Reykjavík University

Mechatronics I

T-411-MECH

Laboratory 6

27/09/2020

Students:
Giacomo Menchi
Silja Björk Axelsdóttir

Teacher:
Joseph T. Foley

Contents

1 Main Tasks

As always, this is the GitHub link with all the code inside.

NOTE: our url shortener doesn't work sometimes, so we reverted back to normal links which go outside of the page boundaries in some occasions. We are sorry about that but we haven't figured out another solution yet (the links are still clickable if the pdf is downloaded, so this should work fine):

`https://github.com/ru-engineering/lab-6-group-10`

1.1 Main Task 1

Task: Create a circuit to be able to supply power safely to the motors.

- Have the power input be reverse protected
- Have the power input filter out spikes
- Have a fuse in the circuit

Pictures: Below are the picture and the schematic of the circuit described in this task.

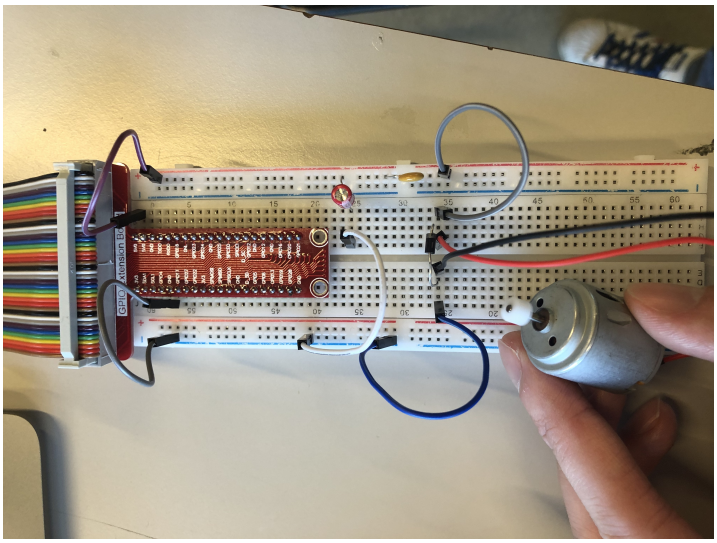

Figure 1: Circuit to apply power safely to motors.

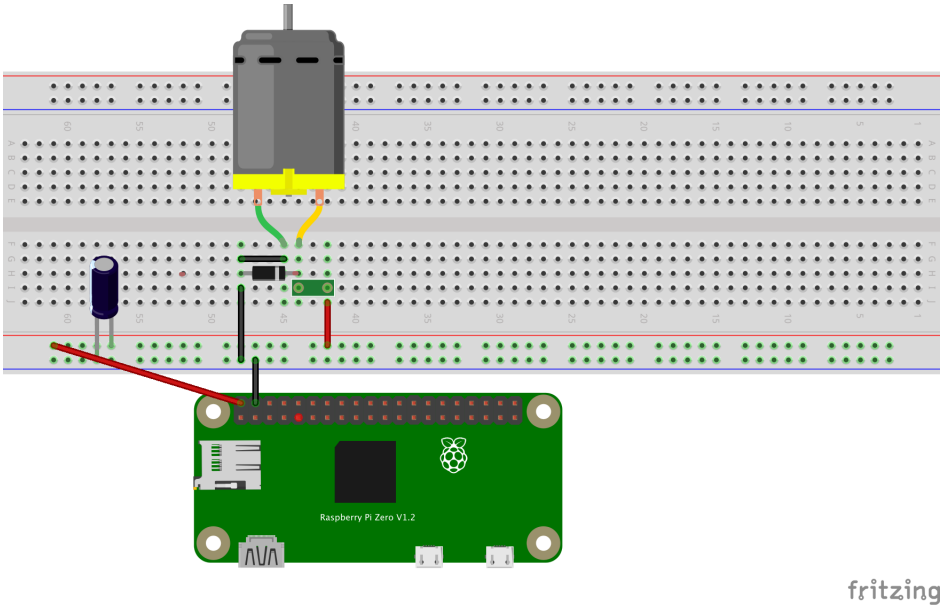

Figure 2: Schematic of the circuit in figure 1.

1.2 Main Task 2

Task: Create a circuit for the DC motor so the motor can drive with variable speed, show it working.

Also, measure the noise on the motor and signal line and make sure that it is not too much for the RPi.

HINT: Use a PWM signal to control the speed.

HINT2: Use a transistor.

NOTE: Use optocouplers to reduce noise from the motors and protect the RPi.

Pictures: Picture and a schematic of the circuit along with an oscilloscope picture of noise on signal and power lines.

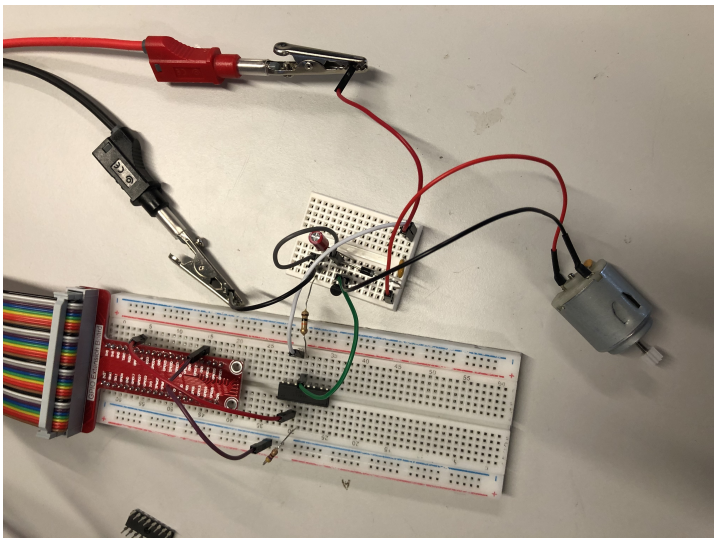

Figure 3: PWM circuit for DC motor with all the components.

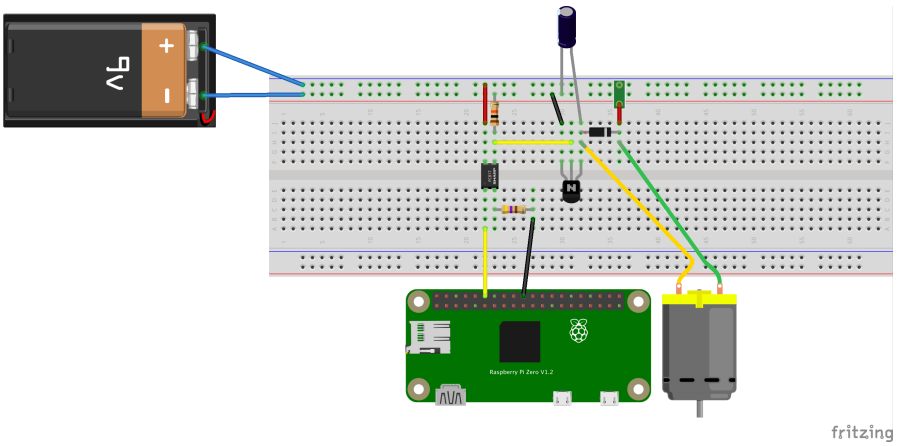

Figure 4: Schematic of the circuit in figure 3.

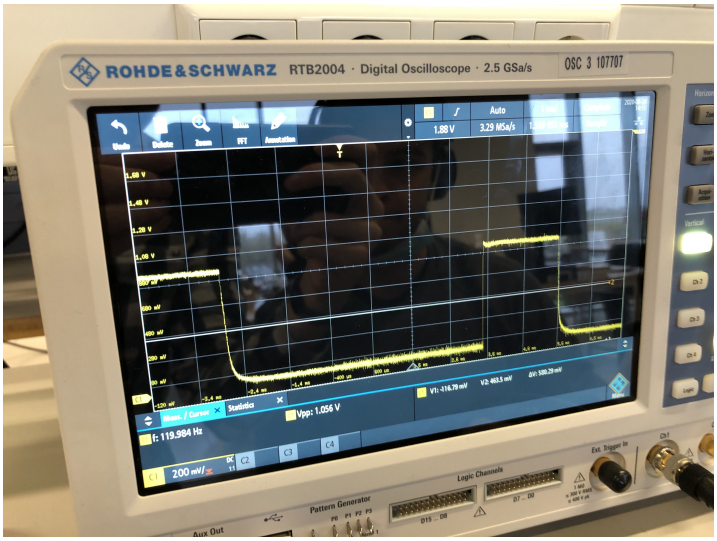

Figure 5: Oscilloscope measurements of noise on signal.

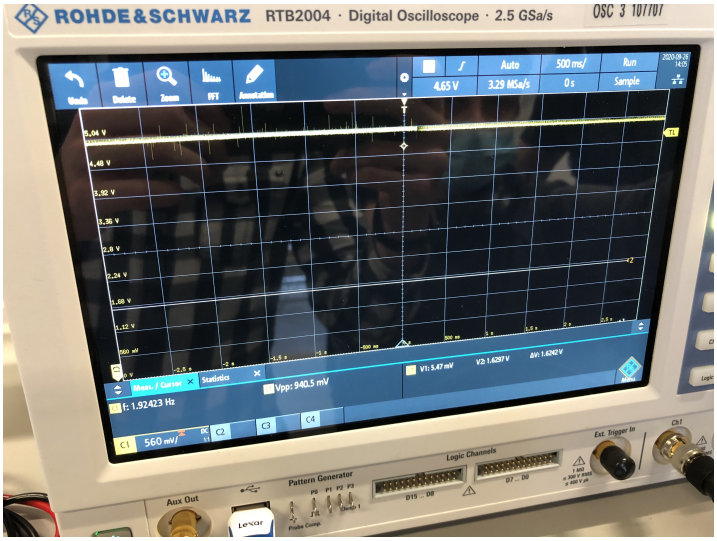

Figure 6: Oscilloscope measurements of noise on power lines.

Video: Video demo below.

`https://drive.google.com/file/d/19XH9sacvQFjzoQSl8LgYIc8I5f3QM8K7/view?usp=sha`

1.3 Main Task 3

Task: Create a circuit and code for the stepper motor to be able to go both directions and track where it supposedly is.

Video: Video demo below.

`https://drive.google.com/file/d/1UYr9nPY3JeDevOiOAnZCNJ4M3b95ReJc/view?usp=sha`

Pictures: Picture and a schematic of the circuit along with an oscilloscope picture of noise on signal and power lines.

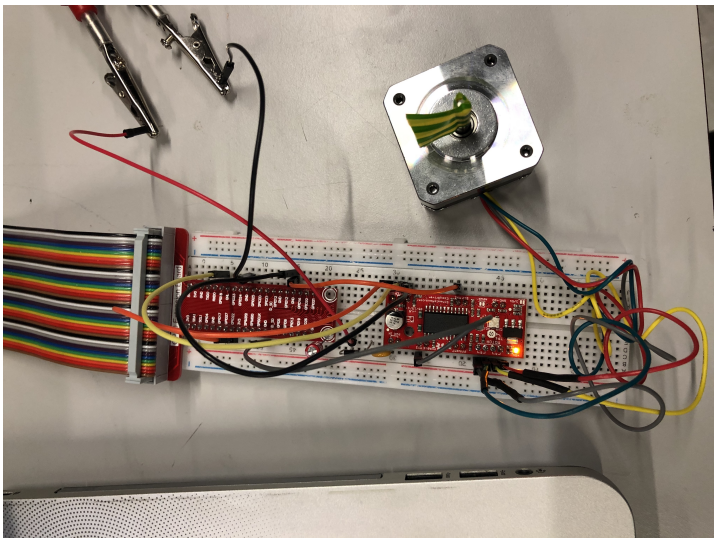

Figure 7: Circuit for stepper motor to be able to go both directions.

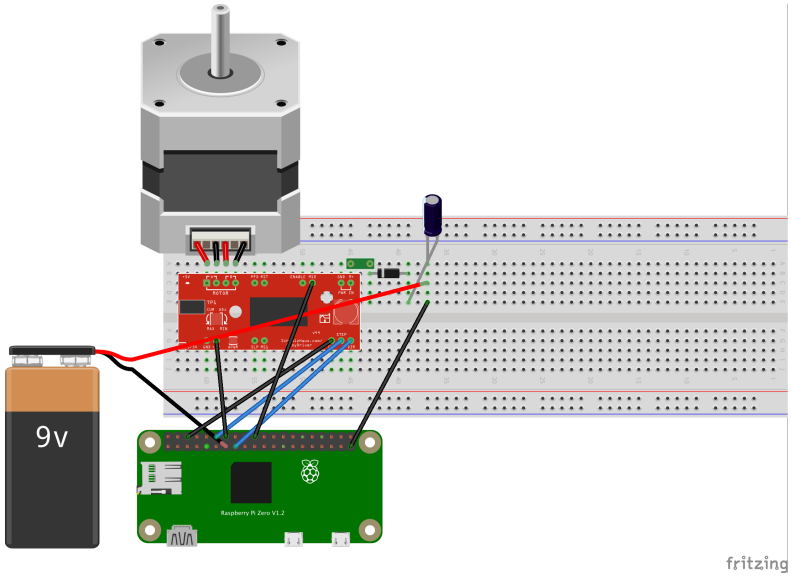

Figure 8: Schematic of the circuit in figure 7.

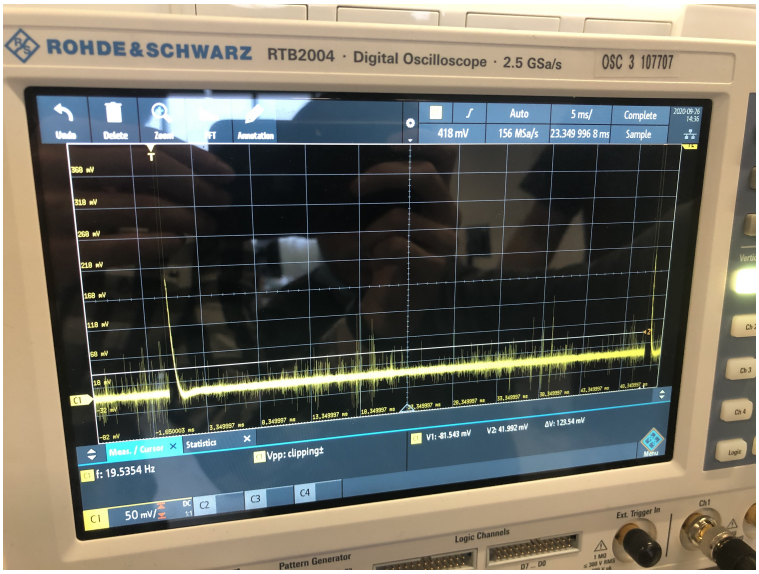

Figure 9: Oscilloscope measurements of noise on signal.

Figure 10: Oscilloscope measurements of noise on power lines.

1.4 Main Task 4

Task: Connect 2 potentiometers to an Arduino and have that Arduino send the data to the RPi.

Pictures: Picture and a schematic of the circuit as well as the reading of the potentiometers sent from the Arduino.

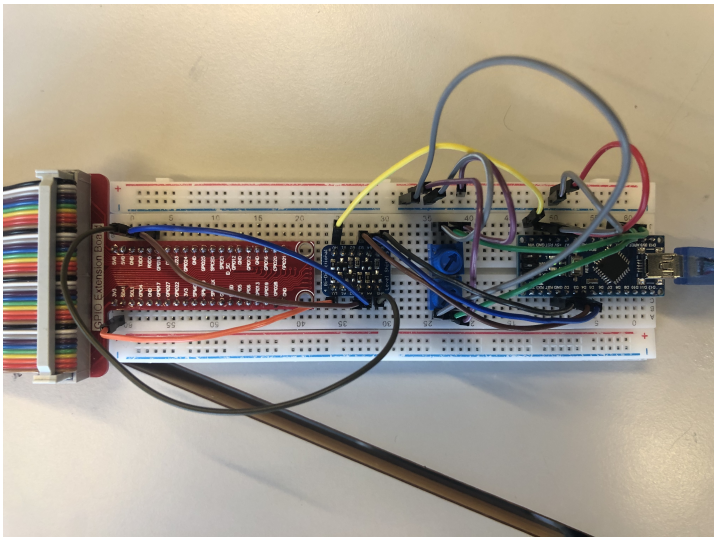

Figure 11: Circuit of the two potentiometers connected to an Arduino.

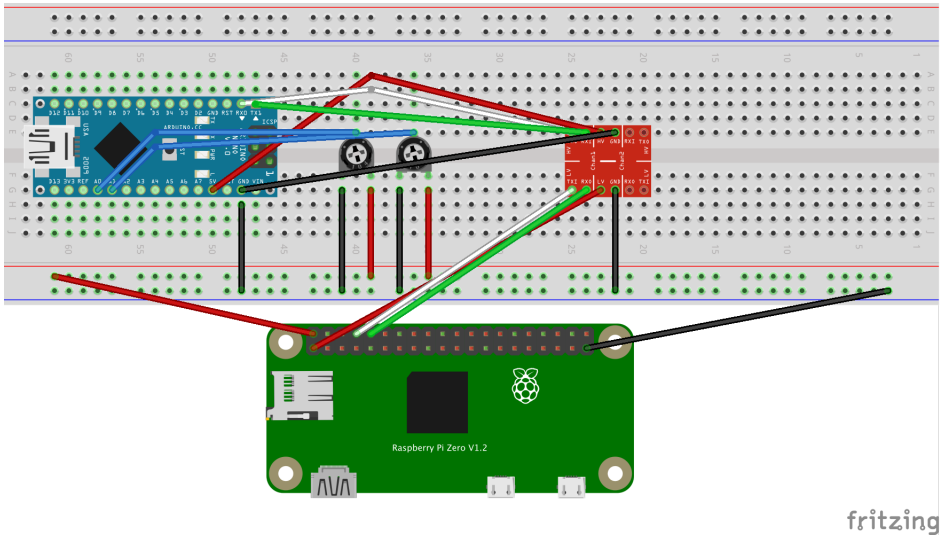

Figure 12: Schematic of the circuit in figure 11.

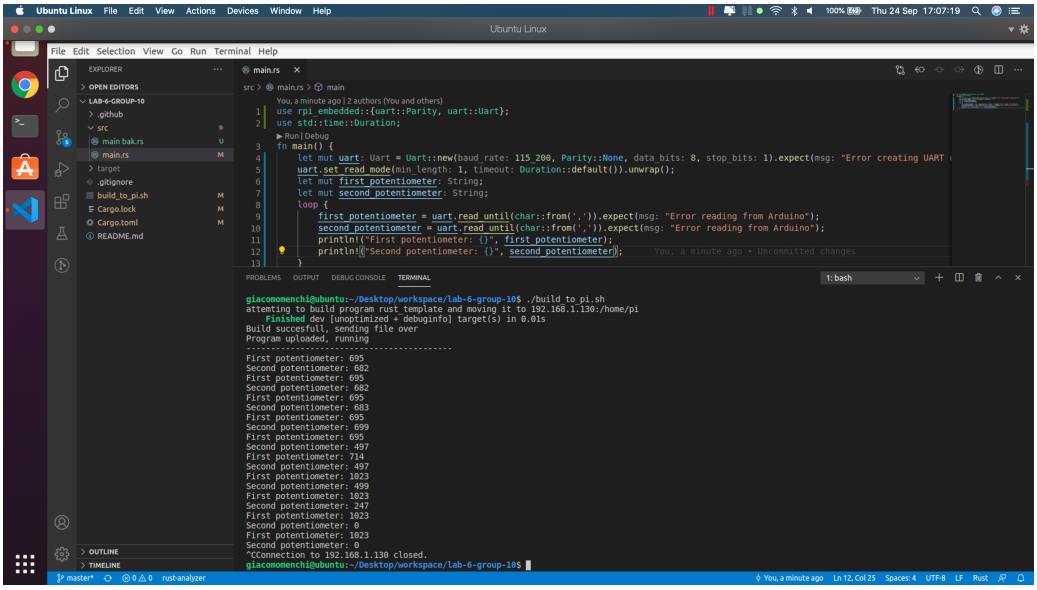

Figure 13: Readings of the two potentiometers.

1.5 Main Task 5

Task: Have the data from the first potentiometer control the current position of the stepper motor and the second one control the speed of the DC motor.

Video: Video demo below.

`https://drive.google.com/file/d/1iZ0ZcfnSdLv5MF9faxIa9ADZkQ3GIG_w/view?usp=sha`

2 Hard Tasks

2.1 Hard Task 1

Task: Create a small arm and connect it to the stepper motor.
NOTE: nothing fancy is required, cardboard and foam-core work for this

Video: Pretty self explanatory, the video is in the link below.

`https://drive.google.com/file/d/1kt8x1zcUEbM-C26Eh03Gv9AoAlFbvl86/view?usp=sha`

2.2 Hard Task 2

Task: Make sure the stepper motor goes to the same position between power cycles.

HINT: have a switch be the zero position for the motor or overdrive the stepper

Video: The video shows how the stepper motor keeps the "origin" between code executions. This was obtained by saving the last position value to a file inside the Raspberry Pi called "position" each time it is read by the Arduino: this value is a number which is positive if the current position is right (clockwise) of the origin, or negative otherwise. If the program is interrupted at any moment, the next execution will revert the stepper back to the origin, which is obtained by just calculating the difference between the current, saved position and the origin itself.

https://drive.google.com/file/d/136hXUO7-cArWvXlSrUqMgbNyfvkOfn_C/view?usp=sha

2.3 Hard Task 3

Task: Connect the dc motor to a string and have the project from Lab 4 (the grabber) dangling on the end of it. The Grabber must do something.

Video: The video below shows the grabber in action. Since the grabber we created in Lab4 was destroyed by mistake, we agreed with Danila to replace it with a simple magnet, which still kept the "Grabbing functionality", and connecting the DC motor to that instead. We know the grabber from this task and from the next one is not the best looking and best working ever, but it was the best we could create with the materials and time we had.

https://drive.google.com/file/d/1yYqpWSVVG696tPcHzqq14gwLTU7s7O0a/view?usp=sha

3 Advanced Tasks

3.1 Advanced Task 1

Task: Make it work as a crane, so the grabber works with the accelerometer, the crane can move up and down and can be hoisted up.

Video: The instructions were not so clear, but we think we got this right. In addition to the setup from the task before, we used the accelerometer movement to control the opening and the closing of the grabber "fork" and the servo to hoist up and down the entire apparatus. Again, this is not the best machine in the world, but it was the best we could build with our limited time and resources.

`https://drive.google.com/file/d/1cDy1dA0YBiQchODaB7-lqIdS0n9Ikneu/view?usp=sha`

3.2 Advanced Task 2

Task: Control the speed of the DC motor with a PID loop.

Explanation: We had a lot of trouble trying to complete this task. The PID loop, which schema we drew and is shown below, requires a controller that gives feedback about the motor behavior: this feedback is then used to correct the motor speed (if out of the correct range) in a new input cycle, thus better approximating the speed we wanted to obtain, and so getting a better and more precise output.

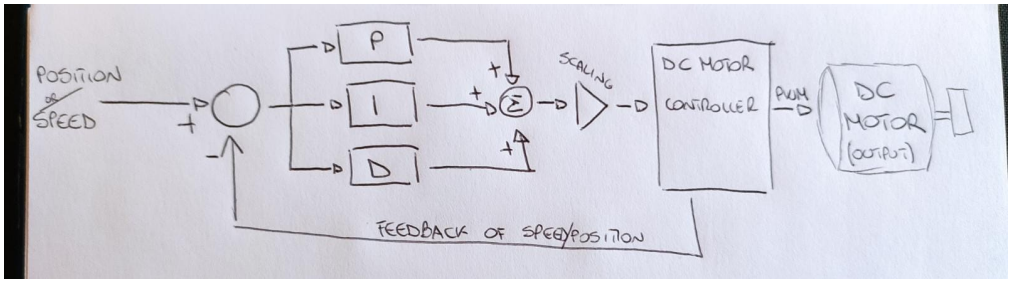

Figure 14: DC motor PID loop schema.

However, since we weren't able to obtain an hardware PID controller anywhere (neither to implement it in any other way) we didn't manage to complete this task in time for the assignment delivery date.