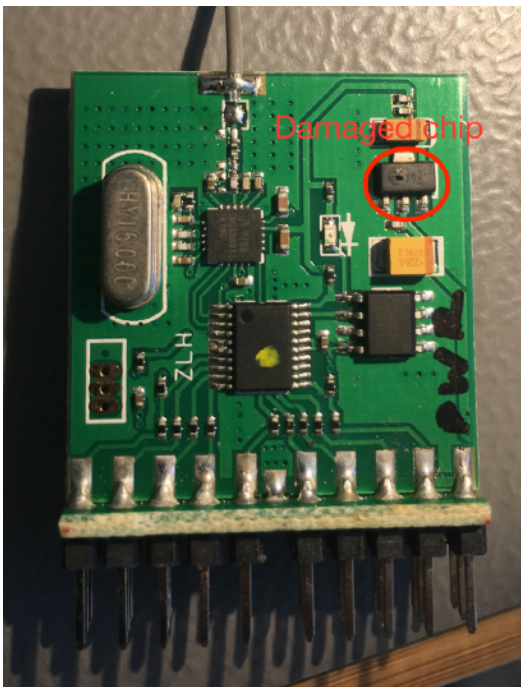
I realised that I didn't know what was going on in some of source 68 (the code I copied for the IMU), so I read through it and improved the formatting. I used sources 70 and 72 to understand the use of the int16_t datatype and the control of the MPU9255 in the code. Whilst reading source 72 I discovered that the MPU9255 does include the AK8963; there must have been some software problem preventing me from using it.

My work today improved my understanding of C/C++ significantly; reading through Kris Winer (source 68)'s code and reformatting it was very useful. Knowing how useful it can be to read through code (even when you don't need to change it) should be useful to me in the future as my university course of choice (Electronic Engineering) includes lots of programming.

17/8/2017

I continued my testing of the IMU today. The IMU must be motionless and flat for calibration[69]. When I was reading through the sample code[68] yesterday, I noticed that it had an accelerometer bias removal calculation that was commented out (intentionally

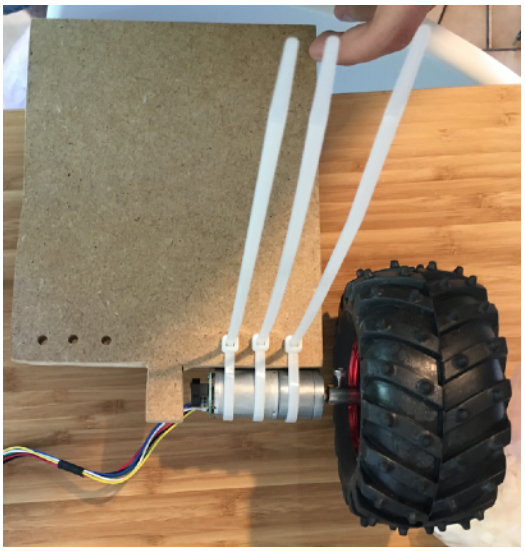
removed). I decided to test with and without this bias removal (see screenshots 14 and 15 for this testing). I recalled that the accelerometer measures force as a proxy for acceleration in g (multiples of 9.81ms^{-2}). Therefore, the vertical axis (z axis) should read 1g, and the horizontal axes (x and y) should read 0. When I tested with the accelerometer bias calculation, the z axis was around 1g as expected but the x and y axes were non-zero. Without the accelerometer bias calculation, the z axis was around 1g and the x and y axes were close to zero. This was the desired result so I kept the code as it was originally (with the accelerometer bias calculation commented out). The GitHub commit for this is called 'No accelBias removal' and its first 7 characters are 006e5a2.



46 annotated.jpg (cropped)
9X receiver with the damaged chip highlighted

Next I decided to connect the Turnigy 9X receiver so that I could receive control signals from my transmitter. My cables wouldn't fit nicely into the receiver with its casing on, so I had to remove it. I then looked up the pin configuration of the receiver so I could connect it. I used source 80, which was of low quality. The source gave the wrong pin configuration and I accidentally damaged the receiver by connecting the wrong pins to the power supply. I ordered a new Turnigy 9X receiver.

While I was waiting for the new Turnigy 9X receiver to arrive, I decided to build the robot's frame. This can be seen in images 47–60. I used a pencil and ruler to mark up the areas I would cut out and the holes I would drill. The first time I marked this out wrong (see image 47) by drawing it in landscape rather than portrait. I just tried again in portrait on the other half of the MDF as it is a cheap material.



61.jpg (cropped)
Completed frame with one motor mounted

I then cut the MDF in half with a hacksaw. I got some of the way with a small hacksaw but I had to find a larger one to finish it as the smaller one wouldn't go all the way through the board (see images 53 and 54). After this I cut out the smaller motor areas, and drilled holes for the zip ties to attach the motors. I then attached the motors by strapping them to the frame with zip ties.

18/8/2017

I decided to add some space on the motor board to attach the encoders (built in to the motors) to the Arduino. I looked up the pin configuration of the encoders at source 24 and added a pin header with power and encoder data lines. This can be seen in images 66–69 (68.m4v is a video of me soldering the motor board).

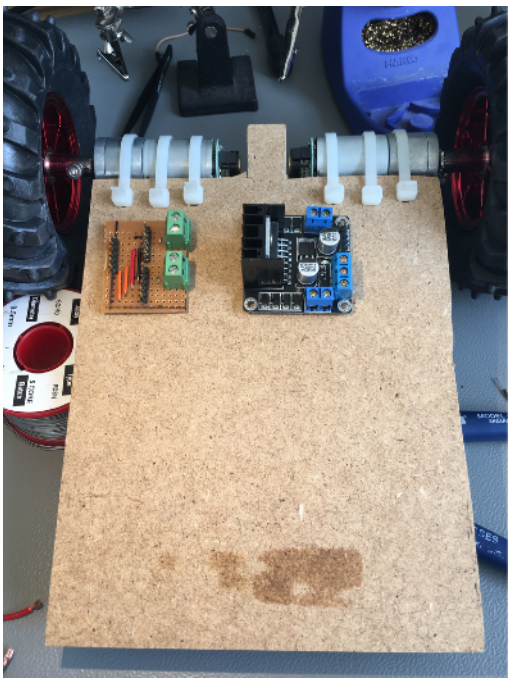
I then mocked up the placement of the motor board and the L298 board on the robot (image 70). I used a hot glue gun to glue the motor board to the frame, before realising I had glued it to the side that didn't have any markings. I decided to continue anyway as I could cover the markings on the unglued side at a later date.

I screwed the L298 board and Arduino to the frame using self-tapping screws, and stuck the breadboard to it using the adhesive backing (image 83 annotated). I then screwed the IMU to the frame and realised that the pins were inaccessible (image 83 annotated). I pried up the breadboard so that I could move it further down and gain some space to attach a cable to the IMU pins, but in the process I damaged the breadboard (images 84–86). I managed to repair it by cutting out the damaged part and carefully slotting it back in (images 86–87). Before reapplying the breadboard I checked if there was enough space for it to be moved further down the frame, but there was not (as I needed room for the LCD as well). This meant that I had to resolder the IMU pins the other way around (compare images 95 and 98 – pins are the other way around in 98).

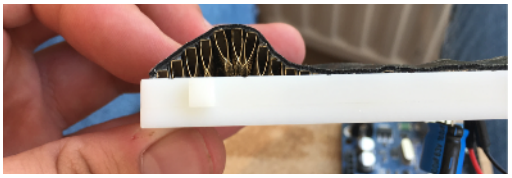
Next I thought about how to attach the battery to the robot. I decided to put a velcro adhesive strip on the robot and on the battery. I put the velcro strip on the front of the robot to help balance out the weight from the components on the back.

I then connected the various parts together. For a more complex project I would have drawn a diagram of this, but I worked the connections out mentally as there were not very many to make for this project. I used a potentiometer to tune the contrast of the LCD[87].

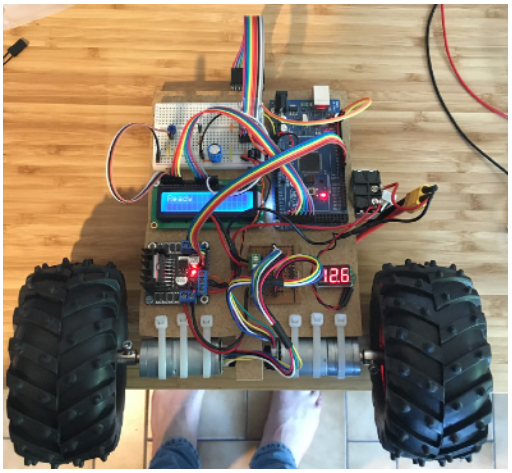
I thought about how I would connect the battery. The issue with lithium batteries is that they must not be discharged below ~3V per cell (9V for my 3-cell battery). I decided to add a small voltmeter module and switch so that I could turn off the robot before it reached this point. I found a small voltmeter module and switch and soldered them to the battery and connector. I had to pry up the motor board and reglue it as it was in the way of the voltmeter module. I also added a button and programmed its detection using source 73.



70.jpg
Motor board (left) and L298 board (right) positions before glueing



84.jpg (cropped)



107.jpg (cropped)
Almost-finished robot hardware

At this point, the robot's hardware was almost complete and just needed the 9X receiver to be finished.

I did some more testing of the gyroscope and found that it was reading the y axis as the x axis. I reread source 72 and discovered that a rotation towards the x axis is indeed the y axis; this is expected behaviour. This can be seen in Screenshot 16 annotated.png.

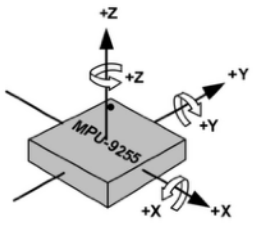


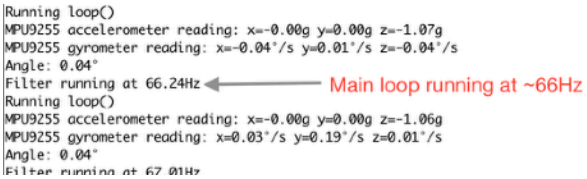
Image from source 72

Having reviewed source 46 I decided to start with the following filter constants: τ=500ms, *dt*=10ms (∴ sample rate=100Hz), *a*=0.980. I decided on these because they are the defaults suggested by the source.

My work today has improved my prototyping and soldering skills. See video 68.m4v for a video of me soldering.

27/8/2017

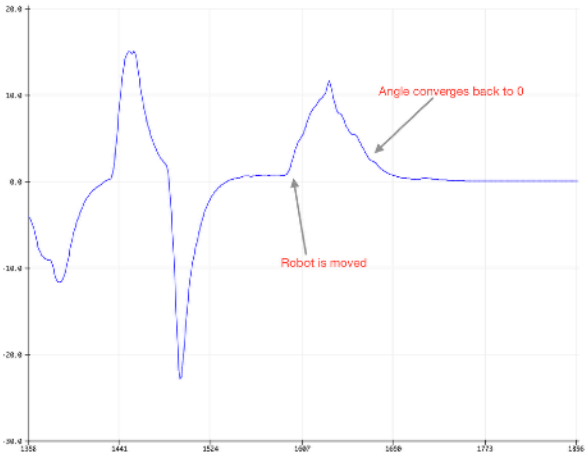
I began today by tidying up the code. This can be seen in commits 'Add filter frequency targetting code' (31dc418) and 'Finish tidying up const variables' (5744539).



Screenshot 18 annotated.png (cropped)

I then checked the maximum complementary filter speed I could achieve. I found that it was around 66Hz so I decided to target a 25Hz filter update rate (1/25=40ms *dt*). I tried to implement code to keep the filter update rate constant (Screenshot 19 annotated.png) but was unsuccessful; the filter kept jumping between two different frequencies. I decided to rely instead on manual trimming using Arduino's delay function.

I changed my mind when I reread source 46 and saw that the filter's time period changes depending on the filter update rate (so a variable update rate could mean variable filter performance). I tried again, this time separating the code out into a separate function (GitHub commits: 'Add filter frequency targetting code' (31dc418), 'Improve filter frequency target by implementing trim' (eeef092)). This time it worked; I must have made a mistake with the previous attempt.



Screenshot 21 annotated.png (cropped)
Shows how angle converged to 0 over time.

I was having an issue with the IMU frequently failing to start up properly; it gave an address of FF, which indicates a hardware problem. I realised that I didn't have any I2C pull up resistors[88] on the 3.3V side of the level shifter. Adding in pull up resistors seemed to make the problem less frequent. I got the resistor value of 4.7kΩ from source 52.

I experimented more with the filter constants. I kept

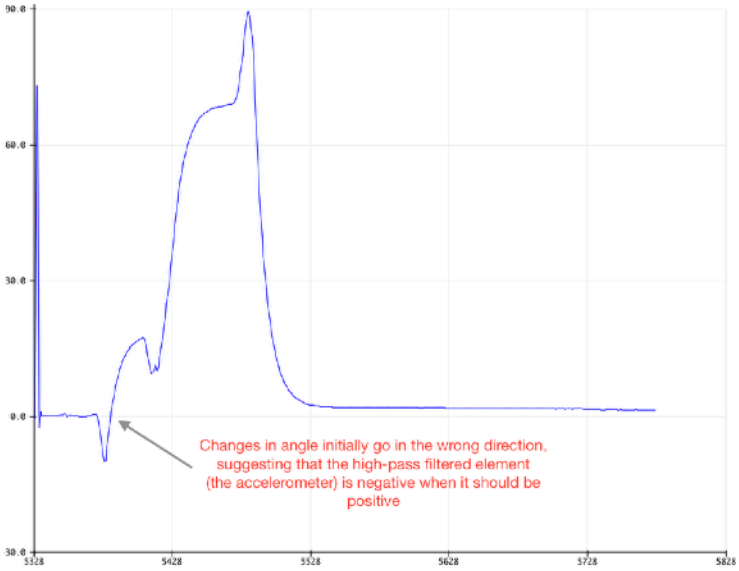
the time constant τ at 500ms, but *dt* was now 40ms so I recalculated *a* ($=\tau/\tau+dt$)[46] as 0.9259. I set the new value of a and plotted angle against time. I found that the angle seemed to be converging to 0 after movement. I tried varying the filter constant a (screenshots 21–24) but I couldn't find an optimum value of *a*.

I added some extra code to test for dead pixels in the LCD at startup. GitHub commit: 'New LCD testing code' (cec4fb6).

I improved my C/C++ programming skills today. The implementation of the filter update rate limiting code was quite tricky but achieving it should help my robot to be more stable later.

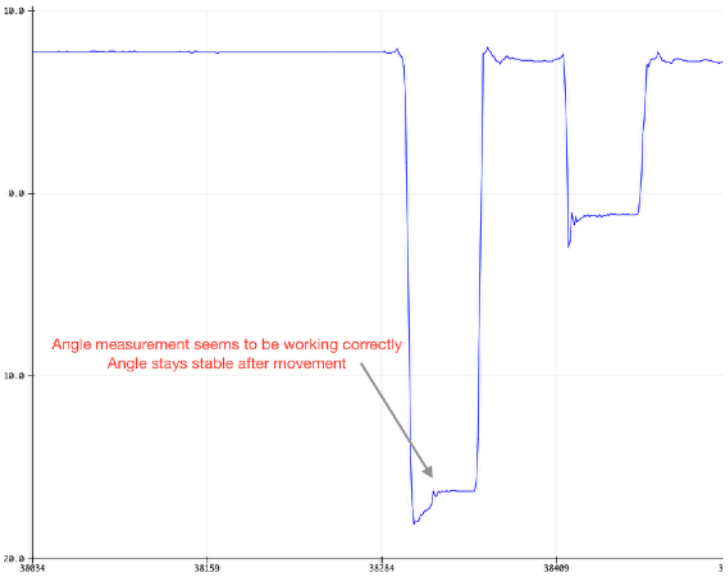
28/8/2017

I continued my testing of the IMU. I set the filter constant *a* to 0 (effectively getting the accelerometer readings only) and looked at the output. To my surprise, at 90° the reading was 1.46, which is approximately 90° in radians. Source 46 confirmed that the accelerometer output is in radians. I therefore found the value of 180/pi[77], which I multiplied accelerometer readings by to get the angle in degrees. GitHub commit: 'Fix angle calculation' (e087c2f). This worked and my reading from the accelerometer was now correct.



Screenshot 27 annotated.png (cropped)

I tested the IMU again, now with a filter constant *a* of 0.98. I found that I had another problem: after an initial change the angle would go in the wrong direction before righting itself. I knew from source 46 that the complementary filter contained a high-pass filter, and given that the error was short-lived, I deduced that the high-pass filtered element (the accelerometer) was negative when it should be positive. To fix this I multiplied all accelerometer readings by –1. After doing this, the angle seemed to be responding correctly after movement. GitHub commit: 'Fix accelerometer component of filter' (5538166).



Screenshot 28 annotated.png
Shows the complementary filter working correctly

I thought that the previous serial speed of 115200 baud might be reducing the maximum filter speed, so I tried increasing it to 2000000 baud. GitHub commit: 'Fix angle calculation' (e087c2f). I found the new maximum filter frequency to be around 180Hz, so I decided to increase my target filter speed to 100Hz instead of 25 Hz. I did this and all seemed fine. GitHub commit: 'Increase filter frequency and trim' (80a90d3).

I was still experiencing issues with the IMU reliability. When the robot was tapped hard, the IMU would loose connection and the robot would stop working. This seemed to indicate a loose connection. I tried replacing the cable between the level converter and the Arduino, but nothing changed. When I replaced the cable between the IMU and the level converter however, it worked much better and tapping it didn't make it stop working.

My work today has improved my problem solving skills. My trial and error technique for fixing problems has worked well and I am more likely to use this systematic approach in the future.

29/8/2017

With the complementary filter working, I worked on the motor control code. Initially I implemented a "bang bang" controller; the output is zero between two set intervals and is at its maximum outside them. In pseudocode: if angle>5° then move forwards at full speed, if 5°>angle>-5° don't move, if angle<-5° then move backwards at full speed. This did not work very well. You can see the initial version in 110.m4v. My initial angles were ±5°. I changed this to ±2° and increased the motor power in 111.m4v, but this still didn't work. GitHub commit: 'Initial motor control code' (0e42dc4).

I decided to try a proportional approach where the motor output is proportional to the angle. GitHub commit: 'Proportional motor control, fix directions' (b5e58e9). I tested different gains (gain = the value that the angle is multiplied by to get the motor output). 120.m4v shows a gain of 4. I did not video every attempt as it would have taken a long time, but they were all similar to 120.m4v; they all failed! My trial and attempt log is below:

- 4
  - too weak, didn't correct strongly enough
- 8
  - too weak
- 10
  - too weak
- 15
  - too high
- 13
  - better but still to high
- 12
  - too high
- 12.5
  - not sure if too high or too low, but not working

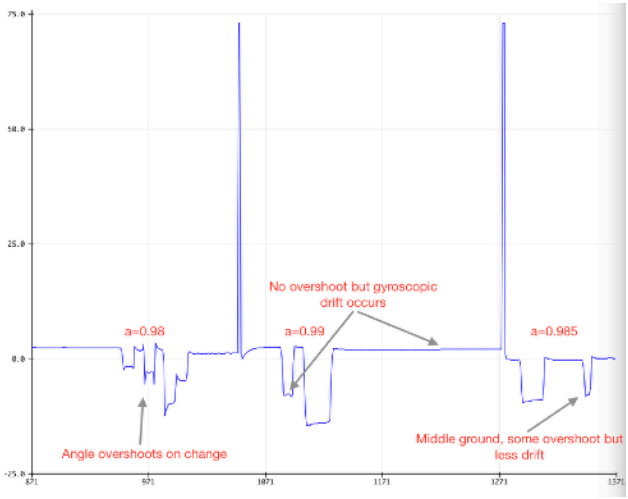
I reread source 46, which mentioned proportional-derivative (PD) motor control. With some searching I found source 79, which states that proportional motor controllers are inferior to PD motor controllers and gives some guidance on how to implement a PD motor controller. I therefore gave up with the proportional controller and tried to implement a PD motor controller.

30/8/2017

I reviewed source 79. My first approach (the "bang bang" controller) was not listed, but my second approach was the 'bad solution' according to the source. It had this to say on the problems I encountered:

> "The problem here is that many times the system will have sufficient momentum to keep going even though there is no more current and no force being applied by the motor. Unless there is sufficient friction, you will overshoot the goal. If this happens once, it'll probably happen on the next pass too – resulting in vibration. This is obviously undesirable since the system never rests at the goal position, wastes energy, and will wear down unnecessarily."[79]

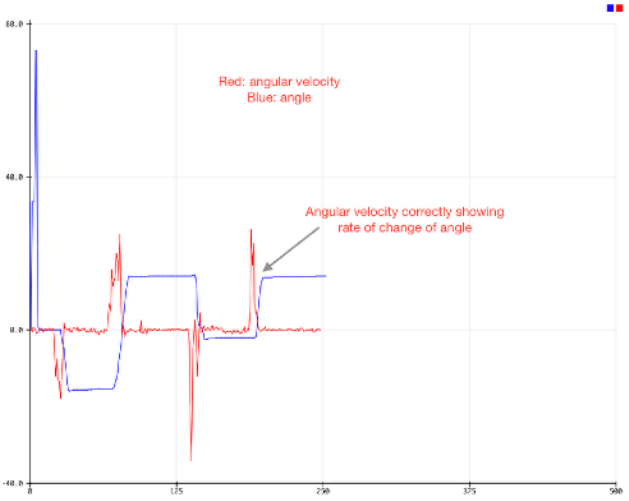
This does sound like what I experienced!



Screenshot 30 annotated.png (cropped)
Shows my experimentation on the value of *a*. The high peaks occurred when the robot was switched off to be reprogrammed.

Before continuing, I decided to make sure that the filter constant *a* was set correctly. I plotted calculated angle against time and varied the filter constant to find which value provided the best balance between overshoot and gyroscopic drift. After trying *a*=0.98, *a*=0.985 and *a*=0.99 I settled on *a*=0.985 as the best compromise.

The proportional-derivative controller required an angle and an angular velocity (rate of change of the angle). To get the angular velocity I could use the gyroscope, but it drifts over time. Instead I chose to differentiate the angle to get the angular velocity. GitHub commit: 'Implement angular velocity differentiation' (256c0db). I then tested this (See Screenshot 31 annotated.png). It seemed to work.



Screenshot 31 annotated.png
Angle and angular velocity against time

I then implemented the PD controller according to source 79. GitHub commit: 'Implement PD controller (not yet working)' (55247b8). At this point it was just a matter of tuning Kp and Kd (the proportional and derivative gain constants) to the

right value to achieve stability.

I used a trial and error approach to find the values of Kp and Kd. I videoed two of these trials: for Kp=40 and Kd=30, see 121.m4v; for Kp=20 and Kd=30 see 122.m4v. I tried around 15 Kp-Kd value pairs but could not find one that would work. This was frustrating but I decided to continue the next day.

Today's work improved my C/C++ skills in numerical derivative calculation, as well as my knowledge of motor controllers.

31/8/2017

I tried increasing the sample time for the derivative calculation. This didn't solve the balancing problem (125.m4v). GitHub commit: 'Decrease frequency of angular velocity calculation' (c5fabc8).

Despite having added pull up resistors to the IMU and replaced its cables, I was still experiencing reliability issues. Seemingly randomly, the IMU would stop communicating with the Arduino and the robot would freeze. I wasn't sure how to fix this.

In addition to this, I was running out of time. I was going to be away at CERN on a Physics trip from the 3rd to the 5th of September, so I had to finish my project by the 2nd to present at EP marketplace on the 6th.

Furthermore, the robot still wouldn't balance. I had tried lots of different filter values with several approaches for choosing them, and still the robot was unstable.

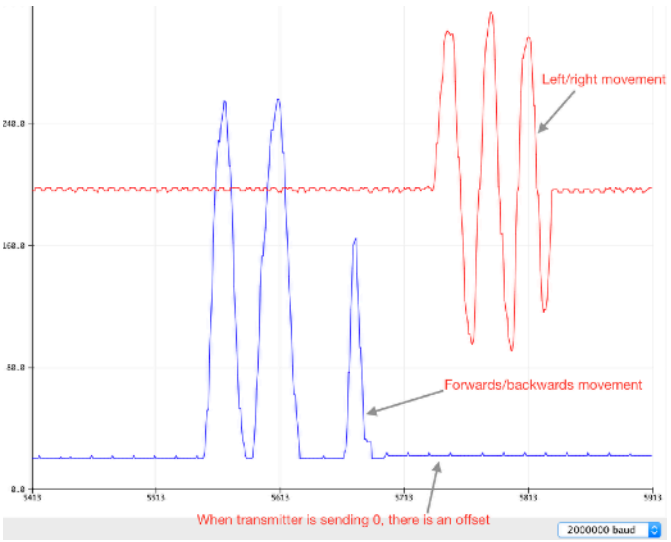
For these reasons I decided to remove the IMU from the robot and convert it into one that drove around with a prop fixed to the front as a third wheel, rather than one that could balance by itself. I wasn't very happy with this decision as I originally wanted my robot to stand up, but I had to compromise at this point. I thought that the decision wouldn't affect my EP mark much as I had documented the knowledge and skills that I had gained from the work that I had done towards a balancing robot, even if I had fallen at the last hurdle.

I removed the IMU-related code from my project. GitHub commit: 'Remove IMU-related code' (8ca5fd8). I still needed to get my robot to receive data from the Turnigy 9X transmitter. I used source 80 to get the pin configuration of the Turnigy 9X receiver, and this time I tested each pin for continuity with power, allowing me to confirm that the top row was the output, the middle was +5V power and the bottom was ground/0V. I needed two channels (forwards/backwards and left/right) so I connected channels 1 and 2 to the Arduino. I then found sources 81 and 82, which describe methods of communicating with a Turnigy 9X receiver. I chose to use the code from source 81; source 82 points out that this is not the best way of programming it as it reduces the efficiency of the code, but this is not a problem as my robot is not completing any other tasks.

I needed to bind my transmitter to the receiver. I found source 83, which gave me instructions to do this. Having bound the receiver to the transmitter, I plotted the output of both receiver channels against time. This showed me that there was an offset; when the

channel should have read zero it was reading around 20. I found the exact values that the receiver output from further testing (see Screenshot 33 annotated.png) – I moved each stick to its maximum and then its minimum and recorded the output. I then looked at the code in source 81 to see how the output from the receiver was mapped to a value. I found that Arduino's map function was used, so I looked it up and found source 84. I used source 84 to set the minimum and maximum values of the receiver output and the offset was fixed. See Screenshot 34 annotated.png. GitHub commit: 'Fix map values' (3a6d9df).

I ordered some googly eyes to stick to my robot, for better aesthetics.



Screenshot 31 annotated.png (cropped) Shows the offset of the receiver output when it should be 0

1/9/2017

I modified the way that the receiver output is changed to motor values. GitHub commit: 'New motor function' (af14f98).

2/9/2017

Today I finished the final code tweaks for my robot. I compiled a list of what I'd spent throughout the project into Finance.numbers, and found that the total was £302.51. In hindsight the price target I set myself was quite ambitious, especially considering that I needed to buy some tools (e.g., the programmer was £92.36!).

I made some posters for the EP marketplace on 6/9. Since I'm away from the 3rd to the 5th this will be the last day that I can work on my EP before presenting it.

6/9/2017

The EP marketplace was today, and it seemed to be a success. I got eight usable questionnaire responses (and two where the person whizzed through and told me they'd "give me a good score"), which I will analyse to find the areas I need to work on. Though the responses were mostly high (probably because most of those completing them knew me personally), the differences between the 8s and the 10s will still show what I need to improve.

13/9/2017

Today I analysed the questionnaire responses from EP marketplace (see "Questionnaire reponses.numbers"). I looked up the RANK function in Numbers (a spreadsheet application) to rank my average scores[85]. Having analysed the data, I compiled a list of things to work on and what action to take in doing so.

Problem	Action
Time management strategies and improvements unclear	Go back and fill in any missing data in timelines. Write about how useful timelines were in PPR/diary.
Evidence that my project is different to my A level is not present	Make a list of skills learned (can take this from EP marketplace poster) and compare to topics in A levels. Highlight any similarities.
Source referencing not clear	Go back through diary and add in source numbers. I need to go back though my diary anyway to flesh out the sections written in bullet points.
Relevance of some sources unclear	Go back through source list. Add more information in 'Did it lead you elsewhere?' as this shows how each source is relevant. Remove any irrelevant sources.
Need more evidence that I have met my objectives	Compare objectives in Design Specification to achieved outcomes (can take this from EP marketplace poster).

21/9/2017

Over the past few days I have been working on brushing up my documentation. I have added some sources I forgot about at the time, and am now checking my diary to make sure skills learnt and problems faced are clear. I am also compiling a document containing some of the evidence images from my project. I will try to get my project printed tomorrow to ensure that there is paper and toner available.