Mech1 Class and Lab Notes

liljag18 - liljag18@ru.is

Fall 2020

Contents

Foreword

Just a little intro before the main meat of this book. This is my, Lilja Yr Gudmundsdottir, book for Mechatronics where all my notes and tips and ideas will go in for the fall semester of Mechatronics I 2020.

There is a book that I will continuously refer to as Exploring Raspberry Pi or the rpi book and that is a textbook about the Raspberry Pi, the tool used for the first part of this semester, and all the useful things we need to know about it. The full name is "Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux" by Derek Molloy but that is super long to write each time so I'll reference it once here to make things easier. The second book being referenced here throughout when it comes to Rust is the Rust book "The Rust Programming Language" by Steve Klabnik and Carol Nichols, with contributions from the Rust Community.

1 Week 1

1.1 17/08/2020

Introduction

- Mechatronics: a mix of electrical, mechanical, and software engineering (something I can do!)

  The electrical part is to power it

  The mechanical part is to allow it to move and give structure

  The software engineering part is to make it actually do something (the most important part in my honest opinion)

- "Mech is about making smart devices taking inputs from the physical world and reacting - smart implies we need a brain" -Foley in class

Rust is high performance but safe in the way it makes sure that all types are correct and the compiler keeps you from making stupid mistakes unlike C++ and Java who let you do them and then just sort of expect you to figure it out on your own. No help there.

Sidenotes:
I need to find lab partner(s) for the final project, could be tricky since I don't know many people but I'll check up on Gunnlaug and see if she needs a partner. Should also sign up for the robotics team at RU, could be fun.
I will also be using rpi to stand for raspberry pi because writing it all out is too much work.

1.2 18/08/2020

Raspberry pi and set up tips :)

- Make sure you ssh the right IP address on the right network once the rpi is set up

  rpi should connect if on the same network, use the research network up at school or hotspot - I'm gonna use my phone hotspot so I can also work from home.

****** READ THE WARNING CHAPTER IN RASPBERRY PI ******
** DON'T UNPLUG FROM POWER SOURCE WITHOUT ***
** SHUTTING IT PROPERLY DOWN UNLESS ***
** ABSOLUTELY NECESSARY ***

- Adafruit pi serial includes an on off button - not included in the kit but might be smart to have to avoid destroying the rpi and to be able to safely unplug it when you can't connect to it - LOOK INTO THIS MORE!!!!
- Backup the rpi regularly on GitHub and all that just in case it does get destroyed

  Could have a setup script on GitHub to make things easier

  Could also have a backup of the SD card on the computer so you don't have to set it all up from the beginning - COULD BE A GOOD IDEA
- QEMU emulates the rpi but is tricky to set up - ONLY DO IF YOU HAVE TIME

Virtual Machine and Rust tips:

- Cisco is a leader in VM products
- I'm gonna use Ubuntu and VirtualBox since I used that last semester for AI and that worked pretty great
- TAKE A SNAPSHOT OF THE VIRTUAL MACHINE BEFORE DOING SOMETHING RISKY!!!
- Get rust to work on your own machine before getting it on the virtual machine, familiarize yourself with it

Sidenotes:

- I HAVE PARTNERS FOR MY FINAL PROJECT, GUNNLAUG AND HER FRIEND AISHA!!!!!
- Need to start brainstorming some ideas for the final project, I have a few already, gonna put them in a google doc but I'll also put them here just in case

- Ideas for final project:

  Paper, metal, plastic sorting machine (sort of like the skittles machine Foley showed us but more useful)

  Dog Toy thrower - useful for my two dogs

  Automatic lights (just like the ones in the hallway up to the lab, the back way we have to take now because of covid)

  Electric scooter (I have one of my own but it would be cool to make one from scratch)

1.3 Tips for setting up rpi and Rust

RPi Set-up

- Rest of instructions after the section of burning SD card and editing it (see canvas under IA 1: Preparing for Success) should be done on the rpi ssh or on monitor (recommended to use monitor for ease of use)
- Send all questions through Discord (turn on notifications for phone)
- need to access SD card in VM but it is easier to do on host machine (Gonna do it on host machine, don't have VM installed on machine where I'm editing the SD card)
- Install Rust using snap and install Visual Studio code on VM
- Boot partition - needed for config, WiFi and ssh
- How to get snapshot of VM:

  Machine -> Take snapshot

Rust Tips

- Variable declaration uses let in Rust
- :i32 - 32 bit integer - defines size and type

  Not necessary but does make it easier to compile
- VARIABLES CANNOT BE MODIFIED WITHOUT MOD AND THE TYPE CANNOT BE EASILY CHANGED LIKE IT CAN BE IN PYTHON OR EVEN JAVA

  Despite that being a bit annoying it does have the plus side of making the language more powerful and robust and easier to compile

1.4 IA 1: Preparing for Success - things to remember from when I went through it - 19/08/2020

- Get IP-address from ifconfig on the rpi on monitor

  Remember to put SD card in and turning everything on before connecting the HDMI cable from rpi to monitor otherwise it wont work! - learned the hard way

  Had to configure WiFi from the monitor because something went wrong, not sure what, but it ended up working eventually after reboot

  IP address for now: 172.20.10.9

- Username is not changed from pi, new password is placed on the bottom of rpi
- N1 - Hostname is liljapi
- When trying to select a setting (i.e. language) on monitor use space, enter and * do not work!
- ERROR: "No locale found" error

  FIX: Just finish all the settings and reboot, everything should be a-okay after that

- "sudo apt update" - shows what needs to be updated, good to check up on every now and again

  ERROR: Some of the necessary updates didn't show up when I typed this in

  FIX: put network in again in sudo raspi-config (the thing I did before to get to all this configuration (see IA 1 description on Canvas)

  rpi-eeprom-images not needed (can do "sudo apt autoremove" to remove but I did not do that and everything seems to work OK so far)

Then just do sudo reboot and everything is all set up! Took a little time but everything went fairly smoothly aside from the occasional error here and there.

Quick access to rpi and using it properly
Connect to rpi use ssh pi@IPADDRESS or in my case "ssh pi@172.20.10.9". If getting the new IP address from terminal use "ip address" unless net-tools are installed.
To turn off properly do "sudo shutdown -h now". This shuts the rpi completely off. If you only want to reboot it (and therefore don't need to unplug and plug in again to start it again) then simply do "sudo shutdown -r now" or "sudo reboot", both work.

Visual Studio Code Quick Access
To open VS code do code& or just open the app itself, both work.

To create a new project do ”cargo init <project name>” and everything will be set up correctly for Rust to be able to run.

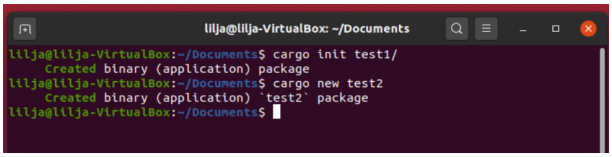

Figure 1: Testing creating a project with cargo in two ways

1.5 21/08/2020

Met my lab partner today over Discord, he's an exchange student from France names Julien. We decided to do this lab on our own rpis at home to make sure that everything worked for us both. We put everything on a Google Doc that we shared to show that it worked for us both.

The lab due date was extended by one week which is good because I was unable to get GitHub classroom to work for me. Kjartan is currently helping me get everything sorted, there is some sort of error on the back-end side according to him.

I had quite the set-up with my monitor and rpi and keyboard

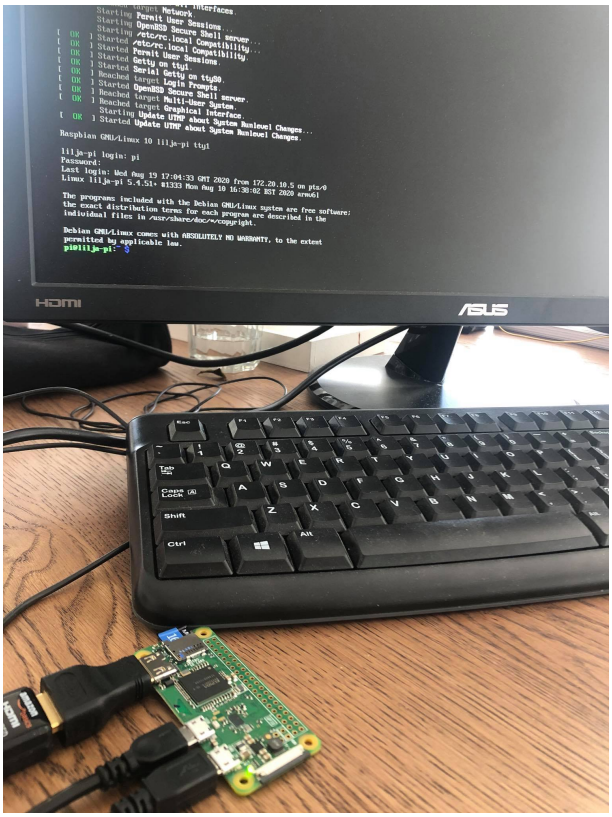

Figure 2: My set up to set up the raspberry pi, lots of wires all over

2 Week 2

2.1 24/08/2020

GitHub was talked about in class, I've used it often with my other classes for the past two years so should be relatively easy, like riding a bike. GitHub classrooms also finally works for me!
Lab closes at 17:00, no exceptions due to COVID so START THE LABS EARLY!!!!!
The soldering class will come later, need to attend it for the safety part, more info will come out later on this week, keep in tune with that so I can actually do something in the lab itself. UPDATE: It will happen in the lab itself, Friday 12:40 according to Kristofer.
Easy to use soldering comic reference. good to have in mind when soldering.
Why choose the raspberry pi?
"Because it is one of the cheapest, single board, Linux computers that is well supported" -Foley
Check out Gumstix, it is not cheap but it is a very good, robust, well supported computer, worth checking out, good for industrial projects. More expensive for lower performance stuff.
Rpi needed to be a linux computer that could be placed in every home in the UK (where founder of rpi Evan Upton is from).
Working with embedded linux:

- Tiny, tiny amounts of memory, about 512 MB of memory
- Limited processing power
- Limited memory
- Small amounts of storage

So all in all I need to pay attention to memory and what is used for what (CODE SMART, NOT FAST and KEEP IT SIMPLE STUPID (K.I.S.S), words to live by as a programmer).
Make interactions between parts robust, it's more difficult to minimize the amount of parts you use, sometimes impossible, but it is possible to make them work together more efficiently. Make them work together and fail in an intelligent way or as Foley puts it "Fail smart or fail safe". Good to also think in what could be the worst case scenario.

"Hoping for the best, prepared for the worst, and unsurprised by anything in between." - Maya Angelou

Read the Therac-25 paper, many lessons to learn from the failures of a machine that promised great things (curing cancer) but had too many software bugs to be as efficient as it could have been. In fact, these software bugs in mechatronics (unlike in most cases when it comes to software with regards to websites and

apps) can kill people instead of saving them. SO BE CAREFUL IN HOW YOU FAIL.

Rpi runs linux which is released under GNU public license - basically a fancy license that allows the free use of it, very handy for people who want to make use of this resource such as ourselves. Only thing is that everything we develop with it has to have the GNU public license. It is there to protect you from lawsuits from people using your code without your knowledge and then blaming you for something you didn't even know about.
ARM: the processor in rpi, it stands for Advanced RISC Machine (had a different meaning before: Acorn RISC Machine, Acorn being a computer that was sold in the 1980s but not anymore. Instead there is apple because apparently naming companies after edible things became a sort of trend from the 1980s to the early 2000s). Smartphones run an ARM processor (everywhere but China).

Bit of a difference in running embedded Linux vs normal Linux. On a standard computer it has the same peripherals with minor changes to the device drivers. But on an embedded Linux, the things that that chip has could be completely different between different models of ARM/other architecture. To address this difference you need a DTBS (Device Tree Binary) file which talks to the Linux kernel and lets it know what tools are at its disposal in the processor. This helps keep the basic Linux kernel the same with different configs.
Linux has a monolithic kernel (one kernel you interact with) and a majority of critical things is embedded in that kernel. There is user space, kernel space and then physical hardware, the latter two aren't easily accessible because users don't have much need to change that part of it, could screw things up. Keep in mind file permissions as well (owner, group, everyone else levels - rwx). To access a folder (directory) you need execute permissions, important to have that in mind.
CHECK OUT CHAPTER 3 IN THE RASPBERRY PI BOOK "EXPLORING RASPBERRY PI FOR MORE INFO ON LINUX AND LINUX COMMANDS - Nice little refresher from Computer Structure course.

Electronics Reminders

- Resistance: how hard it is for current to flow through a circuit
- Voltage: how hard the circuit itself is pushing
- Rpi outputs very very small amounts of current at 3.3V
- Need amplifying device to do more with the raspberry pi than just turn on an LED in order to switch on and off or light many LEDs

  Most common tool for this is a transistor

  BJT (Bipolar Junction Transistors) is often used and talked about in the book "Exploring Raspberry Pi" on page 132 and beyond.

Field Effect Transistors (FET) are another example talked about on page 133 in the book and are voltage control devices. The latter is the preferred tool to use because the battery life can be increased because it creates a voltage difference without inducing a lot of current flow.

MOSFET chips are powerful tools but can be easily damaged or destroyed by static electricity.

opto-isolators could be useful to prevent the rpi from exploding, always good to have tools that prevent explosions, good for noise and working with high power

There is also the problem of noise when pushing the button. This can be fixed with a filter circuit which makes use of a capacitor or Schmitt circuit where the voltage has to pass a certain threshold to be considered high or low. Then there is also software that can take away noise in many different ways.
Later we will play around with PWM (Pulse width modulator). This allows us to be 50% on and 50% off. If this was done straight on the linux computer then the processor would max out.

2.2 25/08/2020 - Rust language

Remember safety course at beginning of lab on Friday, need to attend to use lab!
Useful tips about Rust and the way to program in it

- Analog discovery 2 - good tool to have from diligent

  There is also oscilloscope and more which is used in a few labs.

- toml: Foley said it stood for "Tom's something Mark-up Language"s" but according to the internet (aka Wikipedia it stands for "Tom's Obvious, Minimal Language"

  In Rust "Cargo.toml" is the config file that contains all the dependencies and packages the program needs

  It's similar to the "requirements.txt" file in python or node_modules directory in javascript

- println!("") -> this is a macro, not a function, the exclamation point marks that

- Macro is good for taking in multiple arguments

  It is difficult to write

  Good for no repeat code, like creating functions for operations that are done often in python, reduces amount of code line

  Meta programming

- readline().expect("Error") -> works like try...catch in python

- rng - random num generator

  rng.gen_range(1, 7) -> check documentation in book to see what the actual range is or double check what range it goes to in actuality

  Update: Range in Rust often goes to one less than the biggest number

- Casting is not automatic except in very special cases, safe thing to do it to specify what type a variable is to begin with

- guess.trim().parse().expect("Please input a number")

  trim() -> take away spaces (like strip in python)

  parse() -> checks what this value is

  expect() -> error catch

- guess.cmp(&secret_num)

  cmp() -> compare

  & -> passing a reference (pointing to value) and not passing the original value itself

  Rust refuses to point to things that don't exist, makes sense

- Match statements -> similar to switch case in python
- Use break to break out completely
- Variable scope is the same it is in Python and Java

Enums define kinds:

- What is allowed in a struct (aka data structure)

  Similar to dict or hash function
- Make sure to be exhaustive

Gotta keep in mind that Rust has no garbage collector unlike many other languages I have used. This can be useful real time/time critical systems such as robotics where certain parts have to react quickly to sensor inputs and program calculations. Overall it helps prevent delay. When we start using the Arduino we will use a version of c++ that has no garbage collector just like Rust. This means that we need to be more careful about what objects we trash and when since it is up to us as programmers.

The reason why variables are so annoying in Rust (so type sensitive and ownership sensitive) is because we are dealing straight with the stack knowing the exact size of each variable (strings have a different length than ints). Otherwise we use the and operator to call the variables as pointers to the heap.

Don't make use of clone all the time (even though it might seem like a good idea since it copies everything without you having to write everything up again). It takes up memory and time so not worth it.
Remember to add debug to VS Code! - Backtrace/stacktrace forces debug as well.

2.3 28/08/2020 - Lab 2

Met my lab partner Julien and we got started on lab 2. Things were going alright but pretty slow. We barely got through one of the main tasks before we had to go home for the day.

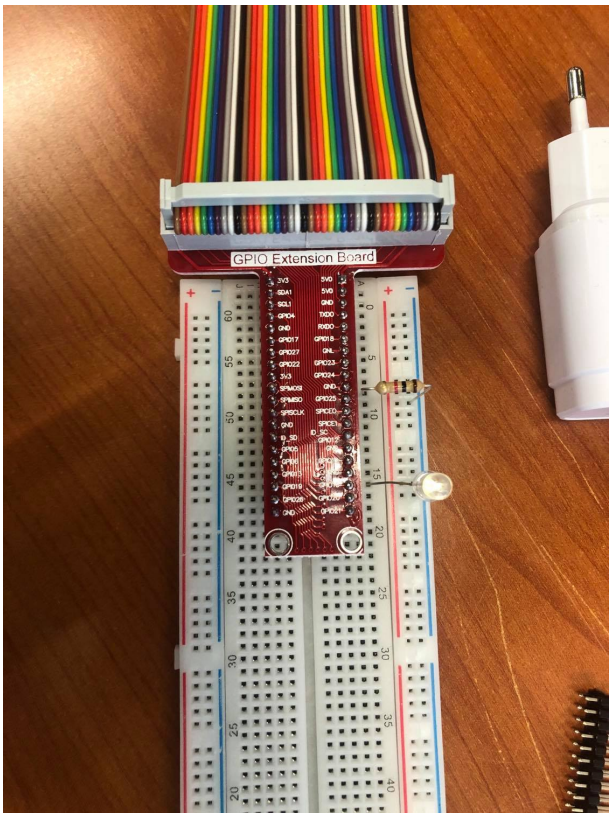

Figure 3: My first circuit of the semester! A single LED and resistor

It took quite a long time to set the environment up. We got Kjartan the TA to help us set up build_to_pi file correctly so that we could compile and run our rust code on our pi.

3 Week 3

3.1 31/08/2020

IMPORTANT - tilde gets you to the home directory

Kjartan has made it super easy to set everything up (which I did last lab session). The way it is setup makes it build on the rpi, not on the processor. "cargo clean" removes target directory that is created upon build. In build_to_pi the main code under src is put in the file "rust_template". In this build_to_pi file there is a line with sshpass that has "-x -t" which means the program will terminate upon completion so it doesn't run forever. In bash then the "$" means a variable follows.

Check out standardized licensing links on canvas, necessary for later on when developing code/robots/software and how public/private the tech should be.

3.2 01/09/2020

- Variable types need to always be defined.
- Rust is just dots unlike c++
- ..user1 to reuse the rest of what is in user 1, saves time and code and errors.
- Rust optimization to keep you from having to type things again and again.
- Tuple structs have fields but no names. Helpful to do things fast. (see chapter 5) [derive debug] - this structure derives from debug traits.
- Rust is sort of an object oriented language
- self important to know where the object is functioning/referencing.
- Use impl, make use of that in Lab2 to create fn gpio()
- More parameters = think of ownership and memory management
- static functions = associated functions in rust (section 5.3)
- closures are functions that you can pass as variables (lambda functions) (section 13) - super cool, now exists in c++, dangerous in c because you are messing with the stack
- packages crates and modules to divide where things live in a smart and useful way
- Everything is by default private (module and crate) for security reasons (section 7), must say that functions are also all public, not just the module, that way you are always aware of what is private and public - use "pub" in front for public
- super is used in rust modules as well, just like in python classes - go up one level, no need for an absolute path (sect 7)
- use - bring modules into scope (sect 7)
- Rust wants to reduce the amount of stuff you cut and paste
- as is used just like in python and all that (sect 7)
- mod to access modules (sect 7) - modularize increase = increase robustness, separates the pieces like classes do

3.3 04/09/2020 - Lab 3

We were unable to start lab 3 because Lab 2 was still unfinished and neither me nor my lab partner had managed to get any further with it since last lab was mostly spent setting everything up.

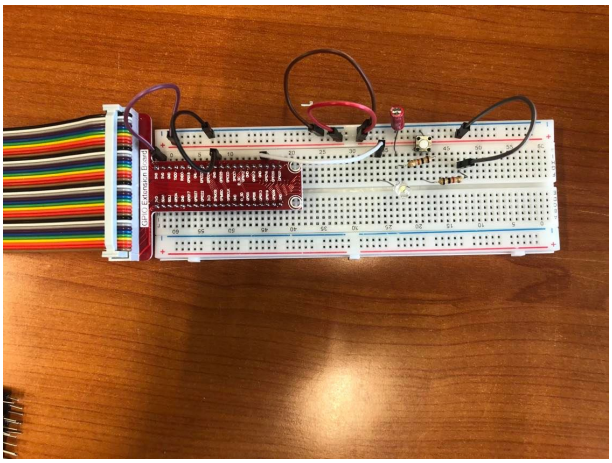

Figure 4: An LED with a button set up to decrease noise when pressing down on a button. Without it the input of the button is very hard to read and utilize.

We did manage to finish all the main tasks and load the videos to YouTube but we still have the hard and advanced tasks we need to finish by Monday.

3.4 06/09/2020 - Lab 3 and 2 (continued)

Sorta stressful getting everything done on time because there are two Labs due at the same time. I'm also not able to find any good programs to create schematics with.
I ended up doing most of the work on these two labs, since Julien was still setting up his raspberry pi and the environment on his computer. It got to be super stressful and I do NOT want to repeat that again.

Figure 5: Image of a bee that I took the other day. It is both calming and also interesting how they hang on the underside of a flower to sleep. Their legs remind me of the legs of those self stabilizing robots. And it's cute

Made a beautiful circuit to imitate the Larson Scanner. It took a lot of LEDs and I need to look for longer wires or strip my own at some point because these little guys barely reach halfway across the breadboard.

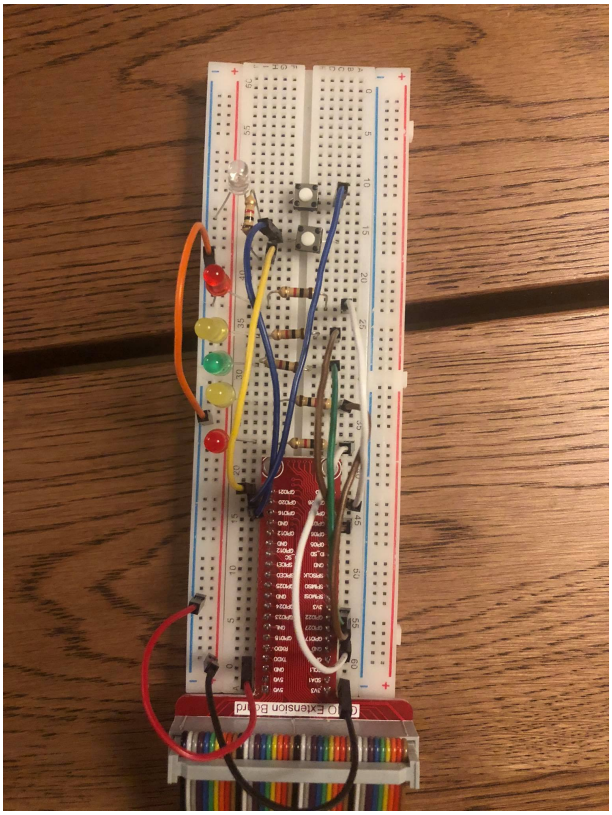

Figure 6: Larson Scanner with a lot of resistors (since there are a lot of LEDs)

4 Week 4

4.1 07/09/2020 - I2C and more

I have some more ideas for the final project:

- Neural network AI self driving car - a car that can follow a line by training it through machine learning
- AR view remote control car - like the ones Nintendo is coming out with soon to simulate Mario kart in real life
- Cocktail maker - great for parties and all that

Foley wants us to be creative in the assignments and to think about what design works best for the task at hand. I unfortunately don't know a lot about circuits so this is proving to be a bit difficult but I'll work that out eventually.

Use peripherals (some element in the micro-controller that takes timing constraints from you and, for instance, turns an LED on and off) to make your life easier - this is the PWM (Pulse Width Modulation - useful code on page 264 in the book) on the rpi, found in many micro-controllers. Use Wire library on Arduino and Kjartan's code for rpi rust for easy use. When using this to control a servo from the raspberry pi then we need to use a transistor to amplify the output from the rpi in order to make it move. They are noisy and you want it only to connect to the signal pin on the micro-controller but have a completely separate power source.

Talking to another micro-controller can be useful and buses are used for that. We will be using I2C to begin with, not very fast but super easy to work with. Depending on length of wire when using I2C an extra capacitor because otherwise the signals could bounce around and get messed up. SPI is also usable, a bit faster than I2C. I2C is already enabled on the rpi (enabled during setup). Addresses for reading and writing are different (by one). Check out page 310 in the rpi book.

SPI is great for speedy connections but it is finicky about the quality of that connection and clock.

Older devices that use Serial make use of UART (Universal Asynchronous Receiver/Transmitter). This way you can communicate between two devices without having a clock (USRT is the version that has a clock - synchronous). The clock isn't necessary because the devices simply agree on what the timing should be instead. Be careful with the voltage when working with UART.

Logic level converter (LLC) acts as a firewall to protect your rpi from being exposed to voltage that is too high. There is no internal clock on the rpi to keep track of the time when the power is shut off. It goes to default value of 0 = January 1, 1970 when Unix was first developed. Bluetooth is just a Serial device.

Closures are useful ways to create helper functions without giving them names, just what parameters you want to put in - **check out chapter 13 in**

the Rust book. There is no difference between closures and lambda (those useful things used in python and other languages).

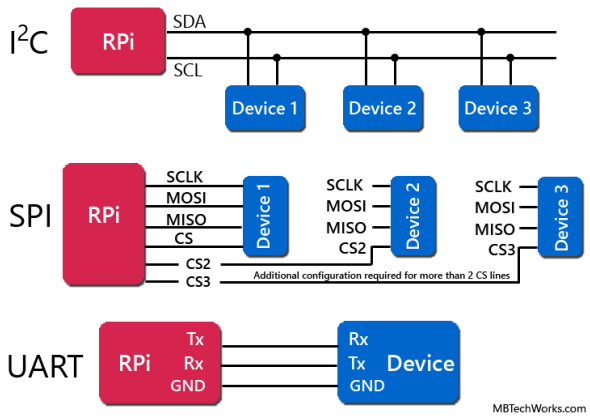

Figure 7: Good explanation of the pins for I2C, SPI, and UART from https://www.mbtechworks.com/hardware/imgs/uart-spi-i2c.png

4.2 08/09/2020

Always good to know how to program the system to make it wait for a signal with threads (not like when making things happen at the same time with threads). Graphical interfaces use this to make it quick to respond (responsive) and be able to do many things at once because we have limited processing power. Users get very impatient and that can lead to broken systems which is never good. Check out portable threads. Too many threads accessing the same data space at the same time can lead to annoying problems - can lead to blue screen of death. There is a wall between the kernel and user space to prevent such crashes.

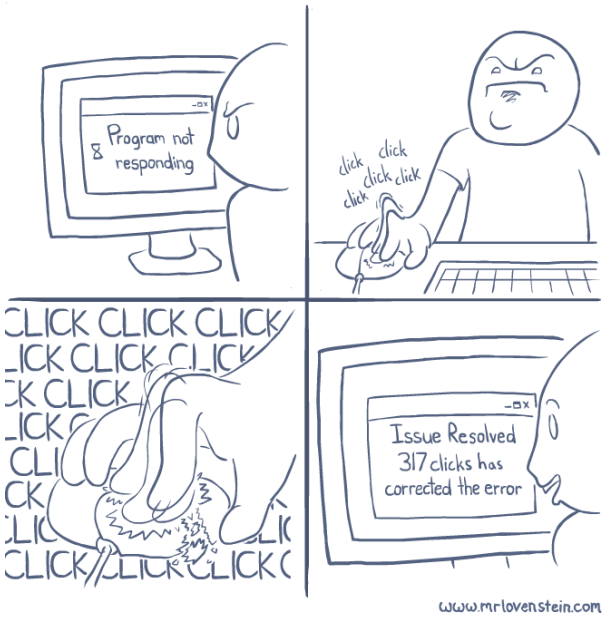

Figure 8: Funny comic from online from Women's Coding Collective (https://wearewcc.tumblr.com/post/153496406150) about how systems DO NOT work.

"Do not communicate by sharing memory, instead share memory by communicating" - From Go language documentation, prevents two processes from accessing memory in a way that is unsafe. Have parallel things operate by "passing notes" (Foley in the lecture) - message passing in Rust.

DO NOT POKE A LITHIUM POLYMER BATTERY WITH A KNIFE, BAD BAD IDEA - also don't overheat, burns in both cases with a fire that cannot be put out. Our laptops are basically bombs waiting to happen, Teslas are like larger bombs.

4.3 09/09/2020 - IA4

I enjoyed this IA thoroughly, it was a bit tough but using an accelerometer and dimming a light in an effective manner was pretty cool to do. There are also so many possibilities with the accelerometer and I can't wait to learn more about it so I can hopefully make use of it in the future.

PWM Resource

ADXL345 Follow the set up steps and check out page 23 (table 19) for specifics on what address on the ADXL345 does what.

4.4 11/09/2020 - lab 4

- Oskar (the TA) recommended to use the oscilloscope to debug circuits, easier than checking each and every component manually
- Managed to finish everything except the Advanced Task 1 and 2
- The accelerometer is so cool!!!!

  Controlling the motor using the accelerometer was like something out of a sci-fi book or reminiscent of telekinesis from comic books

It's kinda difficult to find which way is what in terms of pitch and roll. The reason why we had to figure that out is because there is code already created from Kjartan for the ADXL345 with methods that get the pitch and roll and therefore make it easier to read from the sensor and use the data more efficiently.

Figure 9: Here is the accelerometer, the x-axis (the one up and down on the image) is the one used for pitch calculations in Kjartan's code while the y-axis (the one going side to side on the image) is the one used for roll.

4.5 13/09/2020 - lab 4

ALWAYS REMEMBER TO TURN ON ACCELEROMETER THROUGH I2C OTHERWISE IT WONT WORK!!!! - had trouble in the Lab 4 because I accidentally commented out the code that turned it on and shut off the raspberry pi before starting it up again, essentially turning the device off and never back on - took a while to debug but I did it!

I also made a very pretty claw for the lab which I created late at night because even though this is a lot of work I'm enjoying myself too much to stop.

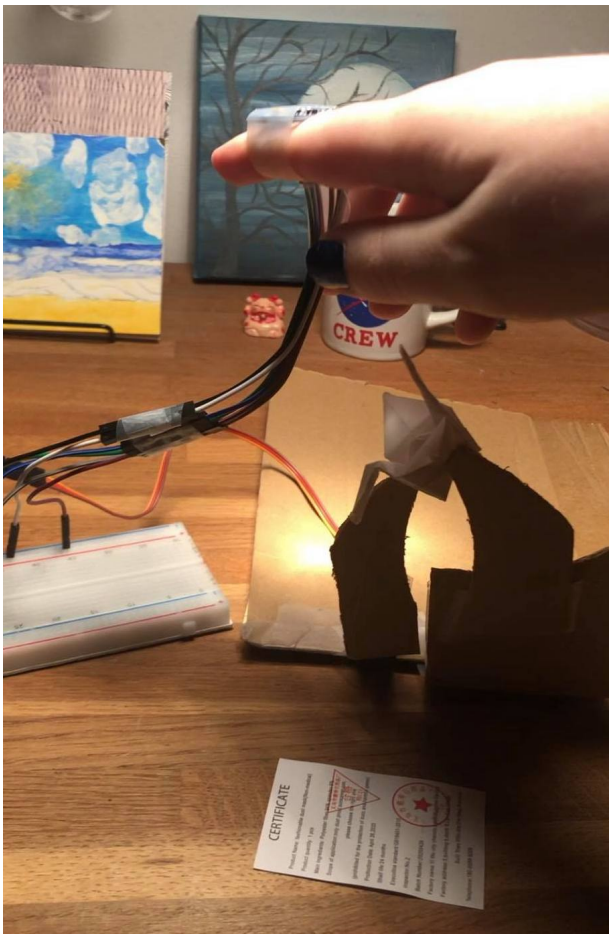

Figure 10: My oh so pretty claw gripping a paper crane as I control it with an accelerometer.

5 Week 5

5.1 14/09/2020 - CAN Bus and more lecture

Use loop-back to test that the data is being sent when trying to communicate with other micro-controllers. Serial not typically used for large amounts of data. Easy to work with but the data is not nicely structured, you need to figure out for yourself what the data is doing. CAN Bus on the other hand (used in industrial motors and cars) has a high reliability of getting data to go where you want it to and has a timing constraint. Again LLC is very good to prevent the rpi from going all fire and brimstone on us due to the Voltage difference between the Arduino (5V) and the rpi (3.3V).

Rpi does not have an analog to digital converter (ADC) to read sensors but luckily the Arduino does have that. Need to learn how to calibrate the sensors based on output from said sensor and the current value of the condition you are attempting to measure. Analog sensors are most reliable and difficult to shift around value wise.

Having two micro-processors talk to one another you can do two things at once instead of faking it like with threads - real multiprocessor. The smaller the voltage range of a device the more likely noise is gonna cause some trouble.

With Arduino you need to use the old bootloader to connect and upload programs to the Arduino.

Improvement on cocktail idea - voice/app controlled due to the current Covid climate.

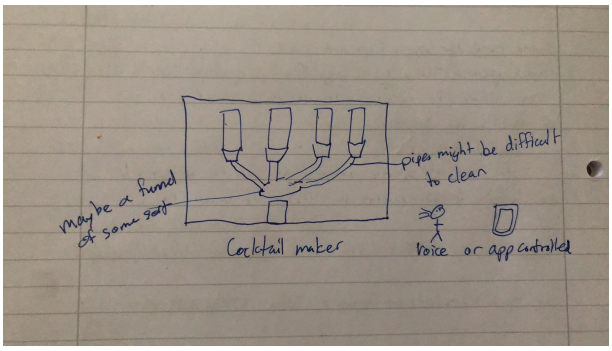

Figure 11: Quick sketch of the cocktail maker idea and how it could be structured and a few things to have in mind about it

5.2 15/09/2020 - Collections

Check out "emacs -m" in terminal, cool little editor, minimal interface, reminds me of nano - good for if you need to change a program on the rpi itself but don't have access to an IDE. This however needs to be installed, need to try it on the rpi later on.

Collections help you collect data together (hence the name collections).

- **Vector**

  Need to define type of data that it's gonna save

  You push and pull values

  Drop a vector by letting it get out of scope like with so many other variables in Rust

- **String**

  Char is defined differently depending on coding language

  Old C only dealt with 8 bits, couldn't tell difference between international characters

  Read section in Rust book about strings to better understand how Rust deals with this

  ".chars()" allows indexing into string

- **HashMap**

  Equivalent to a dictionary in Python

  Unique key paired with data

  When a data has been put into the HashMap then the HashMap has gained ownership of that data

  Use ".get(var_name)" to get data

5.3 16/09/2020 - IA5

MY RASPBERRY PI DIED! Well not completely, the SD card just went kaput and I was unable to finish the individual assignment which was unfortunate. Today's IA did not go as planned, I need to start having a separate SD card with everything already downloaded. I was also unable to reset the SD card because I cannot for the life of me find my USB to micro USB converter for the keyboard. Luckily my good friend Gunnlaug is gonna allow me to take a look at what she did in the individual assignment tomorrow so I'm on the same page as everyone else for the Lab and can understand what's happening.

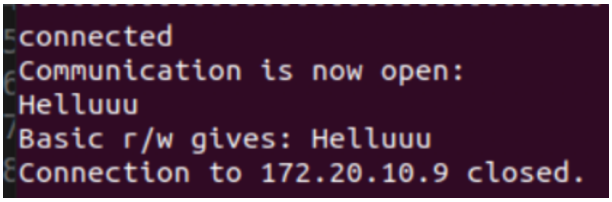

Figure 12: Connection was reached though before my rpi died

5.4 17/09/2020 - lab 5

We started our lab today. Ended up not being able to use Serials to communicate between the Arduino and the rpi (for temperature and light sensor) so we ended up using i2c which is slower (and a bit less reliable) but worked in the end.

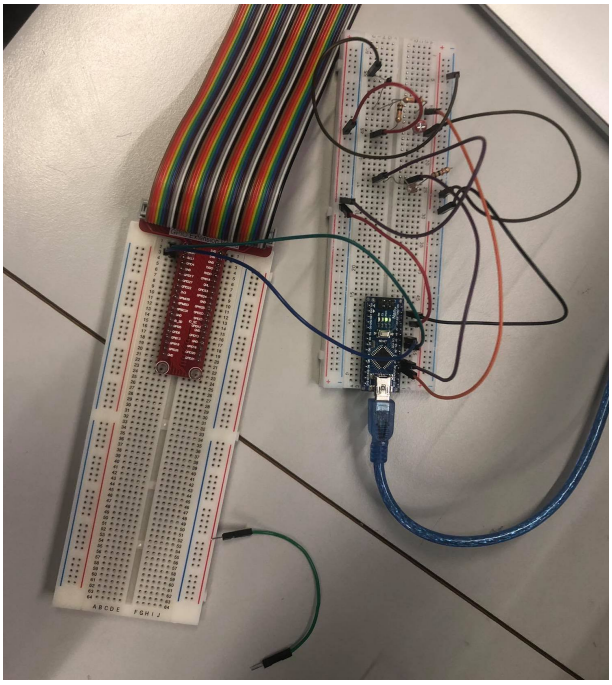

Figure 13: The raspberry pi and Arduino set up so that they can communicate through I2C

6 Week 6

6.1 21/09/2020 - Motors

- We have a hobby servo in our kit usually used for remote controlled airplanes and cars

  Not a lot of power, not super accurate, but respond fast

  Sparkfun has more of a selection (this is where we get all our components from) - **Generic High Torque servo is cheap and useful**

- There are positional servos and continuous rotation servos

  Positional allows you to move to a precise location based on what degree you want to be at

  Continuous rotation is given a velocity and direction

- **Need to be aware of what kind of motor/servo you need for the design you are creating based on torque and speed (degrees per second)**

6.2 22/09/2020 - IA6 started

Started a bit on IA6, it was easier than expected but didn't have time to finish it today. I did however learn some important things about the different motors:

- **Servo**

  Pros: Work in AC or DC drive, no vibration or resonance issues, great precision and efficiency between 0 and 180, speed and position feedback, fast

  Cons: Limited range of precise motion (0-180), can be more expensive than a stepper motor, low torque

- **Stepper Motor**

  Pros: Slow accurate movements, high torque at slow speeds to keep accuracy as much as possible, high reliability

  Cons: If a step is lost then there is no recovering from that error since it doesn't keep track of the position it is in, vibrates as it moves

- **DC**

  Pros: Good for fast continuous rotation, high torque, good for large loads, big speed control, constant torque

  Cons: Doesn't know what position it is at and doesn't really care, just keeps on rotating

6.3 23/09/2020 - IA6: Motorology 101 continued

Finished up the IA6, super easy (bit of trouble with the stepper motor program but went up to school and worked with Aisha and Gunnlaug on it and then it worked perfectly). I also learned a lot about PID systems.

- **P** - Proportional, basically how much are you gonna minimize the error by. If this is too small you might not minimize the error enough and make your system not responsive enough but if it is too big then you'll over-compensate and create a new error and make the whole system unstable
- **I** - Integral, handling errors that occur over time. Make this too big and it becomes less adaptable to current changes, very often the cause of instability in PID controllers
- **D** - Derivative, shows you how the system is behaving between time intervals. Helps improve stability but many motors do not have this part

I got it from this website: PID Robot System.

6.4 24/09/2020 - Lab 6: Design the Lifting Arm

Today was rough. Wiring for a DC motor is super complicated, with a lot of components.

- **Diode** - Reverse protect the power input. This will clamp the reverse voltage to a level no greater than the forward voltage drop (according to Aisha and Gunnlaug). This prevents other components from being stressed by the reverse voltage which is good because stress is bad no matter what.
- **Capacitor** - Used to filter the pulsating the DC output to make sure a nearly constant DC voltage is supplied. The negative side goes to the ground!
- **Polyfuse** - Protects against over current faults in circuits. Important to make sure the DC motor isn't destroyed if the circuit short circuits.

This took forever, and took a lot of tries

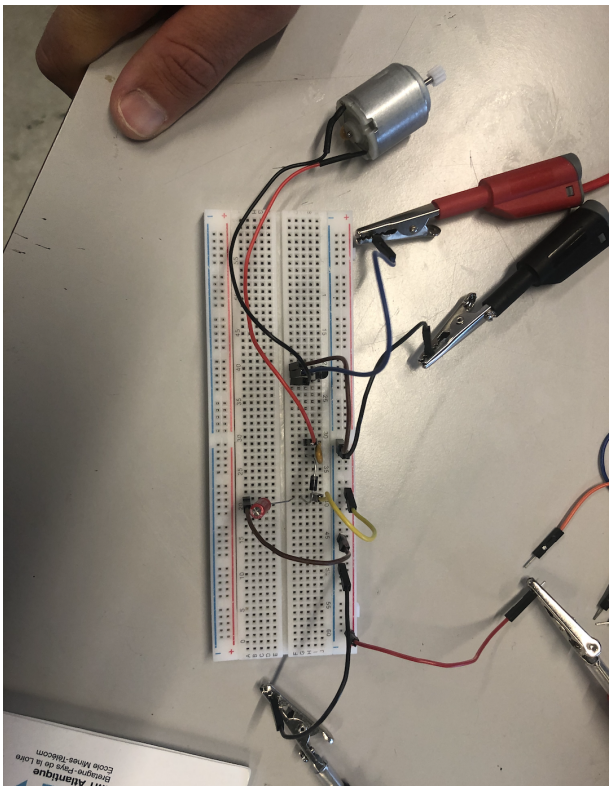

Figure 14: DC motor schematic

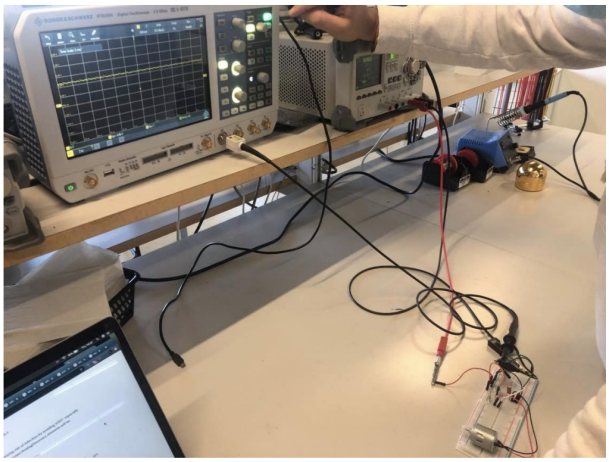

Figure 15: Oscilloscope for Lab6, understanding it a bit

I didn't use optocouplers at first but checked out what the voltage looked like with the oscilloscope and it seemed pretty smooth.

6.5 25/09/2020 - Lab 6 continued

Kept going with the lab, managed to get the potentiometers and fixed the circuit. Added the optocouplers to the circuit which separates the raspberry pi from the DC motor and protects it from the 12 V needed to drive the motor. It prevents the 12 V from flowing back to the rpi and frying it. My partner Julien and I didn't need it yesterday because we didn't use the raspberry pi to control the DC motor but instead a simulated PWM signal.

We assembled our new and improved crane but it didn't go as expected and needs a lot of work.

Julien kept on calling me a "clever girl" which I found to be super condescending. He is not a good lab partner and I can't wait to switch groups next week.

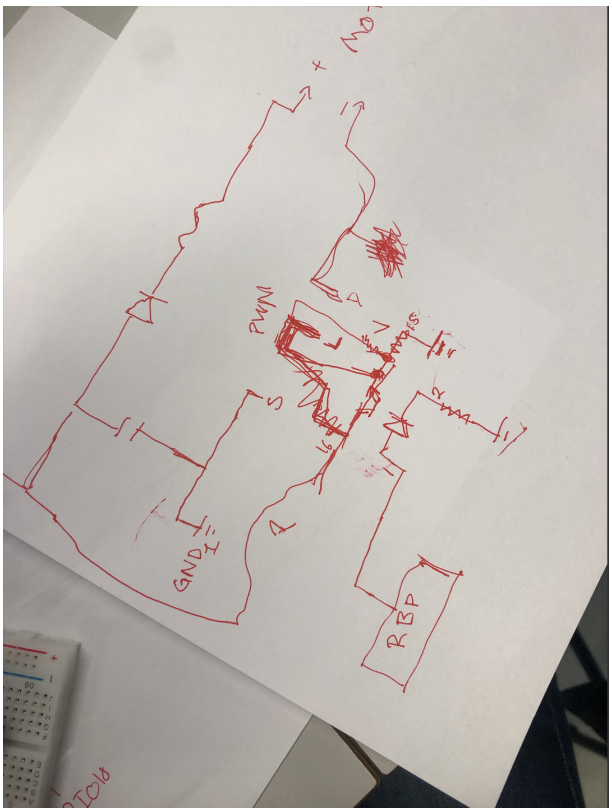

Figure 16: Super confusing schematic that made zero sense

6.6 27/09/2020 - Lab 6 continued part 2

Didn't have time to work on the Lab yesterday so had plans with my lab partner Julien to work on the lab today but ended up not hearing anything from him all day. So I decided to work on it at home because of Covid (don't want to go up to school unless I have to because of the pandemic). I used series connection of two 1.5 V batteries and one 9 V battery to power up the stepper motor (since I don't have a voltage supply box at home). I ended up trying to complete the lab to the best of my ability. At the end of the day after everything was done my accelerometer started smoking and I'm pretty sure it is destroyed.

Then finally around dinner time Julien contacts me over discord and suggests meeting up at school after dinner or early tomorrow morning before the lab is due (which is way too close to the due time for my comfort). But seeing as it was already finished (hadn't heard from him all day and didn't want to save this for the last minute) it was already all done and there was nothing left for him to do. He also wasn't involved enough in the Lab to even know what needed to be done which has been a constant in all the past Labs and I'm just so fed up with it.

7 Week 7

7.1 My lab partner was shit and here's why

I am so happy to be switching lab partners this week. Mine did not pull his weight for the past 6 weeks and I just could not have handled any more of it. Here are a few reasons why I was not satisfied with Julien as a partner:

- Did not seem prepared for the labs
- Did not understand how to use GitHub correctly (in week 4)
- Tried to engage him multiple times and it worked during the lab time (until 5 o'clock) and then he usually "had" to take a road trip with his friends or go swimming or basically do anything but help me with the lab
- Often left alone to finish the Labs by myself unable to get him to do his parts, did all the lab reports by myself and most of the coding was me, especially in the first few labs
- I was unable to finish the labs up to my standard because I was doing the work meant for two people alone
- This affected the quality of my work and my mental health, was super stressed very often because of the workload put on me and the time constraints and lack of help from the person who is supposed to be working on this with me
- Julien wasn't in the labs enough to be aware of what needs to be done
- I can usually work with most people but they need to meet me halfway, I should not be the one to tell a fully grown man to do his fair share of work, it should be common sense at this point in our education
- Would have liked to attempt more tasks in each lab because this was all super interesting but was unable to because I had a very unhelpful lab partner that contributed more than "yes" and "great"
- The programming is at least 80% mine because I DID MOST OF THE WORK
- He often said "clever girl" in a way that made it seem like he was surprised that I could do all this even though most of the lab work was done by me, especially in the beginning
- Constant mental breakdowns because of all of this
- There was a constant lack of communication and I expect more from someone I'm working with. I mean it's not like I'm asking for the moon, just for a reasonable amount of work so it doesn't all fall on me

7.2 30/09/2020 - IA7: Crate Up + Group meeting

Enums do not fit in with this assignment at all! There was no reason for them because it didn't make the code any more efficient than before. Worked with Aisha and Gunnlaug on it and using enums in this did not make a lick of sense. But I implemented it anyways, as useless as it was. Then we tossed our GPS device out the window (as one does) because it doesn't read anything from inside the building due to the concrete. Then I created a crate for reading these values and published it like 8 times (lots of changes and always a new version needed for each publish).

Group meeting about project to go over all the general ideas and hardware. We the group (me, Aisha, Gunnlaug, and Johann) met up at school to discuss possible final projects and what supplies we would need for each of these projects. This is when we decided to go for the mind controlled car and in order to sell it we want to make it a braking system of sorts to reduce delay time in reacting to an accident.

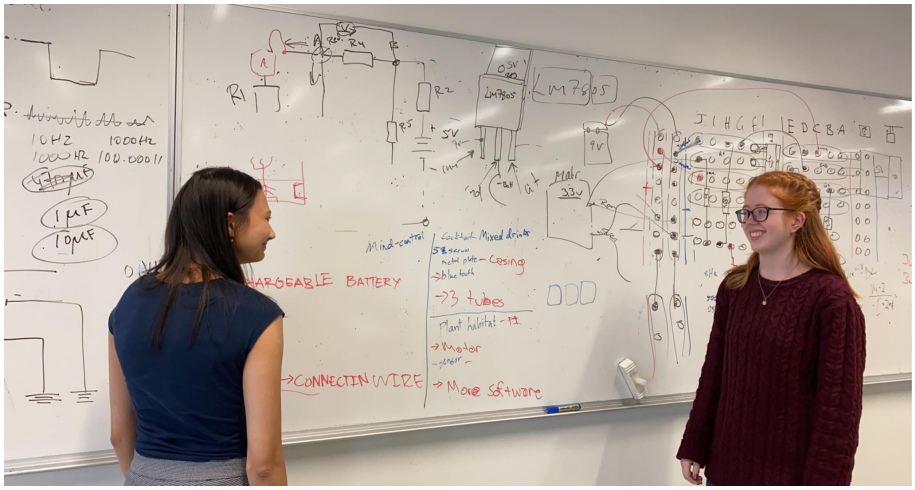

Figure 17: Aisha, Gunnlaug, Johann, and I figuring out what supplies we need

7.3 01/10/2020 - Lab 7: Finding the R.T.G

We had lots of fun throwing the GPS out the window (we as in me, Aisha, Gunnlaug, and Johann, my new lab group!). It sticks to other GPS's and is not the most reliable.

We ended up going outside to get readings since as stated before the GPS doesn't read anything inside Reykjavik University. It was super cold and we were not ready for that at all!

Figure 18: Aisha, Gunnlaug, and I freezing our butts off

We used Google Maps (did not have to pay despite what everyone thought, I have experience using the Google Maps API) and we made use of python and javascript (perks of studying Software Engineering).

We spent a lot of time outside because of the GPS, I'm really starting to hate it. It was super cold and we were not prepared for it at all. Towards the end of the day we were trying and trying to find the right coordinates, walking around in the dark like idiots holding the computer and breadboard with the rpi in a human chain.

Johann and I also implemented Google Maps for the Lab (which we did not have to pay for despite what everybody was saying, I've been working with the Google Maps API all summer for an app I'm developing). It was easy enough to implement in the end.

Figure 19: Us in front of Nautholl after following the corrected GPS

I got the software feedback system working but things were still not working correctly. Our program was telling us to walk 3 km to get to a destination (which we were not ready to do). In the end we decided to offset the values by a bit to get to the correct location.

7.4 02/10/2020 - Lab 7 continued

We (as in me, Gunnlaug, Aisha, and Johann) met again today with the goal to finish up the lab.

We were having current problems in our circuit with the LEDs that were supposed to be indicating North, South, East, West (LEDs were likely drawing too much current from the raspberry pi but we aren't sure and neither was Oskar). We asked Oskar for help and Aisha and Gunnlaug worked with him to fix this. Since LEDs are light emitting diodes and forward biased they switched two of them to make them reverse biased so that only North or South could be on and only East or West could be on (sort of like a switch of sorts).

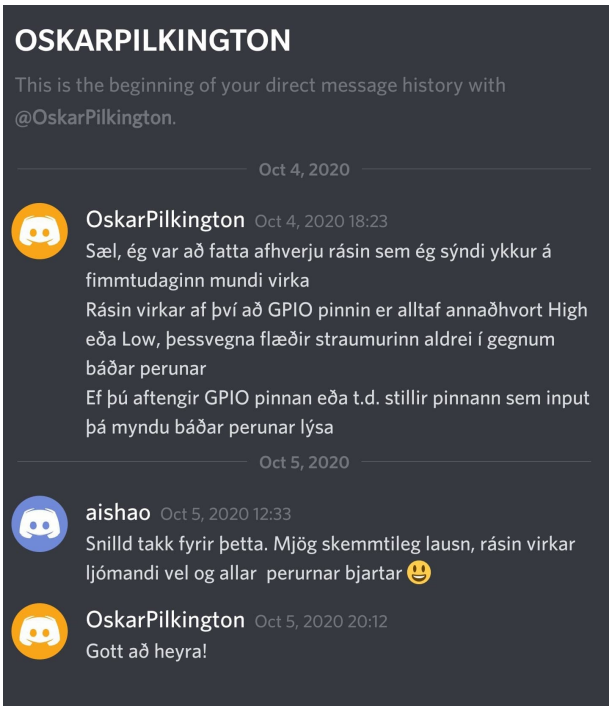

Figure 20: Oskars help with Lab 7

I was working on being able to draw the path we took on Google Maps so that we could finish up the lab. Managed to get it working but it was too late to do anything more in the lab so we decided to finish over the weekend.

7.5 04/10/2020 - Lab 7 continued

I was pretty busy with a Machine Learning Project so I was up at school working on that. I was however able to work with Gunnlaug and Aisha to complete the Lab. There was javascript in our code and since neither Aisha nor Gunnlaug have experience in this area I stepped in every now and then to help out. I also assisted in programming an Arduino GPS substitute so we wouldn't have to always go out to get readings when testing.

Aisha got a zero division error because the min and max values weren't always changing and they were subtracted from one another and another number was divided by them resulting in a null division error as seen in Figure 39.

Figure 21: Null division error in Lab 7

8 Week 8

8.1 05/10/2020 - OLED Dot-Matrix Display and more lecture

OLED screens are cheaper to make and energy efficient. It also has small pixels. This is all the things that make them so popular, especially when making small displays on small robots like we might be doing for our final project. We can plot points and send characters to it. We have the OLED from Adafruit in our kit.

Use the "SSD1306" library if using rust which I don't think we will since python is just so much easier. Use the raspberry pi though because it has more storage space, processing power, and better ability to handle complicated data according to Foley.

C++ can be confusing to use and don't I know it. In software engineering we have had quite a few projects where we had to use C++ despite never being taught it formally. The book goes through the process of using C++ with an OLED screen so it's not as bad as Foley makes it sound. We'll still be avoiding both of these complex languages and using python.

With regards to integrating physical systems with the internet the main goal in Mech 1 is to get things to work. Security and reliability come later. 4 modes to consider:

- **RPi as a web server** - The rpi acts as a web server
- **RPi as a web client** - The rpi acts as a web client, similar to what we have to do in Lab 8
- **RPi TCP/IP client/server communication** - The rpi acts as a web server and a web client (we had to program this sort of thing in computer networking class)
- **RPi web sensor using a platform as a service**

The raspberry pi is an IoT (Internet of Things) sensor which makes sense because it can be used with regards to the internet.

8.2 07/10/2020 - IA 8: Ground Control to Major Tom

The Adafruit dashboard reminded me of a simpler version of Datadog, a lot easier to use because it doesn't need to be connected to a sql database or anything like that.

Triggers were difficult though, adafruit wouldn't allow me to change the triggers and there was a problem making two triggers at once and was just super confusing. I figured it out in the end but shit that was difficult.

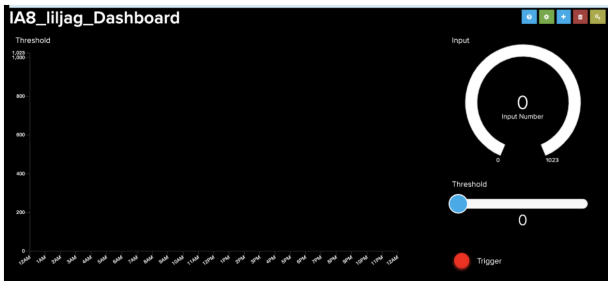

Figure 22: My Adafruit dashboard

8.3 08/10/2020 - Lab 8: Rebooting the Main CPU

Finished all the main tasks up at school with Aisha and Gunnlaug, did most of the programming because this is stuff I know. I used BeautifulSoup which I'm familiar with and have worked with often to scrape information off the weather website. Finished formatting that and then we went home. We worked until advanced task 1. We also simulated the weather data to more accurately test things.

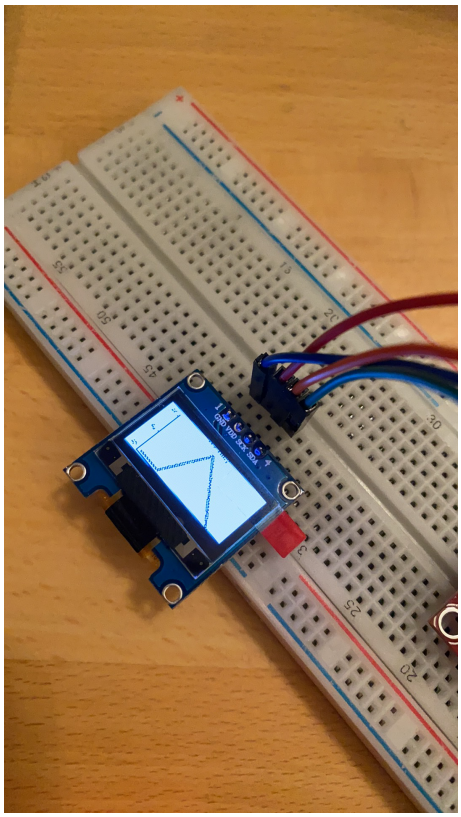

Figure 23: OLED screen with a simple graph on it

8.4 09/10/2020 - Lab 8: Rebooting the Main CPU

We (Aisha, Gunnlaug, Johann, and I) didn't use rust for this lab, instead we used python and web scraping for this. We used BeautifulSoup to scrape a weather website (it returns an easy to use Json output which I'm rather familiar with). Then Aisha, Gunnlaug and I went to Aisha's place and finished up the lab there. We finished advanced task 1 and 2 and it was a bit of a pain but it worked in the end.

Figure 24: Testing out characters on the OLED screen

9 Week 9

9.1 12/10/2020 - Foley Meeting

Instead of a lecture Foley decided to meet with us team 1 (me, Aisha, and Gunnlaug) to talk about the final project and how it was going deciding what to do. We told him about our idea for a brain controlled car. He had a few concerns because ordering parts takes a long time in the current climate and an EEG headset is not a common thing to have. Luckily I had two on hand from a project I did in my third year of high school so that wasn't a problem. It was also the reason why this idea came to us, otherwise we never would have gone for it.

9.2 14/10/2020 - Group Meeting: Conceptual Generation

We met as a group (me, Aisha, Gunnlaug, and Johann) up at school to discuss our final projects and what exactly we want to put into our conceptual generation. We have already decided what our final project is (just need to make sure at least one of the headsets works, if not both, which would be best case scenario). We wrote it all up on the whiteboard.

Me, Gunnlaug, Aisha, and Aron finished IA9 first at school and then at Aisha's, having to share android phones because Gunnlaug and I have apple and there were no resources for how to do the lab on an iPhone which is not cool. The only reason for that is because all the TAs have android. I personally think that if a lab doesn't work for all students then there should be another option or something to compensate for that, even though this is only extra credit. Everybody should have an equal opportunity to complete it. Luckily I have great friends in this class who could loan me an android phone to complete the individual assignment.

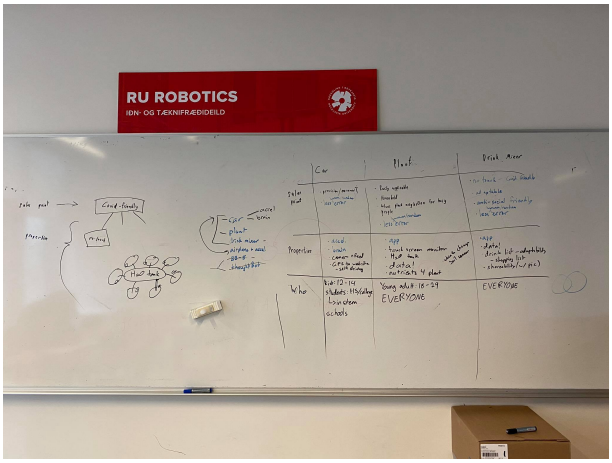

Figure 25: Our final concepts for the concept generation and some ideas for each

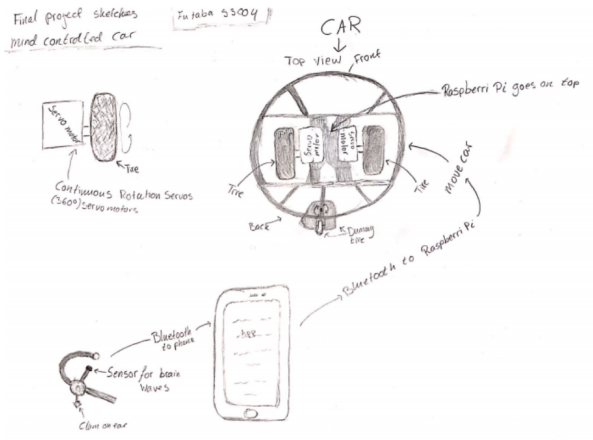

Figure 26: Our brain controlled car idea, drawn up by Gunnlaug

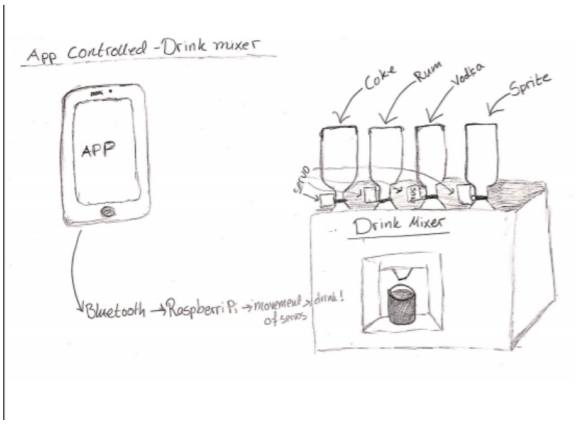

Figure 27: Our drink mixer idea, drawn up by Gunnlaug

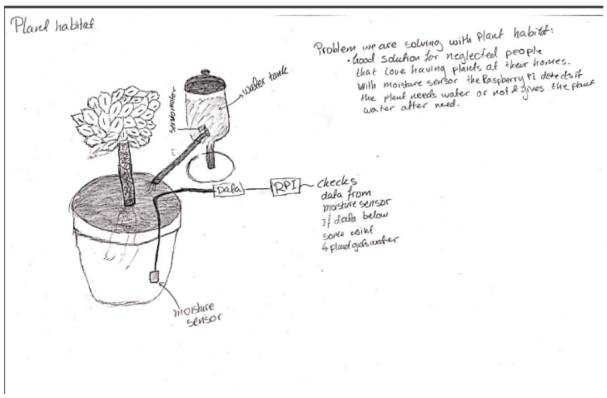

Figure 28: Our plant habitat idea, drawn up by Gunnlaug

9.3 16/10/2020 - Lab 9: Remote Control

I was up at school with Aisha and Gunnlaug to work on Lab 9. We had decided to only spend this Friday on the lab because there is just too much going on in my other classes at the moment and the same is true for them. Then we went to Aisha's to finish up. It was a big mix of different parts from past assignments so we got all the way to hard task 2 before calling it a night. Aisha and Gunnlaug were thinking of finishing up this weekend but I can't help because of the heavy workload I'm dealing with at the moment.

10 Week 10

10.1 19/10/2020 - Concept Generation

We (Aisha, Gunnlaug, Johann and I) met up at school to test the headsets. We also wanted to finish up the concept generation. We also got a base for the car from Hannes to try and test it out. Nothing much else happened today because I had to go work on a TSAM (Computer Networking) project but we agreed to meet later on in the week.

10.2 21/10/2020 - Getting data

We (Aisha, Gunnlaug, and I) met up at Aisha's and decided to test out the different headsets and get it all to start working. We tested out the 14 port EEG headset from Emotiv, the Epoc+ and the single port EEG headset from Mindwave, the Mindwave Mobile 2 on Aisha, the 14 port one didn't get a good connection no matter how much Gunnlaug pressed on the connection pads to get a better connection. This makes sense since the headset is quite old and was a little bit rusted when I took it out of storage.

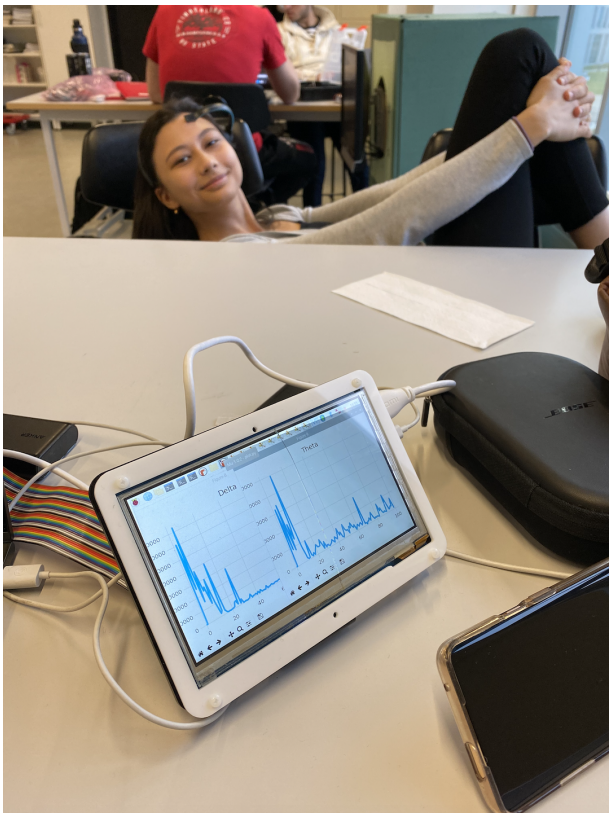

Figure 29: Aisha testing the headset

So we ended up using the single port headset on Aisha and got to work on connecting it to the raspberry pi. First we tried to create an Android app to connect to the raspberry pi and the Mindwave headset to see if we could get a connection going that way. We had already gotten the headset to connect to an app on the phone from Mindwave so we knew there was data coming from the headset.

After much struggling with creating an app and reading the documentation we figured that this was just never going to work. So we read the general

documentation a bit more carefully and after quite a bit of programming and trial and error we figured out how to connect the headset to the raspberry pi directly through python2 (because the headset is pretty damn old). By this time it's midnight and we're just getting started with looking at the data.

We decided to plot the data because looking at raw data is super difficult and never the way to go, especially since we are trying to figure out when a person is shocked since our project is going in the direction of lowering the delay time in braking when it comes to accidents (direct brain to brake connection, no middle man). This only took 3 hours but by the end of it all we had to show for it was a small line that went straight up and down because I had put the x axis as datetime which was too big a scale. Might be a better idea to just have integers from 0 to 100 (as seconds of sorts) but I'll program that tomorrow, right now I desperately need sleep.

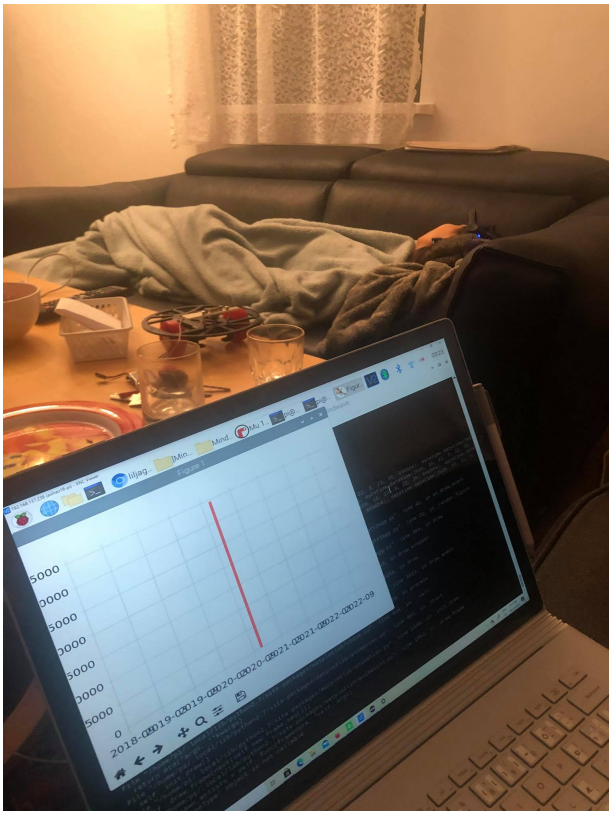

Figure 30: Aisha testing the headset

Earlier I taught Aisha and Gunnlaug about GitHub tricks and branches. This is to make sure that if there are multiple people programming then they will know how to have their own "workspaces" that don't affect everyone else every time they push to git and don't cause merge conflicts every time. They

did really well and learned this all pretty quickly which is great.

I also figured out a way to install better by using the command

sudo python3 -m pip install ....

This was a great way to install and make sure it installed on path and actually happened. Might come in handy in the future.

10.3 23/10/2020 - Fixing the plotting and connection stuff

I implemented the counting thing I talked about yesterday and we (Aisha, Gunnlaug, and I) tested it out and actually got some great graphs! Now we could start testing. We were up at school so we were working in masks which wasn't super fun but it worked. Now we took a few groups of data and after looking at graphs of alpha, beta, gamma, theta, and delta waves we found that beta and theta were the waves that represented a shocked state the best. We used Aisha as a test subject and we put her in a shocked state by having Gunnlaug come up from behind her and surprising her while I looked at the output and seeing when this happened to be able to locate it on the graph in the end.

This is a huuuuge breakthrough and really gives me hope that this project will actually work which is such a relief because this is pretty risky since we are using live data from the real world which is never easy to work with. If Machine Learning has taught me anything it's this.

Before we left I made a CSV file filled with data from all the brainwaves from a test run of Aisha so that we could have a reference from which to build and test out base from (and servos form) without having to run the headset.

Few things to have in mind regarding the Mindwave headset and connecting it to the raspberry pi:

- Establish spp connection on rpi by using lab9 bluetooth via rfcomm pairing instructions at beginning
- Pair the rpi with the headset using spp connection with command: "sudo bluetoothctl" and follow connection directions on canvas for rfcomm
- Follow link instructables up to a certain extent
- Install git repo sudo git clone https://github.com/cttoronto/python-mindwave-mobile
- (DO THIS STUFF FIRST BEFORE BLUETOOTH STUFF)

  Update MindwaveMobileRawReader.py and input mac address 74:E5:. . . . of the headset

  Run read_mindwave_mobile.py
- Mindwave to pi is serial connection
- Pi to servos is I2C

10.4 25/10/2020 - Making the car

We as a group (me, Aisha, and Gunnlaug) decided to design our own car for the project instead of using the car that Hannes provided with the metal plating. Aisha took a Lego plate from home,to make it all more modular because we weren't 100% sure how we were going to put everything together and want to be able to change things as we go along. By using legos the parts were also modular and could be taken apart and worked on one part at a time as everything was removable. We decided to have two servos that controlled the direction and one dummy tire for balance (See figure below to see design). By designing our own car the servos were replaced with two smaller continuous servos.

Car parts for hardware:

- Raspberry pi 4:Link to the product.
- Power bank: Link to the product.
- Bread board: Link to the product.
- Servos: Link to the product.
- Oled screen: Link to the product.

We attached two servos to the plate to complete the base that we were going to build everything off of. Then we put the raspberry pi and power bank on top and attached it all together. After testing quite a bit we found that the car wobbled quite a bit so we decided to add another dummy tire between the two servos for extra support.

Using pulse width modulation to control the servos, Aisha discovered the continuous servos worked differently than the 180 servos from our kit and even worked differently from one another. To see how the servos were different a while loop was placed from 1 to 20 to duty cycle, these where the results:

- Duty cycle 1-3 was the highest speed in a clockwise direction
- Every duty cycle after 4 decreases in speed in a clockwise direction
- Duty cycle 7 stops both servos
- Duty cycle over 7 increases in speed in a counter clockwise direction
- Duty cycle 12 and over gives the top speed in a counter clockwise direction

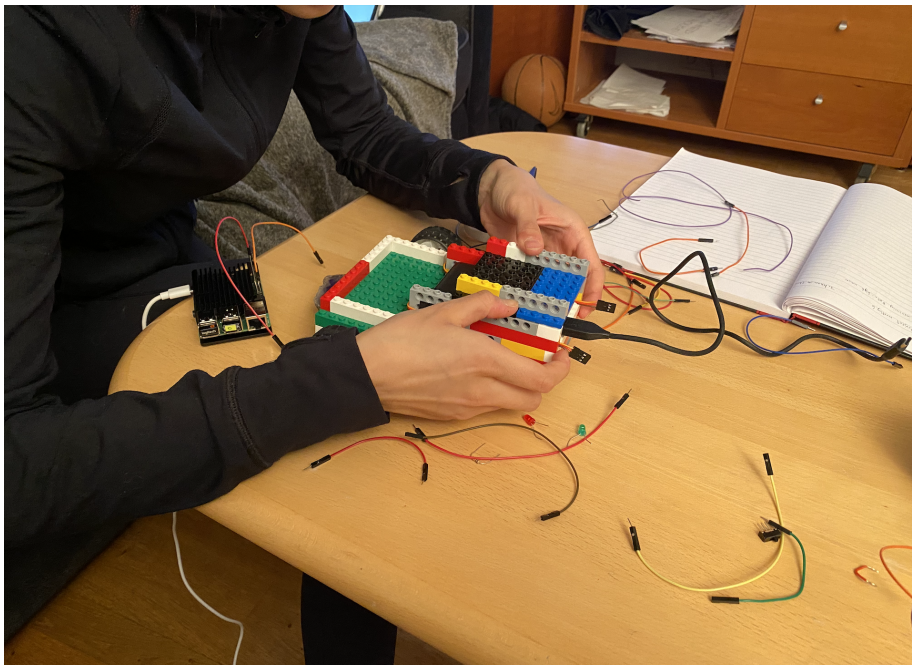

Figure 31: Aisha building the car

10.5 26/10/2020 - Programming and testing the car

While testing the car, one of the tires still proved to be wobbly. We decided to take apart the servo in hopes of making it more sturdy. Gunnlaug took apart the servo after further analysis and Aisha removed the gears. We saw that the pin was loose, so I super glued the pin to the servo to make it more sturdy (if duct tape can't be used then super glue is the way to go). Aisha proceeded to insert the gears back in and Gunnlaug closed the servo. The tire was more sturdy and the car stopped wobbling. The servos operated at different speeds and required more testing to be able to operate the tires together.

After playing around with the car three sets of speed determined by the duty cycle were determined:

- Fastest speed: Servo 1 = duty cycle 4; Servo 2 = duty cycle 13;
- 2nd fastest state: Servo 1 = duty cycle 4.5; Servo 2 = duty cycle 10;
- Slowest state: Servo 1 = duty cycle 5; Servo 2 = duty cycle 9.5;

To avoid crashing the car into other objects Gunnlaug suggested implementing some type of sensor, the group discussed implementing a camera or infrared sensor. I suggested simply limiting the range the car could drive so it could be placed in a location without bumping into objects within a certain range. This way there would be less work for us and we wouldn't have to run around looking for extra sensors. The final decision was to program the car to drive in a certain radius by programming one tire to go faster than the other. To implement the car into the project Gunnlaug and Aisha programmed a state machine that would receive signals from the program that I created to receive data from the Mindwave headset. The state would be determined based on the frequency of the wave.

11 Week 11

11.1 27/10/2020 - Car working!

We (Aisha, Gunnlaug, and I) started up at school recording more data by shocking Aisha while she was wearing the headset and doing all sorts of tasks like pretending to drive and talking and moving her head around. This was to try and eliminate any sort of noise. What would have been idea was to have more test subjects for a wider range of data in order to get a threshold that works for a wider range of people. But what is important is that it all works so we can keep on going.

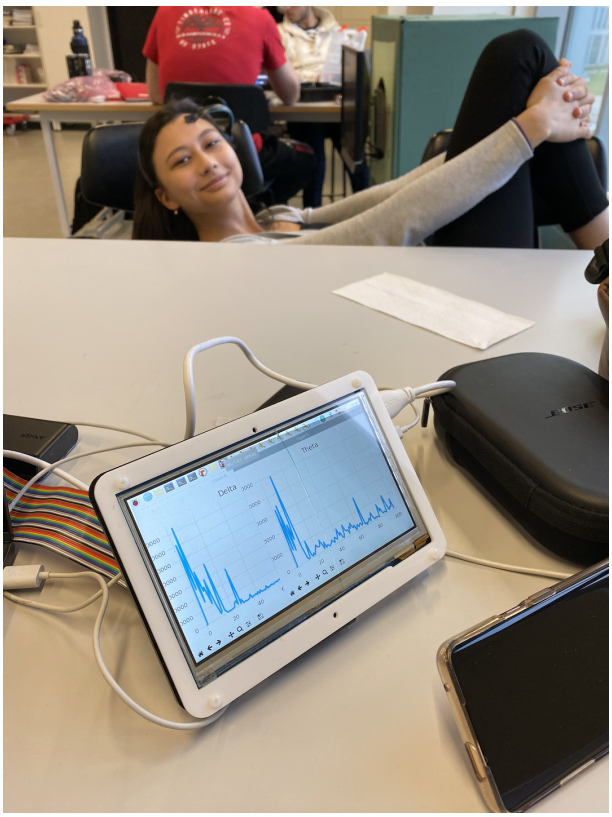

Figure 32: Aisha testing the headset

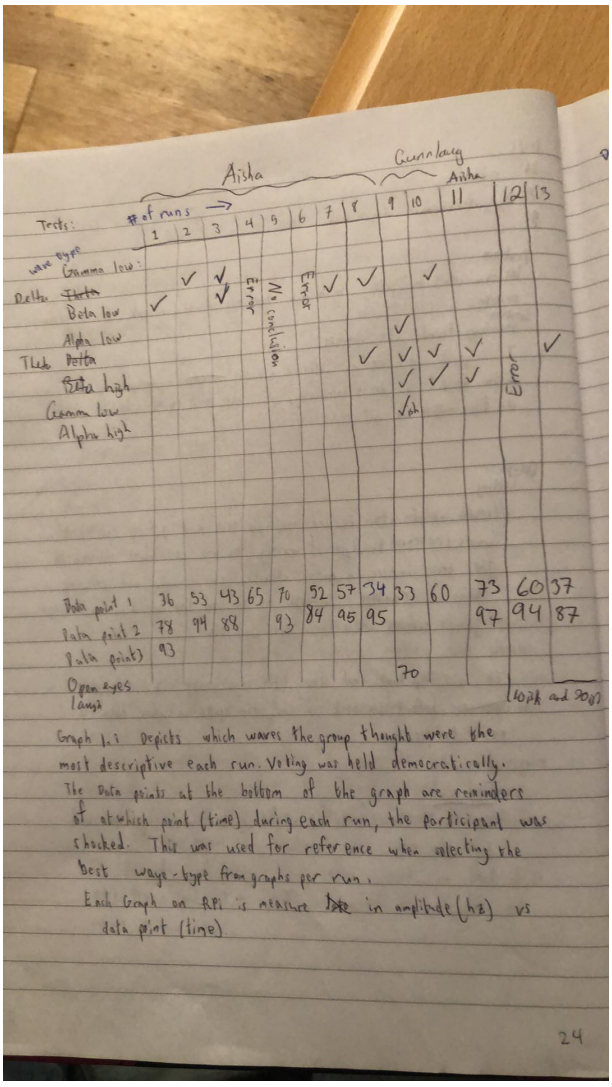

Figure 33: Johann tracked the whole thing in his notebook

11.2 28/10/2020 - Disco Time

We (Gunnlaug, Aisha, Johann, and I) went to Íhlutir to try and find some more parts for the car like a smaller breadboard and some fun lights. We didn't find anything so we went to my house and got a smaller breadboard I had there (almost destroying it in the process but it all worked out in the end). Then we added some lights and an OLED screen. Gunnlaug made the lights be all disco like and Aisha played around with the OLED screen and both wanted to have their thing on it but things started going wrong towards the end, probably too much going on at once.

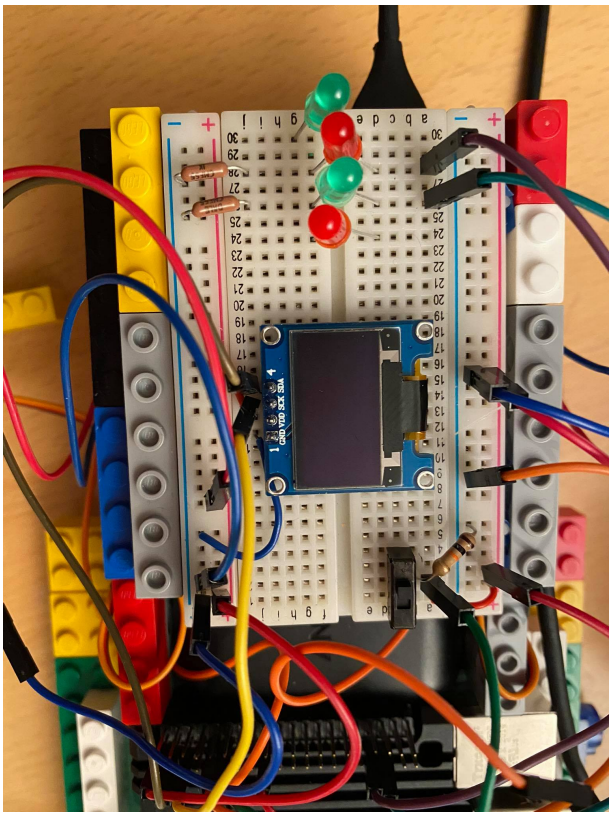

Figure 34: Lights and OLED screen added to the car

11.3 30/10/2020 - Things go wrong and then right

Gunnlaug took disco off, needed to fix the wiring because servos were blegh, connected one wire at a time. We didn't know what was taking too much energy, but the battery was not enough in the Mindwave headset and that was the thing partially causing the problem. Once we put a new battery in and took a few things off of the breadboard everything started to look ok.

Then we recorded videos, implemented power switch, OLED worked for a bit but then went kaput. Ended up taking it off temporarily. Recorded more videos, made mind map of commercial to keep us organized and to fully realize what we wanted in the commercial and what we needed to record to complete it. We went to Smaralind to record for the video, almost finished it, just a few more things left to do. Went home way too late.

Overall the day started off bad but ended well, Mindwave battery was "dead" and the servos were ticking but everything turned out alright. The wheel also fell off the car late at night but we fixed it. I edited the video and made everything look nice.

Figure 35: Gunnlaug scaring Aisha to get more data

12 Week 12

12.1 02/11/2020 - Final Project: Short video finished

Gunnlaug, Aisha, and I worked on the video and the car and added some final touches and added a few more videos to the short video to make sure everything was looking great. Editing videos takes forever! But everything works 100% now which is such a relief and we most likely don't have to record any new functionality videos for the longer presentation video. Watched scary videos to test things out and record some more just in case and took up video of putting car together and put it into the short video.

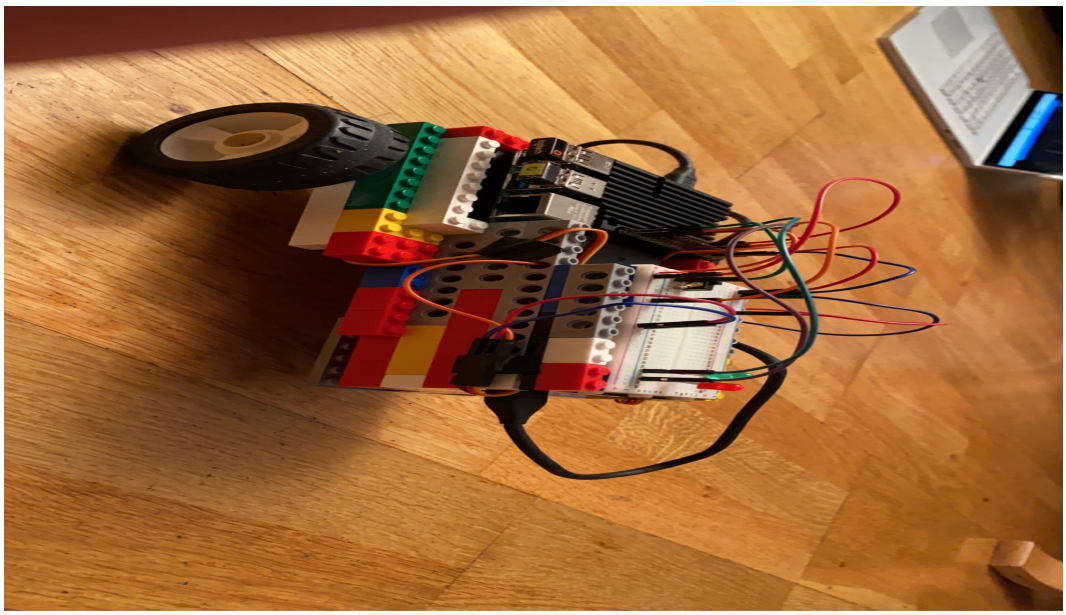

Figure 36: Final car used for the shorter video

12.2 03/11/2020 - Final Project: Starting the final presentation video

We replaced the OLED screen with a 5 inch HDMI screen to better see the frequency. Then we tested the car on Johann and Gunnlaug. Then Johann left but the entire time that he was here the atmosphere was super awkward and standoffish because he had been so little involved in the project as a whole and didn't know what was going on or what our final project really was but kept on trying to assert that his way of thinking was the right way when he had no idea what was going on. But we got way more done after he left and we finished the entire storyboard and script for the final presentation video.

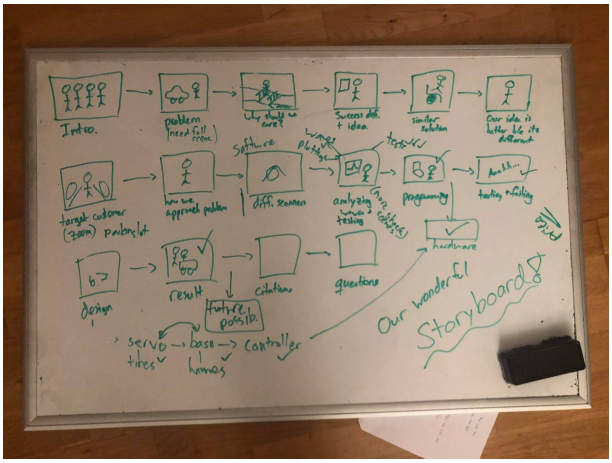

Figure 37: Storyboard for our final video, super organized

12.3 05/11/2020 - Final Project: Finishing up and just so done with this

We record EVERYTHING today, didn't sleep one bit. Luckily it was only me and Johann talking and then I had to edit everything while Aisha and Gunnlaug were editing the presentation slides which we added in afterwards. We checked with Foley whether we could get creative and he said yes so long as all the information was there and so we did that

Figure 38: Aisha with a prop used in our final presentation video

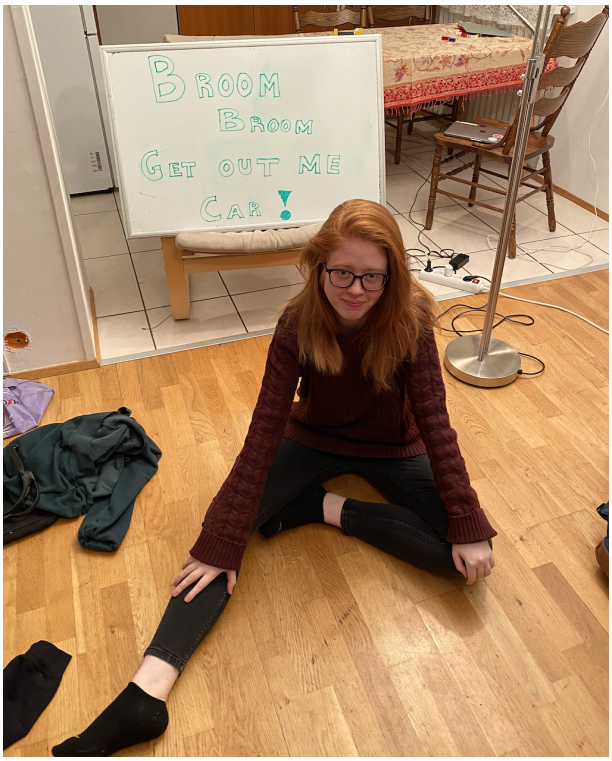

Figure 39: Me super tired and just done with filming but really only halfway done at midnight

12.4 06/11/2020 - Final Project: No more, all done

We have not slept at all, technically still the 5th of November for us because we spent all night recording and editing the final presentation video. Everything is sorta woozy and I really just want to go to bed.