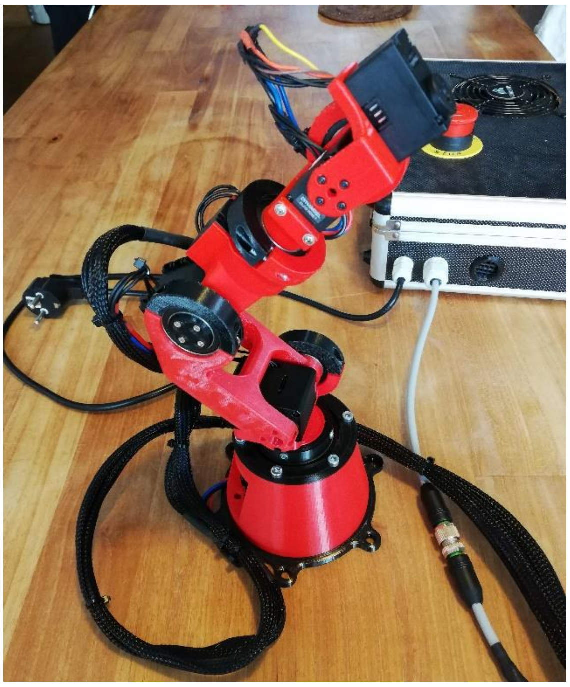
6 axis robot arm model

By Maël Abril

INTRODUCTION:

I am an automation technician apprentice, hopefully going to mechatronic engineering school next year. I was always interested in technology, and in the last years I became more and more fascinated by robotics.

In November of 2021 I decided to make the first version of my robotic arm that you can see below.



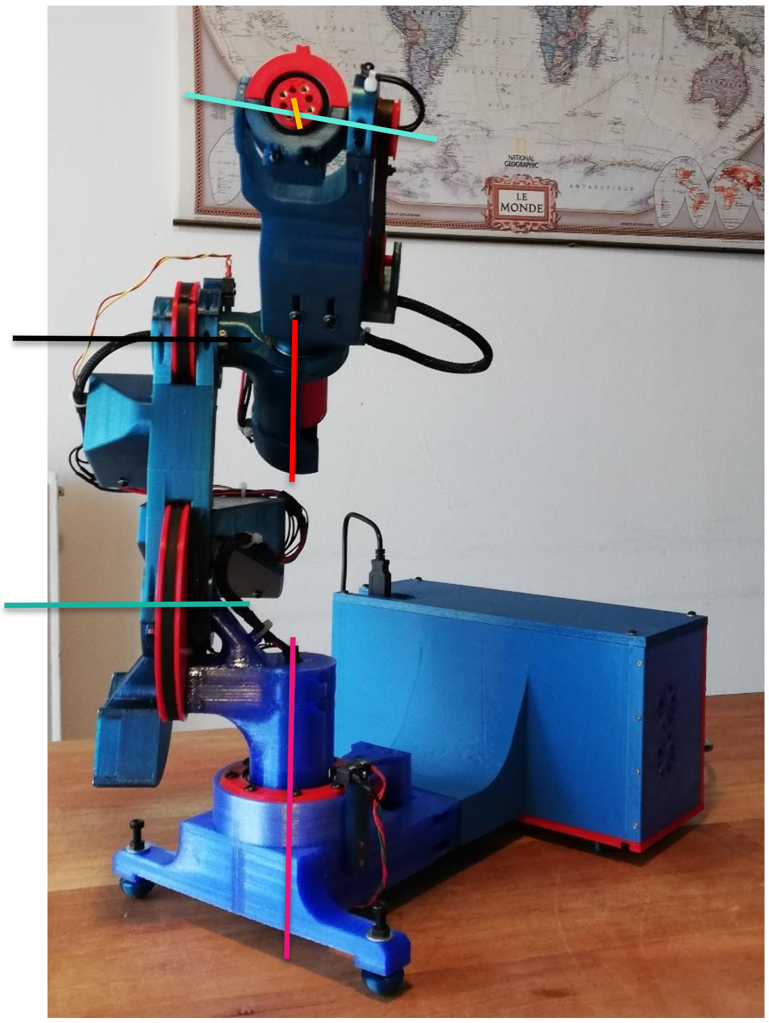
It was a very fun project, but I wasn't satisfied with the design. Therefore I decided to make a second version to hopefully correct the flaws of the first one. This document is a presentation of this second version, it is open source and you can find the CAD, python code and electrical drawings on github.com/liiamarl.

GENERAL :

This is a working model of an industrial 6 axis robotic arm with a maximum effective radius of 300mm and a payload of about 200g.

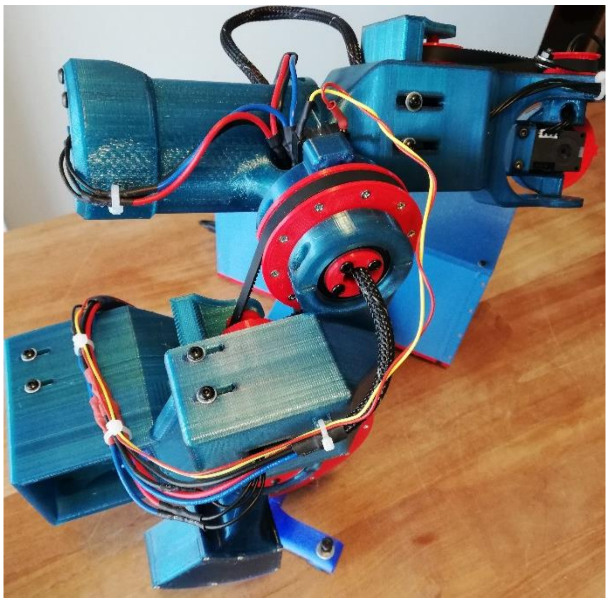
The structural pieces are 3D printed in PETG and each joint are powered by a Dynamixel XL330-M288-T servomotor.

The program runs on a raspberry Pi and there is a Robotis U2D2 communication converter that handles the communication between the raspberry and the motors. All the electronic are kept in a box directly attached to the robot's base.


Axis 1 Axis 2 Axis 3 Axis 4 Axis 5 Axis 6

MECHANICS :

All joints have the same motor, but the first 3 axis need more torque and less speed than the last 3. Therefore a reduction is necessary on the first 3 joints. The servomotors have a stall torque of 0.52Nm at 5V, but the max torque for smooth operation was not specified. I assumed a value of 0.2Nm. I went for a belt & pulleys reduction, with a ratio of 4.86 for joints 1 and 3 and 9.6 for joint 2, which have to carry the most weight with the longer lever arm. Joint 5 also have a belt and pulleys, but they are just to transmit the movement and not have a motor sticking out of the wrist, that would not be practical. The pulleys are 3D printed, which caused problems since the printer can't make sharp angles that are necessary in order to get good result with such small teeth. A lot of trial and error was needed, as well as a lot of tension on the belts in order to make it work. The tension is made by pulling on the motors blocs, which can slide when they are not tightened, and tightening them while pulling. The third joint's pulley can be seen in the following picture.



The load pulleys don't have teeth, the belt is stuck in the pulley, on the opposite side of where the motor is. This prevent the pulleys to make much more than half a turn, but that's enough for this application.

The second joint has a counterweight helping it to lift the rest of the robot. It is filled with bolts and nuts locked together and glue.

The CAD model was made in fusion360, and all pieces can be printed by a crealty ender3, but the electrical box parts must be printed diagonally in order to fit on the buildplate.

There is still one big design flaw : the connectors for the joints 2 and 3 are not accessible and if you pull on the wires and disconnect it, the only way to put it back is to disassemble the joint, and for joint 2 this is a lot of work.

ELECTRICAL :

The electrical system is quite simple, thanks to the dynamixels servomotors being connected in a bus. The whole robot is powered by a 5VDC 20A power supply. 10A should be enough but I didn't trust the cheap ones I had to really be 10A so I went for 20. There is 2 switches : one for the 230V going to the power supply, and one for the +5V going to the motors, so we can turn off the motors without turning off the raspberry Pi. There are also 2 small fan, just making sure the air doesn't stagnate and become too hot in the electrical box (probably not necessary though). $

The raspberry Pi is connected to the U2D2 by USB. The dynamixel bus power comes directly from the power supply, passing by the switch, and the data signal comes from the U2D2.

The last feature are the 3 end switches, which are connected between the 3.3V supply of the raspberry pi and the GPIO 17, 22 and 27.

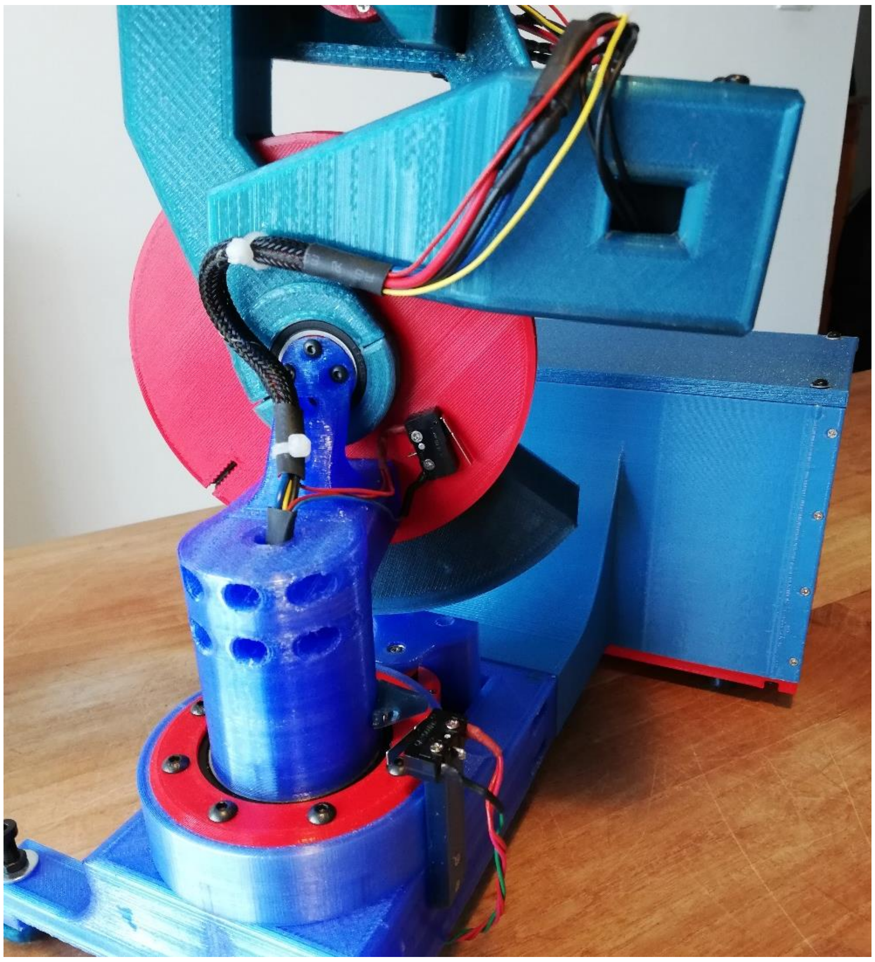
PROGRAMMING :

The program was made in python, using a few libraries like numpy or RPi.GPIO, and DynamixelSDK for the communication with the servos.

The program handles forward & inverse kinematics, joint & linear moves. The only frame disponible for now is the world frame, but I will add the tool frame and the possibility to define and move in user defined frames in the future. There is no TCP estimation, the TCP must be measured in CAD and written in the program. TCP estimation might also be added in the future.

The inverse kinematics function computes all possibilities for reaching the desired TCP position, (even though half of them will almost never be possible) and then select one of them based on how much different the joints angles would be from the current ones.

Before doing anything, the robot must be homed. Because the first 3 joints have a reduction, they make multiple turns and the software must have a way to know in which rotation the motor was when power was turned on. This is why the first 3 joints have end switches and there is a homing routine built in the robot class constructor. The home offset is then stored in and handled by the servomotors. The switches for joints 1 and 2 can be seen in the picture below.



The program includes an way to control the robot with a keyboard, and to create, save and run robot programs. The robot programs consist of moves, which can be made in joint or linear mode, in a user defined amount of time. Programs are not tool dependant so you can run the same program with different tools and it will adjust the movements so that the TCP folows the right trajectory.

Rotation is represented by ZYX euler angles in the robot programs, but rotation matrix and other representations are used in the main program. Almost half of the robot_maths.py file is just made of functions that convert one representation to another. I never thought 3D rotation would be that hard to work with.

CONCLUSION :

I'm verry pleased with how this robot turned out. It is far from perfect, but the improvement from the first version is huge. The mechanic is working quite well, and I'm almost shocked that I was able to make software that can control it as well as it does. It have a precision of 1mm with no load, but it moves by +-5mm if you apply force on it. I think it is quite good considering it is 3D printed and costs less than 450€.

That is of course not going to be the end of it, I'm already thing about making a version 3, using stepper motors this time, and with an aluminum plates structure. It will have to wait though.

Thank you for the interest you give to this project.