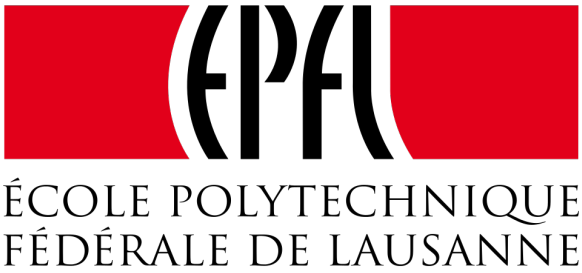




# Low-Cost Monopedal Hopping Robot

Semester Project

*Author:*
Iris Rentsch

*Supervisor:*
Amy Wu
Prof. Auke Jan Ijspeert

Spring 2018

# Contents

# List of Figures



# List of Tables

# 1 Introduction

As Raibert stated already in 1986 [1], legged vehicles can travel over difficult terrain, whereas for wheeled vehicles prepared surfaces such as rails and roads are necessary. He explained it with the example of a ladder: Rungs provide footholds for legged systems whereas the spaces between the rungs prohibit the ascent of wheeled systems. Legged systems can thus choose among the best footholds, while a wheel must negotiate the worst terrain.

There is another advantage of legged systems: a legged system can step over obstacles by letting the body have a smooth trajectory. This is due to decoupling the body trajectory from the foot trajectory.

Humanoid legged robots nowadays, Asimo [2] for example, can walk on even ground, but are too complex to go over uneven terrain. In contrast, there are quadrupeds and hexapeds in different forms existing that perform well. For example, there are robots with spider-like legs, with caterpillars or with wheel-like legs that perform more or less well on uneven surfaces. However they are very expensive (tenthousands of dollars) and cannot be used by most research institutes to come further with testing new gaits or for education in the classroom. This is why we want to build a low-cost bipedal running robot.

As there is already Rando existing [3], the bipedal walking robot, it would be a good continuation to have a running robot. It should follow the spring loaded inverted pendulum (SLIP) motion model and have a tabletop size.

As it is complicated to directly build a two-legged walking robot and to build in two legs before having tested its behaviour in one leg only, we decided to first build a one-legged hopper. This is thus the scope of this project.

We want to build a low-cost one-legged hopping robot which moves in 2D. Its price should be below 100 CHF. The project involves design, construction and control parts. It will give insight in the choice of components for a low-cost construction and also in leg behaviour.

This report begins with a literature review, followed by the design and construction process used, together with the control strategy employed. Then experimental results are presented and conclusions are drawn. In the end, some possible future work is suggested.

# 2 State of the Art

## 2.1 Working principle

A first inspiration was drawn from the Raibert hopper, developed at the MIT in the 1980s [4]. This hopper was about one meter in height and was actuated with pressurized air provided through the boom it was fixed on. It had sensors to measure the pitch angle of the body, the angle at the hip, the length of the leg, the tension in the leg spring, and contact with the ground. The control of this machine was partitioned in three parts. These parts are described later in section 2.3.

For this one-legged machine running and hopping were the same. One cycle has two phases, stance and swing. During the stance phase the foot stays at the same position on ground and the mass, which is concentrated at the hip, tips over like an inverted pendulum. During swing, the center of mass covers a ballistic trajectory with the leg in

the air and free to move.

The movement template described before is known under the name Spring-Loaded Inverted Pendulum (SLIP) and is generally accepted as a model to describe mono- and bi-pedal locomotion.

As the name indicates, the system consists of a mass on top of an ideally massless spring. At the end of the flight phase, the foot lands with a certain angle of attack on the ground. The leg is contracted and stores energy in the spring. In the same time the mass tips over the foot and then the stored energy is recovered and the foot leaves the ground again. This is the beginning of a new flight phase, where the angle of attack has to be adjusted in order to prepare for a new landing and with this for a new stance phase. The stance phase can be symmetric or asymmetric, but only symmetrc stance phases lead to cyclic movement trajectories [5].

Fig. 1 shows the stance phase with the corresponding angle of attack. It is the angle of attack that determines the stance phase behaviour and also if the mass is accelerating or decelerating. It is thus crucial to control in order to achieve stable locomotion.

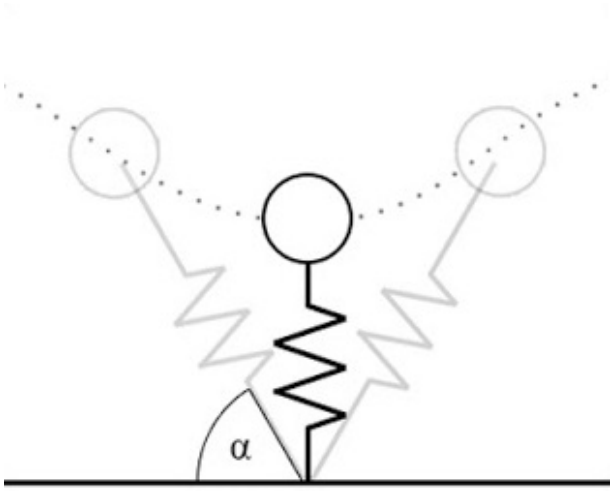



Figure 1: Stance phase of the SLIP model, $\alpha$ is the angle of attack

The SLIP model is an ideal description of motion and does not account for friction losses. It will therefore be necessary to provide energy to the system before each jump in order to continue the cycles. Even when a spring is added in the leg, it will not be sufficient for long-term hopping. To provide energy at each jump, a motor has to be added to the leg, not only to change the hip angle, but also to compress the spring.

One solution to provide energy to the system is to add a tail. The robot using this principle is the Penn Jerboa [6] developed at the Pennsylvania University in 2015. Each leg has one motor to adjust the hip angle and an additional motor is added to the tail in order to provide energy to the sytem. A tail was not used for this project to make a more human-like robot.

## 2.2 Design

For a table-sized low-cost hopper as it will be built in this project, it is not possible to use pressurized air to change the leg length as it was done by Raibert in the 1980s. Therefore, inspiration was searched in the internet by asking for "low-cost" and "hopper". What

came out was a flea inspired jumping robot, built at the University of Southampton in a course [7]. The leg has a parallel configuration and both sides of the leg were connected through a spring and could thus be tensioned when a force from top compressed the leg. When the force was released, it jumped up in the air. This process is shown in Fig. 2.

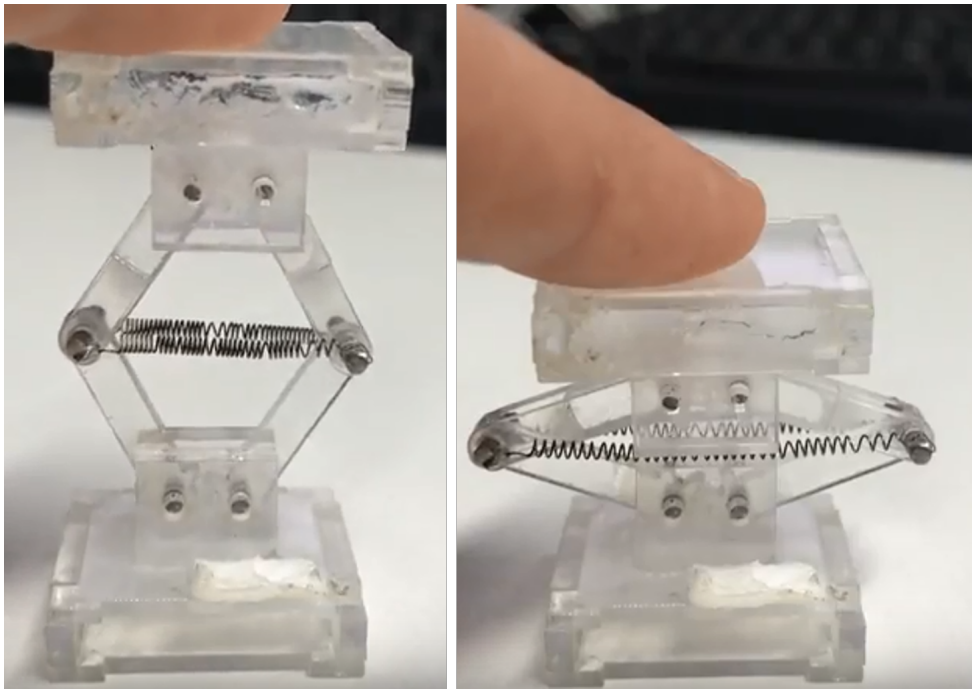


Figure 2: Flea inspired hopper built at the University of Southampton

Being inspired by this mechanism, I found Minitaur [8], built by Ghost Robotics company, which is a quadruped robot, having the same parallel leg configuration. It uses two direct drive DC motors per leg and it is able to jump about 20 cm high with its weight of 5 kg. It is shown in Fig. 3. Minitaur is a small, highly agile, lightweight and dynamic legged R & D platform and its price is 11500 US$. It can measure the force that is put on the leg using a torque sensor in the motor and deduce from the measured force on what ground it is walking. It is thus able to adjust balance on sand, ice and dirt.

At the RoboMechanics Lab [9] they developed a low-cost version of minitaur which costs around 200 US$. It does not have the characteristic parallel leg configuration and is thus not able to jump either. It is shown in Fig. 4.

## 2.3 Control

Raibert [4] had the following partitioning of control for his one-legged hopper: One part controlled the hopping, a second part the forward speed and a third part the posture of the body. According to the SLIP model, the hopping describes an oscillating motion. During the support phase, the springy leg is compressed and decompressed, and during flight, the mass travels a ballistic trajectory, governed by gravity. To regulate forward speed and acceleration, the hip angle can be adjusted during flight, and thus the angle of attack can be defined before each landing. Depending on the positioning of the foot, the body will keep the speed or accelerate or decelerate. The one legged hopper of Raibert [4] took into account the actual forward speed, the desired speed and a simple model of the legged system's dynamics in order to calculate a suitable position for the foot. During

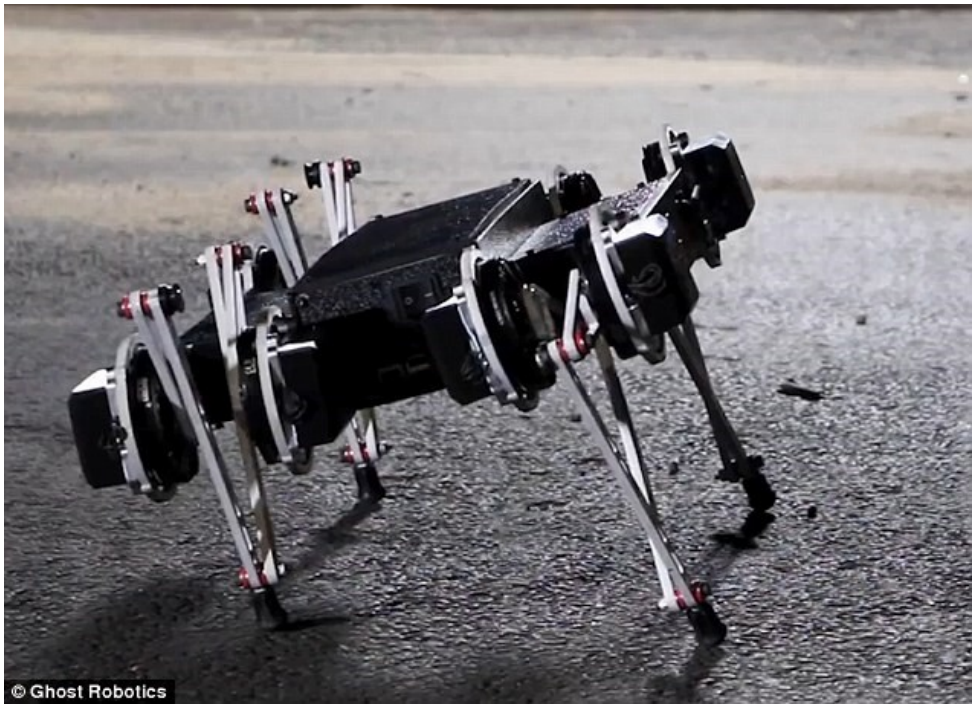



Figure 3: Image of minitaur [8]

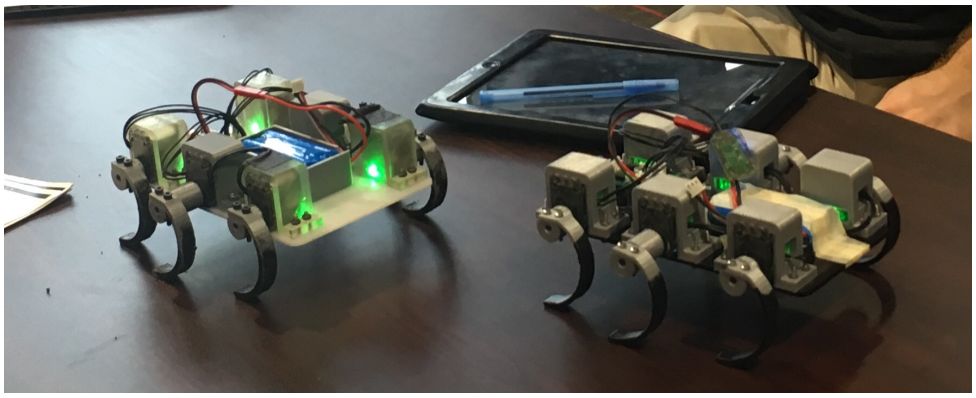


Figure 4: Low-cost version of minitaur [9]

each stance phase, the body is restored to an upright position in order to stabilize the pitch angle of the body. In Raiberts hopper [4], this was done with a linear servo.

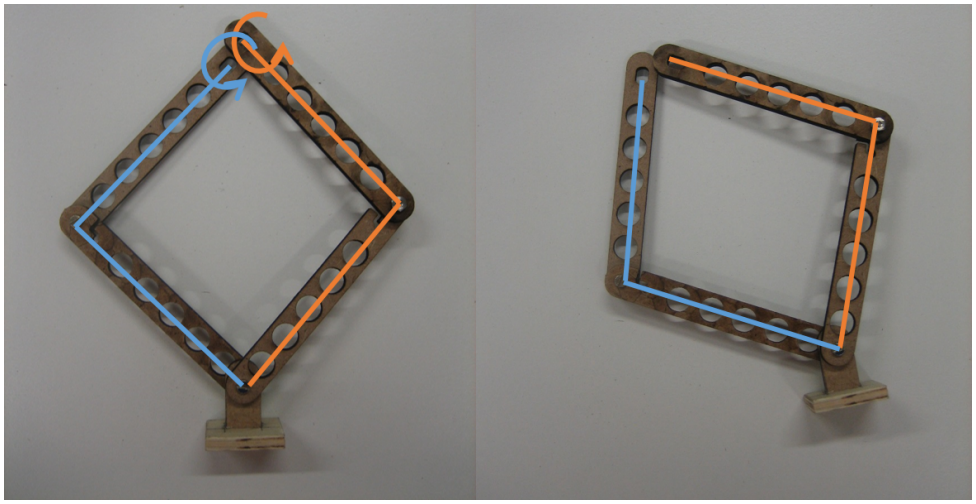
# 3 Design and Construction

To design the robot, Catia for Students was used to draw the parts. They were then saved in the 2D view as a pdf file. This pdf file can be opened by the graphics software used by the laser cutter. First trials were made with MDF wood plates of 3 mm thickness, some parts were 3D printed and later some parts were cut in 2mm thick acrylic.

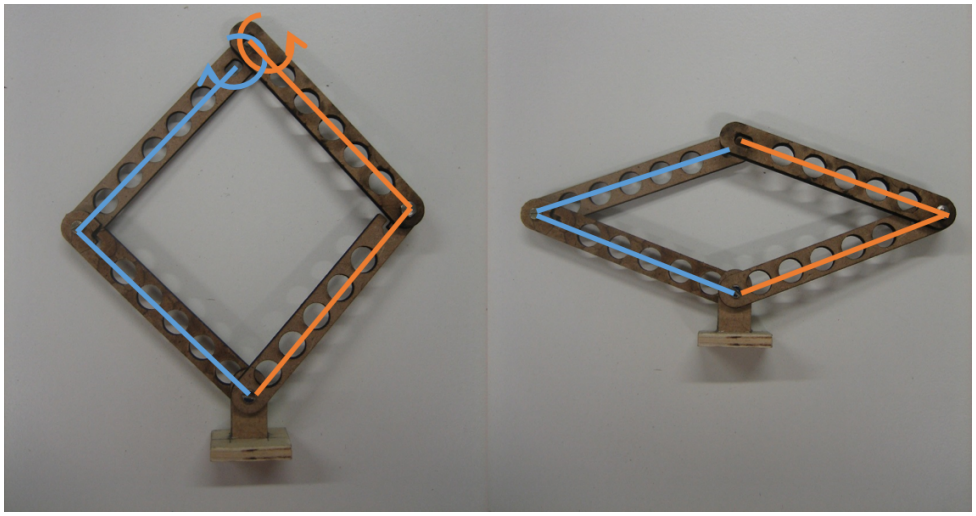
The parts were either glued together or fixed with screws when a movable joint was needed.

## 3.1 Working principle of the robot

The robot has a trapezoidal leg, meaning that the leg has two sides going down from the hip to the foot. It is thus a parallel configuration. Each side has its own motor. When both motors are turning in the same direction, the hip angle is changed (see Fig. 5a) and when they are turning in opposite directions, the leg length is changed (see Fig. 5b).



(a) Hip angle adjustment when both motors are turning in the same direction



(b) Contraction of the leg when the motors are turning in opposite directions

Figure 5: The working principle of the robot legs

At the joint on each side of the leg, several rubber bands are added in order to store energy upon compression. The energy can also be provided by the motors when they contract the leg. This will be used to make the robot jump and to store energy from the spring compression during stance, as captured by the S (spring) of the SLIP model.

## 3.2 Robot design

The first design (shown in Fig. 6) was simple, the structure was not yet optimized to save weight. The legs (number 4 and 5 in the figure) had a length of 100 mm, the top plate (6) had slots to glue the motor holders (3) and to put the nano microcontroller board and the IMU on. The cables would need to pass below the top plate and could interfere with the leg movements. The foot (1) and its adapter to the leg (2) were not optimized yet. Especially, the foot was big to provide stability during stance, but was also heavy.

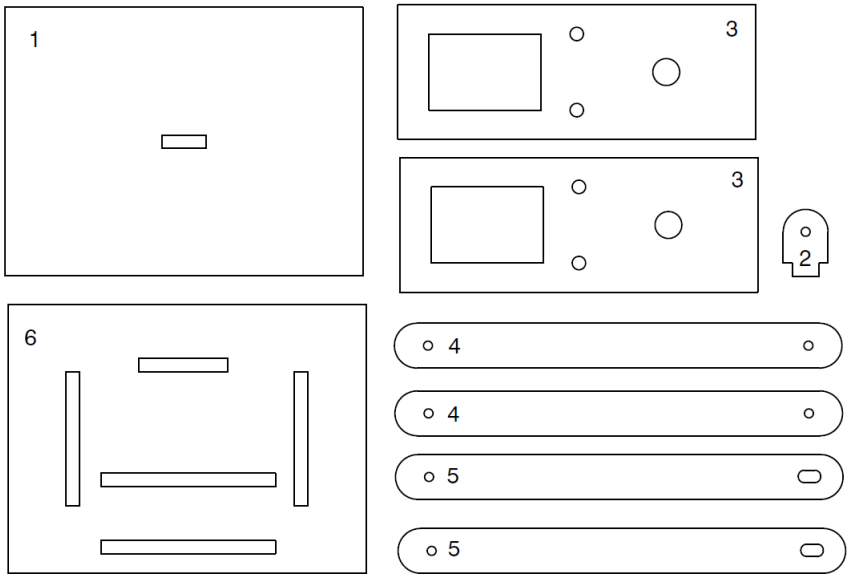



Figure 6: Parts to cut out of 3 mm thick MDF, first trials. Scale = 60% of original size

As a first thing to change, the legs were shortened to 50 mm, which turned out to be too short and finally defined to have a length of 75 mm. This leg evolution is shown in Fig. 7. On the left side are the legs with a length of 100 mm, followed by the ones with a length of 50 mm. The black ones were 3D-printed, because it was in my holidays that they had to be replaced and I did not have access to the laser cutter. They have a length of 70 mm. The last legs have a length of 75 mm and have holes cut in, to save weight. It turned out that this length was fine, because the legs were just a bit longer than the motors and can easily pass aside without touching the motors with the screws.

As the gait model used is the SLIP model, the mass has to be concentrated at the top. This will not be entirely possible but it is tried by making the legs as lightweight as possible. The foot should be small. Having done so in the very first prototype, it became clear that it has to be bigger for trials in order to provide more stability. It became thus biger and in a next step lighter. Later in the design phase, an arc foot shape was added. As stated in [10], the arc should have a radius of 0.3x(Leg length). This last arc foot has also an integrated switch in order to know when the foot is touching the ground. This is important for the control, as it will be discussed later. The evolution of the foot is shown in Fig. 8.

The top motor holders (number 3 in Fig. 9) were optimized to be glued more easily and to ensure they are perpendicular to the top plate (6) when glued. They were also made lighter by removing more material. The top plate was adapted to the improved motor holders, holes were put where the Nano and the motor controller board had to be fixed. There was space left for the battery between the top plates and the motors. The arc foot

profiles (7) are designed to fit on the foot (1). Two of the arcs have a hole to fix the switch-holder (8) in. This last design is shown in Fig. 9 (the parts to be cut from MDF) and 10 (the assembled parts on the robot).

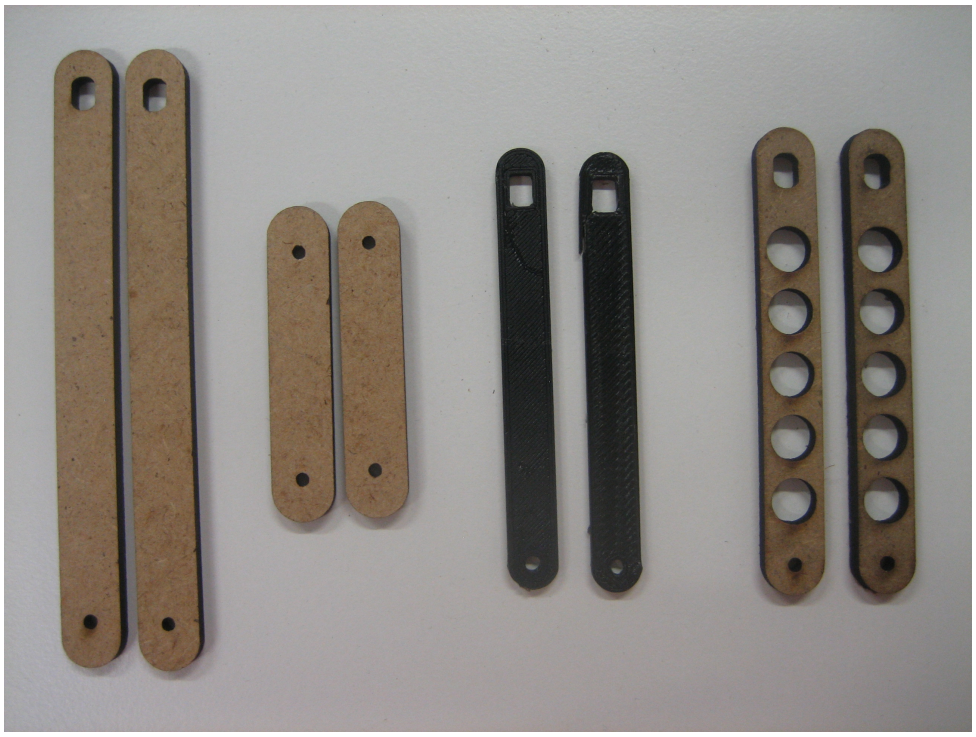


Figure 7: Evolution of the leg design, from right to left, from the earliest to latest respectively.

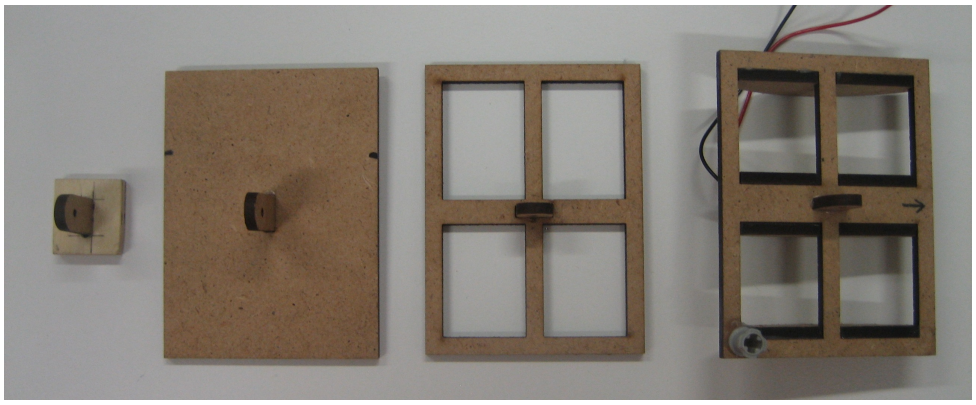


Figure 8: Evolution of the foot, from right to left, from the earliest to latest respectively.

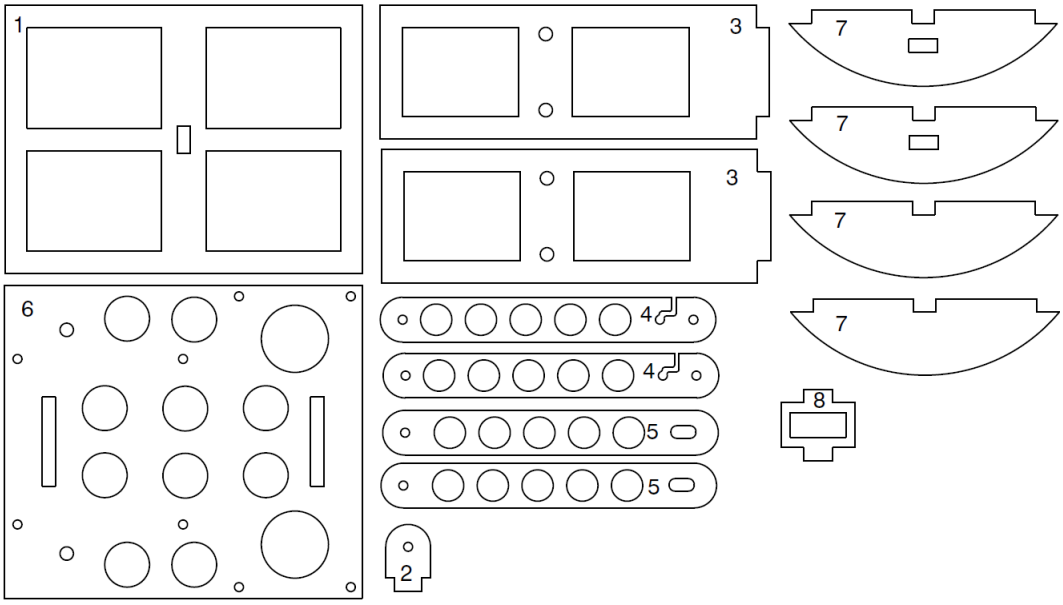



Figure 9: Optimized parts, to be cut from 3mm MDF plate. Scale = 60% of original size

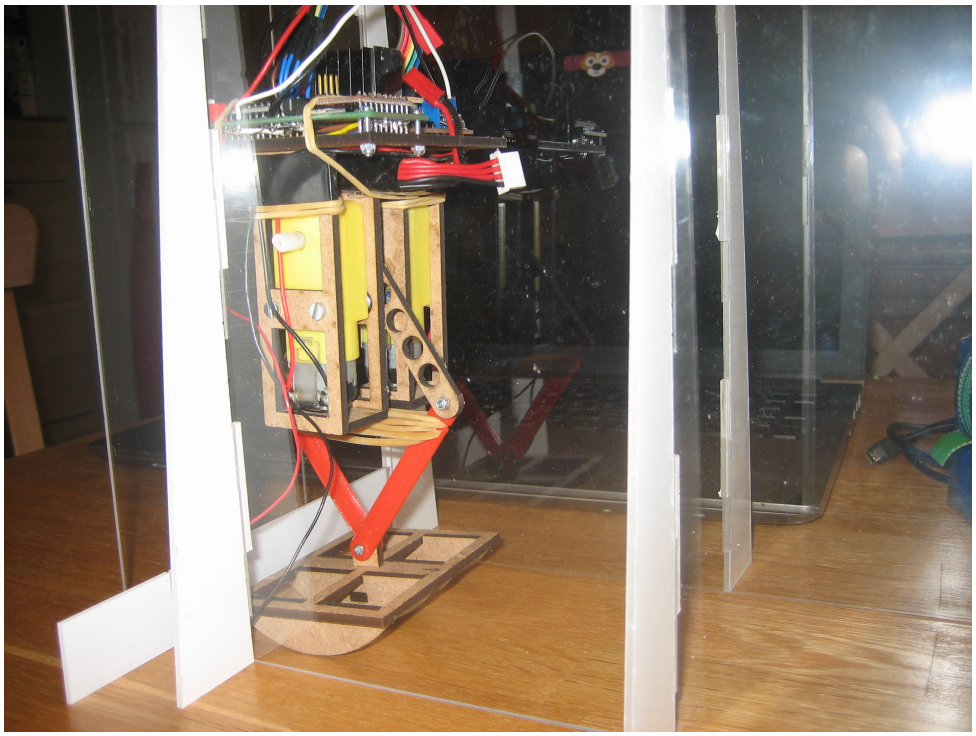


Figure 10: Optimized parts, assembled on the structure. Only the lower parts of the legs were 3D printed because it was in the holidays that they broke and I did not have access to the laser cutter.

It turned out that the legs were not stable enough. The fixation hole to the motors is increased in size, so that it turned freely. This is why the last trials were done with legs cut out of acrylic. This is also what I would suggest to continue with in future work.

## 3.3 Supporting structure

The design of the first supporting structure, blocking forward- and sideward motion, was simple and low-cost. There were just long parts cut out of MDF and glued to a base of MDF with holes. See Fig. 17a for an image of the structure. The height of the structure is 30 cm in order to allow the robot to jump. The design for this first supporting structure is shown in Fig. 11.

Figure 11: Parts to cut from MDF, the size of the plate used is 400x150 mm.

For later trials, a second supporting structure had to be built. This time, it should allow forward movements, but provide lateral stability. Therefore walls were constructed out of acrylic. Acrylic has the advantage of being transparent. It is a bit more expensive, but the plates being only 2 mm thick, the structure is lighter. The dimensions of the structure should allow the robot to jump, so it should have a height of about 300 mm and the robot should also be allowed to hop forwards and backwards. As the delivered plate has a size of 400x500 mm, the length of the walls was chosen to be 490 mm. In order for the walls to stand, there were gussets added. These should be cut from the same plate and minimize the wasted material. In order to see the final structure and to decide on the detailed assembly, a 3D model was generated. It is shown in Fig. 12.

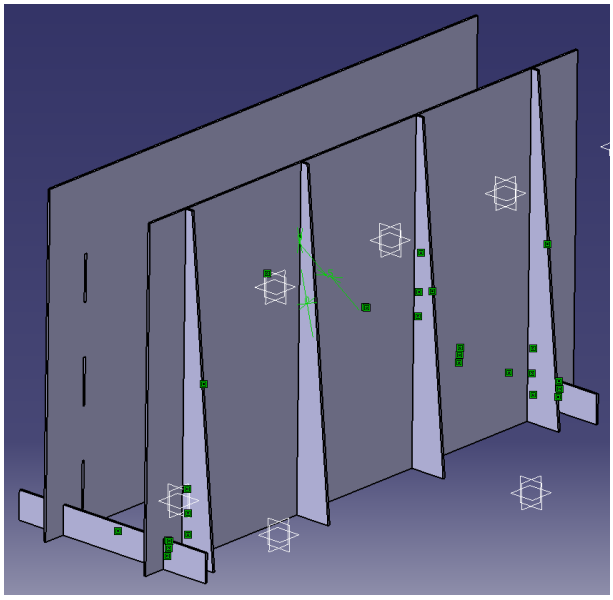


Figure 12: 3D model of the walls

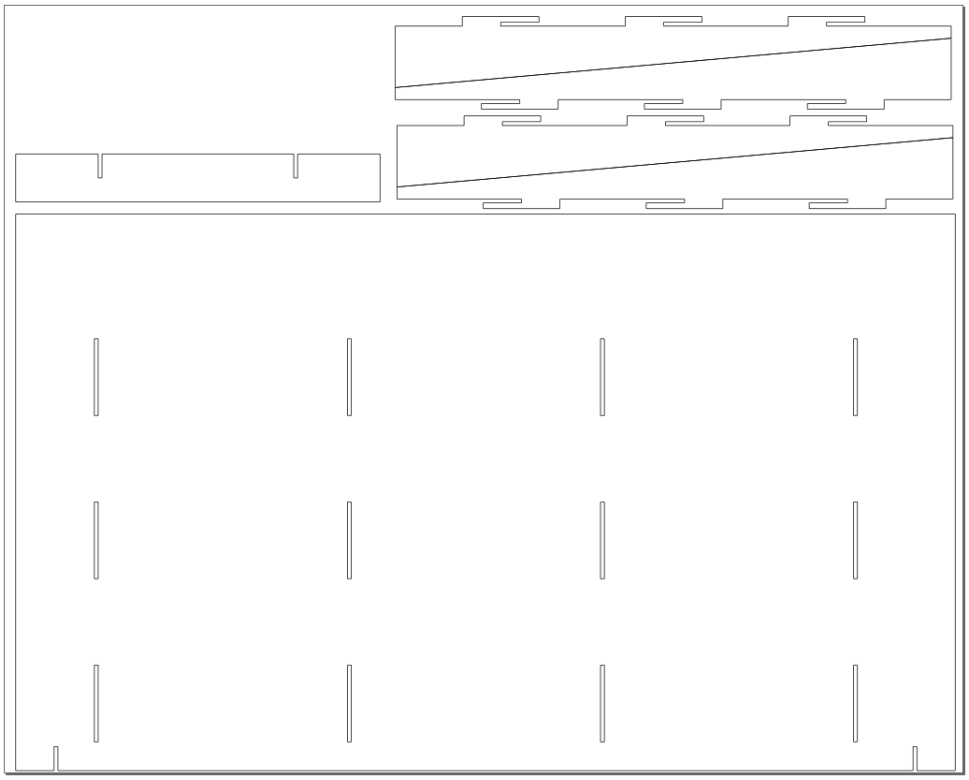


Figure 13: Parts to cut from acrylic, the border around shows the size of the acrylic plate

## 3.4 Components

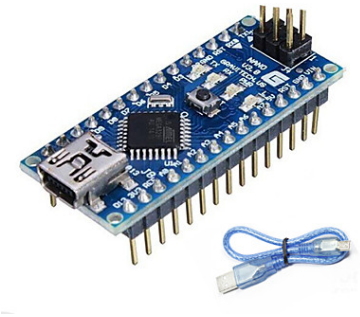
In this section, the different components are described. In the end, the linkage diagram is shown.

**Arduino Nano** *6.30 CHF at miniinthebox.com*, shown in Fig. 14a. It is a compact microcontroller board with digital and analog inputs and outputs. There are some digital output pins that allow pulse-width modulation (PWM) as it is used by the motor controller board to control the speed of the motors. The specifications can be found in [11].

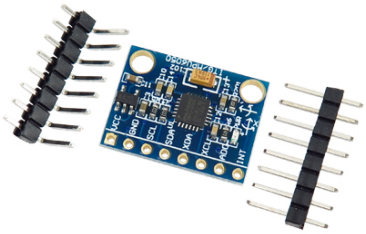
**MPU 6050** *6.50 CHF at Yampe Suisse*, shown in Fig. 14b. This is a 3-axis MEMS accelerator and a 3-axis MEMS gyroscope combined in one chip. The specifications and tips for usage can be found in [12].

**Hyperion Arduino Compatible L298N Dual H-Bridge Moto** *16.40 CHF at digitec.ch*, shown in Fig. 14c. This H-bridge can control two motors at a time. The turning direction and speed can be given to the board. The specifications are found in [13].

**Sparkfun 1:48 DC Gearmotor** *2x 3.90 CHF at Digikey.com*, shown in Fig. 14d. These are the very low-cost Arduino motors. The ones I chose have a gear ratio of 1:48 and have thus a speed of 140 RPM with a supply voltage of 4.5 V. This is the no-load speed. As these motors are also supporting voltages to up to 9 V, and we know that the motor controller board causes a voltage drop of about 2 V, the whole system is powered with 11.1 V.



(a) Arduino Nano

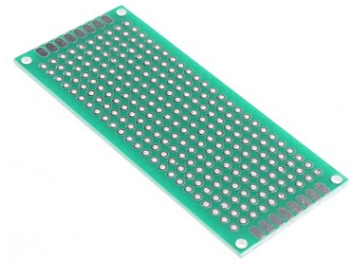


(b) MPU6050

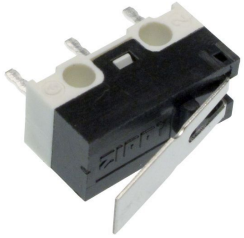
(c) Motor controller board

(d) Motors

(e) Battery



(f) Prototype plate



(g) Foot switch

(h) 1 kΩ Resistor

Figure 14: Components used

**Swaytronic 3S 11.1V 330mAh 45C/90C JST Battery** *13.20 CHF at digitec.ch*, shown in Fig. 14e. To provide 11.1 V to the robot, a 3S battery was chosen. The current used is very small and 330 mAh are well enough to power the small robot for hours. The weight of the battery could be optimized by buying small cells and soldering them together. Like this a battery of 10 grams could be achieved.

**Prototype board** *3.15 CHF at miniinthebox.ch*, shown in Fig. 14f. To link all the components conveniently, a prototype board was used.

**Switch and Resistor** *0.25x 1 CHF at protosupplies.com*, shown in Fig. 14g and 14h. The robot has to know if it is touching the ground or not, in order to use the state controller as Raibert [4] was using it. In order to link the switch, a resistor of 1 kΩ is used.

The total price of the robot is 53.60 CHF for the components, excluding the price for screws, cables, rubber bands and the laser-cut parts. In total we can state, that the price for the robot will be below 100 CHF which was the goal to achieve.

The linking of the components was done on a prototype plate and they were directly soldered to it. The linkage diagram is shown in Fig. 15. There is an additional voltage regulator to the built-in one on the Nano because the Nano microcontroller are very low-cost and do not have good components. I have broken two Nanos because there was too much voltage transmitted. The motor controller takes the total 11.1 V and gives about 9 V to the motors. The Nano takes 5V via the VIN pin and the IMU takes 3.3 V from the Nano. All the ground voltages have to be linked in order to have the same reference.

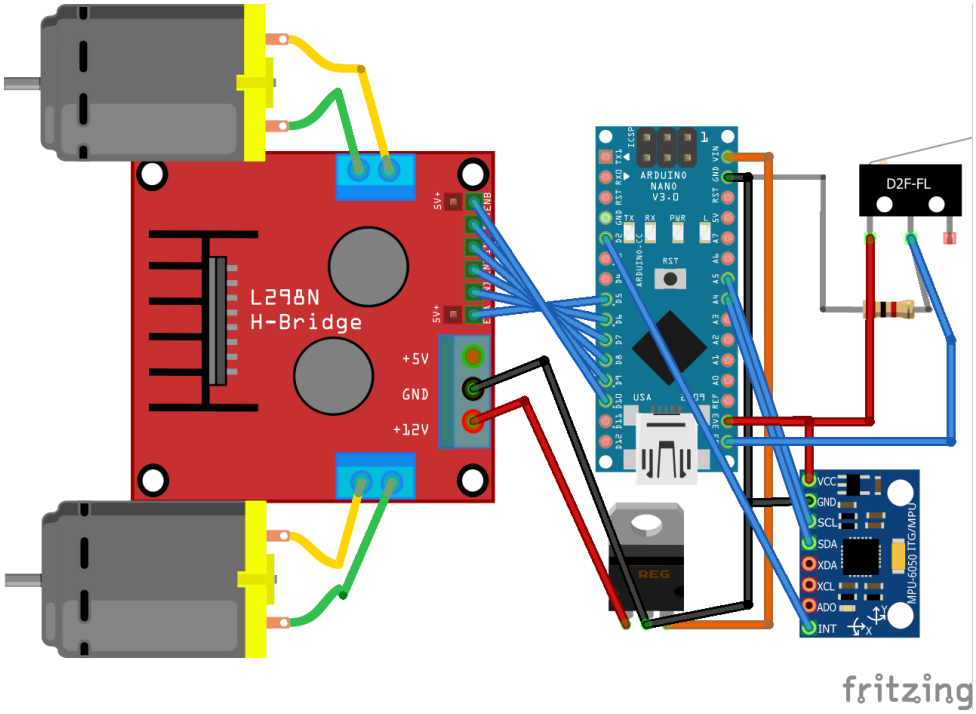



Figure 15: Linking diagram of the parts

## 3.5 Control

In the robot built during this project, control was performed with an Arduino Nano. To control the DC motors, the L298N motor controller board was used. The inclination of the top surface with respect to ground is measured with an IMU (MPU6050) and read out with the Nano. This reading helps in calculating the duration and direction the motors have to turn.

This part of the project is the least developed and should be investigated in more detail in a following project. For now, the robot is able to sense if it is touching the ground with the foot due to a foot switch and can thus differentiate between two programs. When the foot is in contact with the ground, it is in stance phase and should thus prepare for a jump. The leg is thus contracted due to the motors. When a new jump begins, the motors turn in the direction to extend the leg and the energy stored in the rubber bands is released and the robot jumps. As there is no more contact with the ground now, the robot is in flight phase and the IMU comes into play. It delivers the angle of inclination of the top surface and the Nano microcontroller calculates the amount of time the motors

have to turn in order to achieve the right hip angle, adjusting thus the angle of attack for the next landing. The Code to control the robot has been commented and is shown in Appendix A.

As the IMU has to be calibrated in order to provide absolute angles, a code from [14] was used. This code is found in the documentation of the project under the name `IMU_offset_calculation`.

# 4 Experimental Results

The very first experimental trials concerned the IMU and the servo motors only. To test the IMU, a code from [15] was used. In this code, the angle from the IMU is used to make turn the servo motors in different directions, corresponding to the direction of inclination of the IMU. As a result, the servo motor has always the same orientation as the IMU. This is shown in Fig. 16. As this was easily achieved, the robot was designed and built and the next trials concerned already the jumping of the robot.

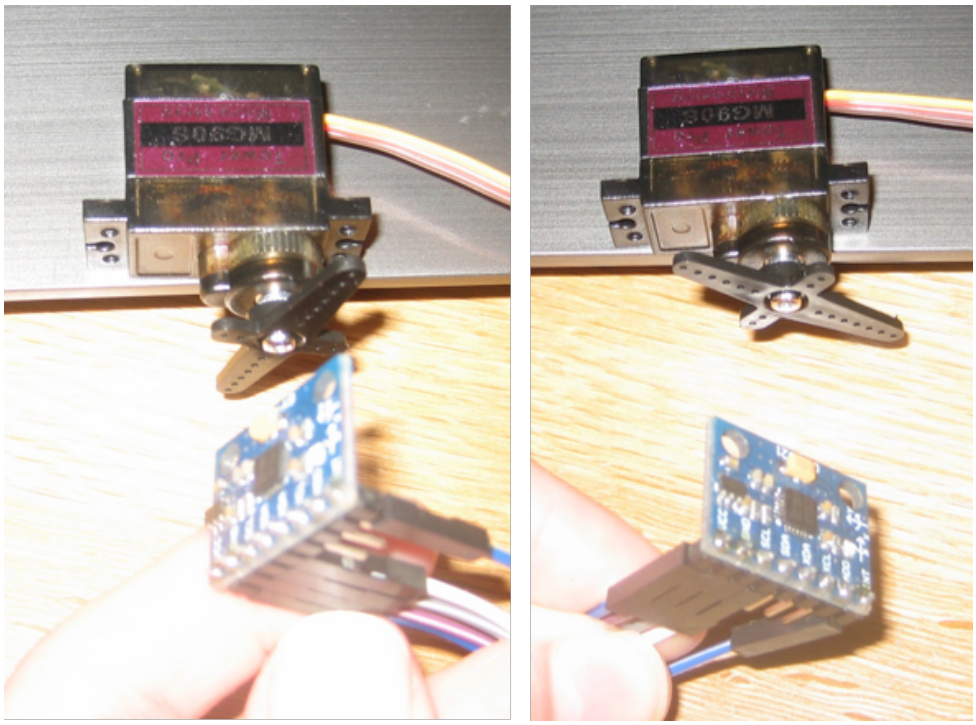
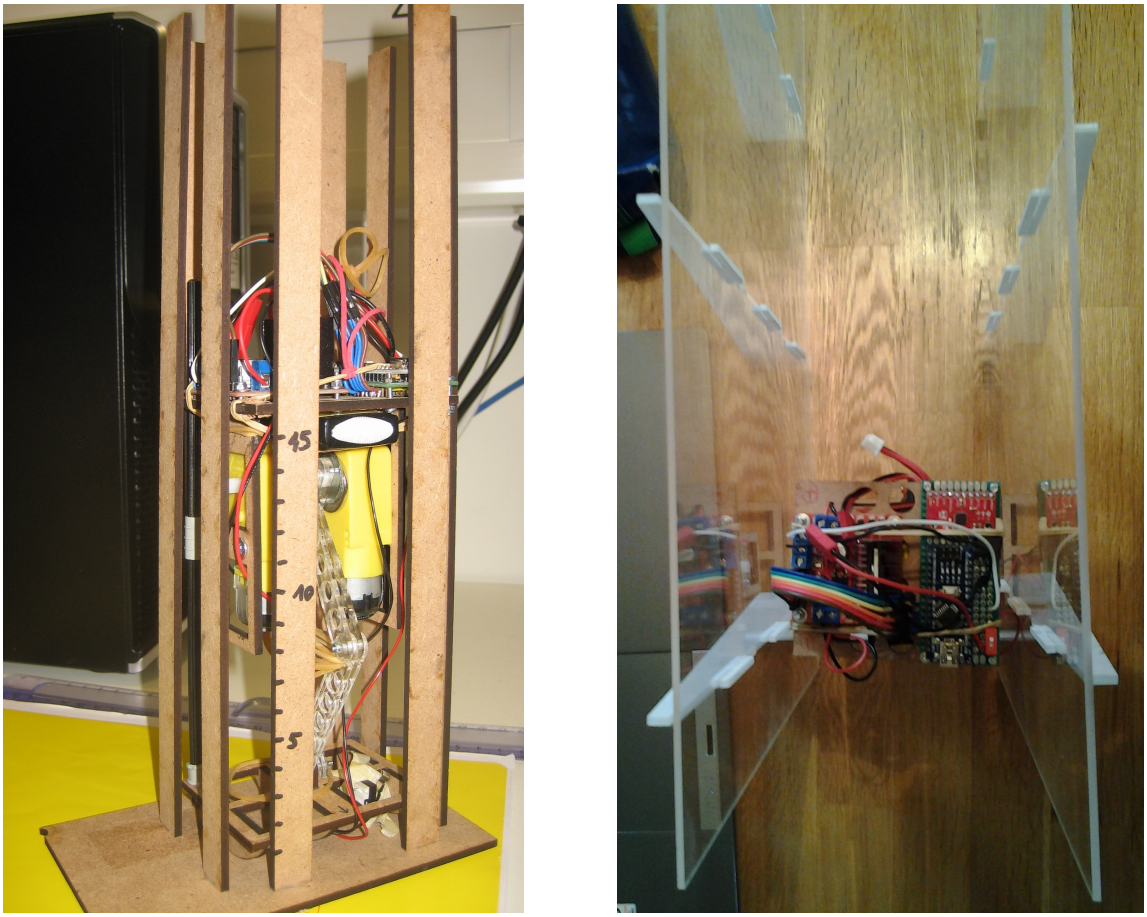


Figure 16: First trials with servo motors. The motor has always the same orientation as the IMU.

The first jumping tests were done with low-cost servo motors (as the ones shown in Fig. 16). But they were not fast and strong enough to make the robot jump. As they cannot move freely when no command is applied, they could not compress a spring to be let free after. This is clearly an advantage of the DC motors. With them, the spring only needs to be loaded, and the fast unloading is performed freely and is thus faster than the loading. However, the DC motors do not know their position and it is not possible to know in what state the leg is. But as the rubber bands are strong enough to alway elongate the leg, it will be elongated as soon as there is no more force on the motors to counteract the rubber bands and an equilibrium position will be attained.

First, jumps in free space were tested, no direction of movement was constrained. Then a structure was built to allow jumping in vertical direction only. Lastly, two walls were built to allow forward movement during the jumps. The robot was thus stabilized laterally. The last two setups are shown in Fig. 17.




(a) Robot moving in 1D

(b) Robot moving in 2D, constrained sidewards

Figure 17: The different setups used for the experimental tests.

## 4.1 1D jumping

Using the DC motor with rubber bands in between the leg parts yields different jumping heights, depending on the voltage applied and the number of rubber bands used. Tests were done with a voltage generator and later with a battery (which means additional weight). These tests are summarized in Tables 1-3. The program chosen references to the Arduino code and the number in () is the amount of time in milliseconds, the motors are turning. bothUP(50) means for example that both motors are turning in the direction to extend the leg for 50 ms. bothDOWN(100) instead means that both motors are turning in the direction to contract the leg for 100 ms.

Fig. 18 shows the jumping height with two rubber bands used. For 8 V employed voltage, it lifts off, but the jumping height is only about 1 mm. When the voltage is increased, the jumping height increases linearly. As these test were successful, I asked myself how to increase the jumping height even more by keeping the voltage at 11.1 V and added more

| **Program chosen** | **# Rubber bands** | **Voltage** | **Height** | **Comment** |
|---|---|---|---|---|
| bothUP(100) | 1 | 6 | No | Straight legs |
| bothUP(50) | 1 | 6 | No | |
| bothUP(50) | 1 | 7 | No | |
| bothUP(50) | 1 | 8 | No | |
| bothUP(50) | 1 | 9 | No | Just lifts off |
| bothUP(50) | 1 | 10 | ∼1 cm | |
| bothUP(50) | 1 | 11 | 2 cm | |
| bothUP(50) | 1 | 12 | 3 cm | |

Table 1: Jumping height with different conditions, extension of the legs from the equilibrium point was controlled.

| **Program chosen** | **# Rubber bands** | **Voltage** | **Height** | **Comment** |
|---|---|---|---|---|
| bothUP(50) | 2 | 6 | No | |
| bothUP(50) | 2 | 7 | No | Just lifts off |
| bothUP(50) | 2 | 8 | <1 cm | |
| bothUP(50) | 2 | 9 | 2 cm | |
| bothUP(50) | 2 | 10 | 4 cm | |
| bothUP(50) | 2 | 11 | 6 cm | |
| bothUP(50) | 2 | 12 | 8 cm | |

Table 2: Jumping height with different conditions, extension of the legs from the euilibrium point was controlled.

rubber bands as a consequence. Table 3 shows the jumping height when the voltage is at 11.1 V, but the number of rubber bands is increased.

Tables 1 and 2 use control sequences to extend the leg with the motors, whereas Table 3 shows conditions where control sequences were used to contract the leg and charge the rubber bands with it. The extension is done solely by the rubber bands, the motors turning freely. This solution is preferred, as the speed of the motor is not crucial as long as they are strong enough to charge the rubber bands with enough energy to make a hop. The last trials showed that the jumping height can be even increased, when a short impulse to hop is given from the motors after the rubber bands have been charged. This is shown in Table 3 in the last line.

Last tests with the battery added were finally done. Therefore, 35 g were added to the robot. The battery provides also 11.1 V and the robot did still jump 10 cm high. The next steps of making it jump forward can thus be started.

## 4.2 2D forward jumping

While the main goal of this project was to make a hopping robot, which was successful, I wanted to test jumping with forward motion. Therefore two walls were created that prevent the robot from falling sidewards. They were cut out of acrylic in order to be transparent. The robot hang on rubber bands to prevent it from falling on ground upon slipping. This setup is shown in Fig. 17b.

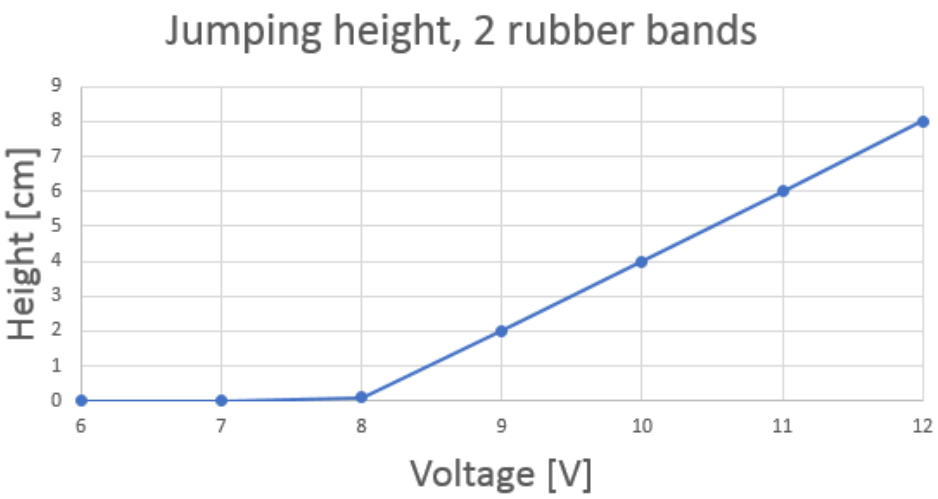



Figure 18: Jumping height for different voltages, leg with 2 rubber bands. It is very linear because the measured values were rounded to the next integer as they were measured by eye.

| **Program chosen** | **# Rubber bands** | **Voltage** | **Height** | **Comment** |
|---|---|---|---|---|
| bothDOWN(100) | 3 | 7.4 | No | Hits the bottom |
| bothDOWN(70) | 4 | 11.1 | No | Hits the bottom |
| bothDOWN(70) | 5 | 11.1 | 2 cm | Does not hit |
| bothDOWN(70) bothUP(50) | 5 | 11.1 | 10 cm | |

Table 3: Jumping height with different conditions, this time only contraction of the legs was controlled, extension was done due to the rubber bands. After each command there was a delay of 3000 ms to allow interruption by hand.

Because of the legs being sometimes blocked by the motors and the motor still turning, the legs broke. Therefore a design change was done and the motors were covered by MDF in order to prevent the legs from blocking. Having this done, the test were still not successful because there were too many degrees of freedom that cannot be controlled. In fact, the IMU measures only the inclination of the top plate, but does not know how it is with respect to the foot. In order to block this degree of freedom, the foot and the top plate have to be linked by a bar. This bar makes sure that the top plate is always in the same orientation as the foot and that they remain parallel. As a first prototype for this stabilizing bar, a LEGO piece was used, as is shown in Fig. 19. This can certainly be improved and the bars should also be put in every corner in order to prevent rotation between the top plate and the foot.

Some trials were done with this stabilized structure, but very soon, the motors broke because there were too many times the legs blocked, so that the gears inside the motors became hot and broke. As this happened in the last week of the project, I decided not to change them, but to leave a prospective student with a well explained report and all the necessary drawings for laser cutting, so that he/she can create a new prototype in the beginning of the next project on this subject.

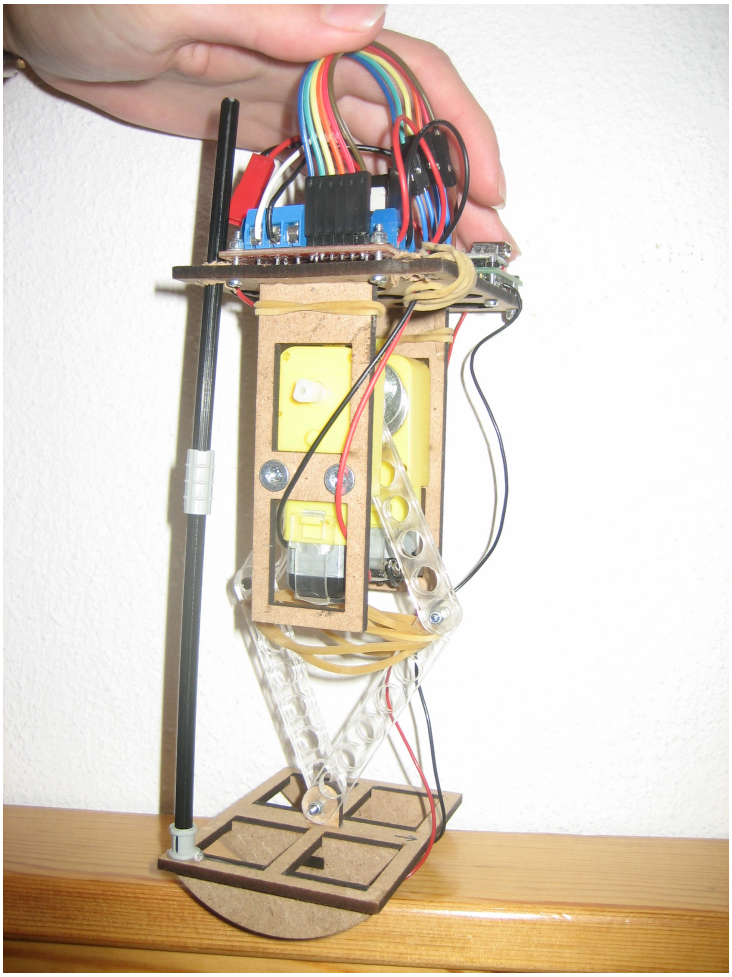


Figure 19: LEGO-bar stabilizing the structure.

# 5 Conclusion

Building a low-cost one-legged hopping robot is possible. However, the parts have to be exchanged regularly when it is not ensured that the motors cannot apply a big force when the legs are blocked for example. But once the control is working, and the legs can no more getting blocked, the low-cost components should do their job well.

MDF is not strong enough for the legs, indeed the hole where the motor is fixed get bigger when too much force is applied. But this was mainly due to the legs getting blocked and the motors applying too much force to them. Again, when the control is working well, these problems can be avoided. In between, legs cut out of acrylic can be used, as they are harder. In that case there should be used two leg parts on each side in order to prevent bending of the legs. As the acrylic is only 2 mm thick, compared to the 3 mm of the MDF plates, it bends more easily.

As the motors broke in the very last days of the project, I decided not to replace them, but to prepare all so that the next student working on that project can directly start with new components, which he will be sure are working.

Building a low-cost one-legged hopping robot is possible and it is worth continuing on this project. Probably the design has to be changed a bit, but more importantly, the control part has to be improved a lot.

# 6 Further Work

The first step to undertake is to add struts from the foot to the top base in order to restrict these degrees of freedom. Because now it is possible that the top plate is horizontal, but the hip is bent, so that the robot falls directly and cannot correct. When there are struts added, the top plate and the foot will have the same orientation and this orientation can be changed by changing the hip angle. Additionnally, friction between foot and ground will need to be added, that there is really forward motion and not falling down due to slipping. For now the robot just slips away and is not able to push itself from ground.

Most important, the control has to be improved in order to make the robot hop forwards and to control the balance. When the top plate tips over, the legs should move to stabilize the robot.

Later milestones would be to build a 2-legged hopping robot with synchronized legs in order to get rid of the stabilizing walls. Then finally, a two-legged running robot will be built when the two legs can be alternated independently.

And once the robot is running as it should, scientific questions can be investigated. For example one could look on stiffness variations of the leg and the rubber bands or spring in the leg, or on the effect of angle of attack variations. These are only two examples, there are also questions about control strategies to be answered.

# References


[1] "Research on legged machines can lead to the construction of useful legged vehicles and help us to understand legged locomotion in animals." In: *Legged Robots tl of Balance* 29.6 (june 1986).

[2] Honda. *Asimo*. http://asimo.honda.com/. accessed on 13.05.2018. 2018.

[3] Matthew A. Robertson, Jamie Paik, Auke J. Ijspeert and Amy R. Wu. "Rando Walking Robot". In: *DRAPA Challenge, Dynamic Walking 2017* ().

[4] Marc H. Raibert. *Legged Robots That Balance*. Cambridge, Massachusetts: MIT Press, 1986.

[5] William John Schwind. "Spring loaded inverted pendulum running: A plant model". In: *University of Michigan, ProQuest Dissertations Publishing, 1998. 9909991.* (1998).

[6] Avik De and Daniel E. Koditschek. "The Penn Jerboa: A Platform for Exploring Parallel Composition of Templates". In: *2015 IEEE Intl. Conference on Robotics and Automation* (May 2015).

[7] ELEC6212 Biologically Inspired Robotics 2016/17. *Flea Inspired Jumping Robot*. https://www.youtube.com/watch?v=tj5UfTWCB2c. accessed on 27.02.2018. 2017.

[8] Gavin Kenneally, Avik De, and D. E. Koditschek. "Design Principles for a Family of Direct-Drive Legged Robots". In: *IEEE ROBOTICS AND AUTOMATION LETTERS* 1.2 (JULY 2016).

[9] RoboMechanics Lab. *Low-cost minitaur robot*. Dynamic Walking 2018, Pensacola, FL, USA. 2018.

[10] Peter G. Adamczyk, Steven H. Collins, and Arthur D. Kuo. "The advantages of a rolling foot in human walking". In: *The Journal of Experimental Biology* 209 (2006), pp. 3953–3963.

[11] Arduino Store. https://store.arduino.cc/usa/arduino-nano. accessed on 28.05.2018. 2018.

[12] Arduino Playground. https://playground.arduino.cc/Main/MPU-6050. accessed on 28.05.2018. 2018.

[13] Geeetech Wiki. http://www.geeetech.com/wiki/index.php/L298N_Motor_Driver_Board. accessed on 28.05.2018. 2018.

[14] Stan. *Arduino script for MPU-6050 auto-calibration*. http://42bots.com/tutorials/arduino-script-for-mpu-6050-auto-calibration/. accessed on 11.04.2018. 2015.

[15] Probotic. *How to control servo using MPU6050 Gyroscope with Arduino*. https://www.youtube.com/watch?v=vSKEH0FwhUE. accessed on 09.03.2018. 2017.

[16] Roland Pelayo. *How to Build an Arduino Self-Balancing Robot*. https://maker.pro/arduino/projects/build-arduino-self-balancing-robot/. accessed on 11.04.2018. 2017.

# Appendix

## A Robot control script

Code to control the robot, taken from [16] and adapted to my needs.

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

MPU6050 mpu;

#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// Input-Output definitions
int ENA = 5; // has to be a PWM pin
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10; // hat to be a PWN pin

int Switch = 13;

// Initialize some variables used
double IMU_angle = 0;
double Kp = 1;

bool stance = false;            // define state of the leg

int reading = HIGH;

double correction_time = 0;
```

```
double correction = 0;

// ================================================================
// ===                INTERRUPT DETECTION ROUTINE                 ===
// ================================================================
volatile bool mpuInterrupt = false;     // indicates whether MPU
    interrupt pin has gone high
void dmpDataReady() {
  mpuInterrupt = true;
}

// ================================================================
// ===                    Functions to call                       ===
// ================================================================
void motorAup(double t = 50, double spd = 255) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, spd);
  delay(t);
  analogWrite(ENA, LOW);
}

void motorAdown(double t = 50, double spd = 255) {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, spd);
  delay(t);
  analogWrite(ENA, LOW);
}

void motorBup(double t = 50, double spd = 255) {
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, spd);
  delay(t);
  analogWrite(ENB, LOW);
}

void motorBdown(double t = 50, double spd = 255) {
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, spd);
  delay(t);
  analogWrite(ENB, LOW);
}

void bothUP(double t = 50, double spd = 255) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, spd);
  analogWrite(ENB, spd);
  delay(t);
  analogWrite(ENA, LOW);
  analogWrite(ENB, LOW);
}

void bothDOWN(double t = 50, double spd = 255) {
```

```
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, spd);
  analogWrite(ENB, spd);
  delay(t);
  analogWrite(ENA, LOW);
  analogWrite(ENB, LOW);
}

void hipBWD(double t = 50, double spd = 255) {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENA, spd);
  analogWrite(ENB, spd);
  delay(t);
  analogWrite(ENA, LOW);
  analogWrite(ENB, LOW);
}

void hipFWD(double t = 50, double spd = 255) {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENA, spd);
  analogWrite(ENB, spd);
  delay(t);
  analogWrite(ENA, LOW);
  analogWrite(ENB, LOW);
}

// ================================================================
// ===                      INITIAL SETUP                       ===
// ================================================================
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  Serial.begin(115200);
  while (!Serial); // wait for Leonardo enumeration, others continue
   immediately

  // initialize device
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);

  // verify connection
  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful
   ") : F("MPU6050 connection failed"));
```

```
163
  // load and configure the DMP
165  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();
167
  // supply your own gyro offsets here, scaled for min sensitivity
169  mpu.setXGyroOffset(167);
  mpu.setYGyroOffset(6);
171  mpu.setZGyroOffset(-6);
  mpu.setXAccelOffset(903);
173  mpu.setYAccelOffset(-584);
  mpu.setZAccelOffset(641);
175
  pinMode(ENA, OUTPUT);
177  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
179  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
181  pinMode(IN4, OUTPUT);

183  pinMode(Switch, INPUT);

185  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
187    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
189    mpu.setDMPEnabled(true);

191    // enable Arduino interrupt detection
    Serial.println(F("Enabling interrupt detection (Arduino external
   interrupt 0)..."));
193    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady,
   RISING);
    mpuIntStatus = mpu.getIntStatus();
195
    // set our DMP Ready flag so the main loop() function knows it's
   okay to use it
197    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;
199
    // get expected DMP packet size for later comparison
201    packetSize = mpu.dmpGetFIFOPacketSize();

203  } else {
    // ERROR!
205    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
207    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
209    Serial.print(devStatus);
    Serial.println(F(")"));
211  }
}
213
// ================================================================
215 // ===                      LOOP                                ===
// ================================================================
217 void loop() {
  // if programming failed, don't try to do anything
```

```
219   if (!dmpReady) return;

221   // reset interrupt flag and get INT_STATUS byte
      mpuInterrupt = false;
223   mpuIntStatus = mpu.getIntStatus();

225   // get current FIFO count
      fifoCount = mpu.getFIFOCount();
227
      // check for overflow (this should never happen unless our code is too
        inefficient)
229   if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
231     mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));
233
        // otherwise, check for DMP data ready interrupt (this should happen
        frequently)
235   } else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY short
       wait
237     while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

239     // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);
241
        // track FIFO count here in case there is > 1 packet available
243     // (this lets us immediately read more without waiting for an
       interrupt)
        fifoCount -= packetSize;
245
        mpu.dmpGetQuaternion(&q, fifoBuffer);
247     mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
249
        // IMU calculation
251     IMU_angle = ypr[2] * 180 / M_PI;
        Serial.print("IMU_angle:\t");
253     Serial.println(IMU_angle);

255     // Switch
        reading = digitalRead(Switch);
257     if (reading == LOW) { // SWING -> adjust hip angle to a predefined
       value -> PID has not to be done around 0 !!
          correction = ((IMU_angle) * Kp);
259
          if (correction > 3) {
261         correction_time = correction;
            if (correction >= 50) {
263           correction_time = 50;
            }
265         hipBWD(correction_time, 80); // 100,100 is the max -> when it
       was pointing down before, it does not touch the top base then
            delay(20);
267       }

269       if (correction < -3) {
            correction_time = abs(correction);
271         if (correction <= -50) {
```

```
          correction_time = 50;
        }
        hipFWD(correction_time, 80);
        delay(20);
      }
    }

    if (reading == HIGH) { // STANCE -> contract the leg
      bothDOWN(100);// can go up to 150 when the leg was straight before
    (Value with 6 rubber bands -> lower when less rubber band predent)
    }
  }
}
```

Robot.ino