PROCESS DOCUMENT

Project Self-balancing App-controlled Robot

Abstract

This document includes a description on tips and tops of the process of this project.

Konghon Choo and Eveline Ververgaert

Year 2 Special Topic

Table of content

Assignment

For the paper Special Topic we have chosen to build a self-balancing robot. It consists out of two parts:

Robot

The robot is the main part of the project. If it is not doing anything at all it should be able to stay balanced on 2 wheels. It should also be aware of obstacles like walls. And the last function is having a Bluetooth module to be able to communicate with the app.

App

The app is able to control the robot when connected over Bluetooth. When Bluetooth is connected messages can be send to the robot from the Android app to make it move the direction and speed of your request. In return the robot can send the app messages, e.g. battery status.

View Requirements documentation for more on functionality.

Way of working

We worked partly with V-model (starting with documentation, then implementing the code and building the robot itself) instead of full agile, because the project is simply too small to update the documentation constantly as well. But the iterations we did work with were mostly for the implementing phase best, because this way we constantly had something, no matter how small, to show for. The documentation that has been made has all been reviewed by other team member when considered finished.

Result

Currently the result is the robot can indeed balance by itself for a few seconds. However due to probably a near empty battery and imperfect structure it is unable to stay balanced. With some more finetuning like lengthening the chassis and making it more top heavy you could make the robot easier to balance out.

The app is currently not working, because of a Kotlin update from JetBrains. However, the joystick to control the robot has been implemented. A start of the Bluetooth implementation has been made, unfortunately the Bluetooth LE module broke, and is therefore removed from code. (In the GitHub there is the neat version, with only working parts, and the versions including the Bluetooth code which has not been tested

Conclusion

Tips:

- Get the hardware ordered earlier.
- Don’t stay too long with unnecessary problems.
- Don’t forget to keep back-ups and/or to use GitHub more.
- Make better schedules and take this time the amount of people into account.

Tops:

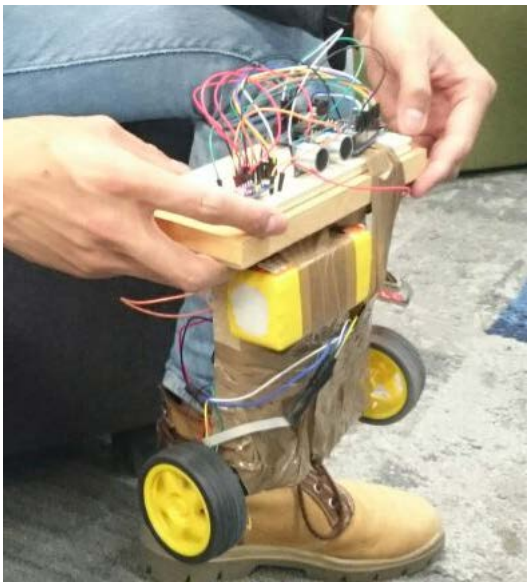
- Documentation was good.
- Recycling hardware scraps.
- Having an adorable robot.
- Research on Kotlin.
- Self-study / experimental subjects (learning new things without any lecturer and no experts around).

Self-reflections

Konghon

I was mostly focused on the hardware side, combining sensors and trying to fine tune everything. We based the robot initially on designs other people build. We had a basis for the hardware, from there we had to adjust a few things to fit our specific requirements. We had multiple iterations of the robot due to the available hardware at the given moment. The final robot parts only arrived at the last moment and that made it hard to finetune for it.

The final product is a cool robot, which shows the intended behavior with some finetuning it might run like a champ. The good thing about this project is that the staff at polytechnic were very supportive towards this project. When assistance was needed it was provided quick, communication was also very good when there were questions they would be answered quick. The initial phase went quick and good, the startup of the project went without any problems. The middle was a bit muddy, things got delayed and the progress was slow. The traditional sprint methodology was not kept in place and it was more like "do what ever there is to do" mindset. In this phase the code got written but never tested or finetuned. The final phase was the better part of the project in terms of progress. With the parts I've gotten from Vaughn I was able to build a first iteration. The first iteration was nothing more than the parts connected and working and not interfering each other. This worked, and it was then put onto a "frame", everything was taped together onto a piece of wood. This was known as the "tape everything" iteration.



This Enabled the us to see everything work together for the first time. The wheels were slipping and not meant to fit on to the motors, which caused a lot of wobble and flex in the frame. The tape mounting was also not ideal, the components kept shifting.

The next phase was to replace the frame and motors and wheels. The frame got build from old broken robot frames, the wheels and the motors arrived. The frame was fitted with the old motors at first and with the new wheels. The wheels came with an adapter but wouldn't fit the new motors. The shaft is 6mm and the adapters were made to fit 4mm motor shafts. So, the first test was done with the old motors with the new wheels. And finally, I was able to get to engineering to get the adapters drilled out to fit the new motors. Due to the size and weight of the motors the balance of the robot changed, and I was unable to balance it out again.

For the future iteration if I had time I would be do the following things:

- make the robot taller

- Add weight on top. (Top heavy is easier to balance “Inverted pendulum”)
- Make the tuning variables easier to change. And not having to upload new code when I need to finetune it.
- Make the chassis more durable (crash bumpers)
- Integrate the wireless module.

https://youtu.be/ctpfV8l90X4 Demo Clip

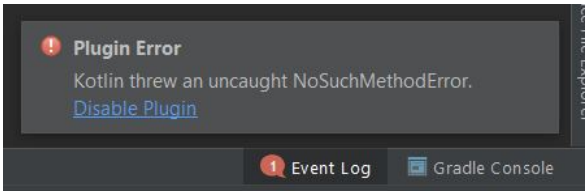
Eveline

Documentation

On the documentation part Konghon and I made the PID and Requirements document together. Then the UML design document we also made together, however it consists of mostly Konghon's parts because we decided to split the project into 2 (one the robot and one the app). Furthermore, I worked on the Kotlin research document and the complete UI design document. I believe it was important to do the design in this case, because of the Multimedia paper we entered this semester as well.

Overall I'm happy with all documentation, because it made the project really clear in my eyes. And I am also a fan of documentation because it prevents already some potential miscommunications and interpretations.

I like to add that the Kotlin research truly helped me. Even with some issues like not enough experts/documentation yet on the subject, I can definitely see why Kotlin is the future. It is easy to use, I learned it quite quickly and it's easy to read (for a C/C++/C# developer). However the choice of using Android Studio 3.1 Beta version might be regrettable. There were a few minor issues with the beta version, but overall it worked fine. When JetBrains came with an update for Kotlin, everything crashed and I send a report to the developers. Even when disabling it (as the image showed), nothing worked anymore. But it is unclear if this update only is send to the beta version or also to the normal version.



App

I'm not satisfied with the app that I build, because of a few factors. I learned in multimedia lectures that gradient is starting to become more fashionable in apps these days. So I chose it because of following trends. However, putting gradients in your action bar (the top bar of an android app), turns out to be a real pain. Android studio has easy ways to change the color, but then it has to be a solid color. It took me too long to figure out how I could possibly do that. The same thing goes for the navigation drawer (when you click on the hamburger icon on the top left of an android app it opens a side menu). When loading the drawer in by adding an activity, android studio will generate 4 new xml files (with 4 new UI layouts) and I had no clue how that worked. In the end I am very happy with how I solved it (I just wish I did it sooner). I build the entire drawer by hand and not by generating the activity. Now I understand is so much better. The joystick took a bit more time than I expected because of measurements on where your finger is, but no real issues there. And last I also worked on the Bluetooth part. I looked at the ways the Android developer site instructs you how to do it. I watched a few YouTube videos on how people do it and then I looked at different kinds of code. This is how I got a nice start on checking of the phone has Bluetooth, how to enable and disable it and how to pair. But I knew this one was going to be a challenge on its own because android has only 3 types of Bluetooth (for headsets, for Advanced Audio Distribution Profile (A2DP) and for health devices), none have technically to do with the simple module like ours. But I just wanted to try and pair. The moment you are ready to try and pair, that is the moment you hear that the module is broken.