Master Thesis
Master's degree in Automatic Control and Robotics

Design and manufacture of a biped robot to implement the Inverted Pendulum Foot Placement algorithm

Report

Autor: Elliot Vargas Martín
Director/s: Manel Velasco
Call: September 2015

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Summary

This project aims to design and manufacture a bipedal robot specifically designed to be controlled with the *Inverted Pendulum Foot Placement* algorithm. This algorithm, models the robot as an inverted pendulum. This inverted pendulum is formed from the support points of the leg of the robot with the ground, to the center of gravity of the robot. Then using the kinetic and potential energy of the inverted pendulum, the correct position of the point that represent the center of gravity, is calculated so that the robot does not fall to the ground.

The project includes the design and manufacture of the robot's electronics, mechanics and software. But not the implementation of the Inverted *Pendulum Foot Placement* algorithm although, a *Test Algorithm* will be created in order to test that all the mechanics, electronics and that the robot can be controlled by this type of algorithm.

This robot is powered by lithium batteries and it uses six Dynamixel motors to move. The input signals are from an IMU (Inertial Measurement Unit) and the encoder inside the motors. All the electronics are controlled by a microcontroller, but the algorithm is implemented inside of a Microcomputer. Concerning the mechanics, the entire robot parts were designed in 3D models and printed using ABS plastic. There is a second design of the robot that is connected to a PC, as well as, to a power source in order to have infinite autonomy. Also a *Stability Assistance* has been made in order to help the algorithm development.

Contents

List of Figures

List of Tables

Glossary

- **CoG** Center of Gravity
- **IMU** Inertial Measurement Unit
- **ADC** Analog to Digital Converter
- **DCM** Direction Cosine Matrix
- **IPM** Inverted Pendulum Model
- **DOF** Degree of Freedom
- **ZMP** Zero Moment Point
- **GUI** Graphic User Interface
- **EMI** Electromagnetic Interference
- **AC** Alternating Current

Preface

Motivation

Since I can remember I liked everything about the robotics, mechanical devices, etc. All gifts I received during my childhood were related to this topic. When I was a child I liked to disassemble appliances (printers, computers, etc.). All the work I have done during my academic life has been related to robotics. For high school, I manufactured an automatic classification of nuts, made with recycled parts, mainly from the devices that were removed during my childhood.

It was no surprise that for a Master Final Thesis (TFM) I selected to manufacture a robot. At the beginning I was thinking of a quadruped robot, articulated arms ... When I heard about creating a robot to implement the*Inverted Pendulum Foot Placement* I decided that it is the robot that I want to build, mainly because it was something newfangled, different and a challenge for me.

TFM structure

The Figure 0.0.1 illustrates the structure of this TFM starting with a short introduction of the robot, the objective and the reach of this work. Then, the state of the art of the bipedal robot is explained, in the point of view of the different types of control. Finally it is explained the *Inverted Pendulum Foot Placement* algorithm.

Then in Chapters 3 4 5 6 are explained the design of the robot, starting with an introduction and specification of the robot's components. In the following chapters exposing more widely the different parts as it can be seen in the subsections of the Figure 0.0.1.

In Chapter 7 it is described a *Test Algorithm* designed to check if the robot works correctly and if it can be controlled by *Inverted Pendulum Foot Placement* algorithm. Next, the results of this algorithm are shown and discussed. This TFM finishes with the conclusions, future improvements and the budget.

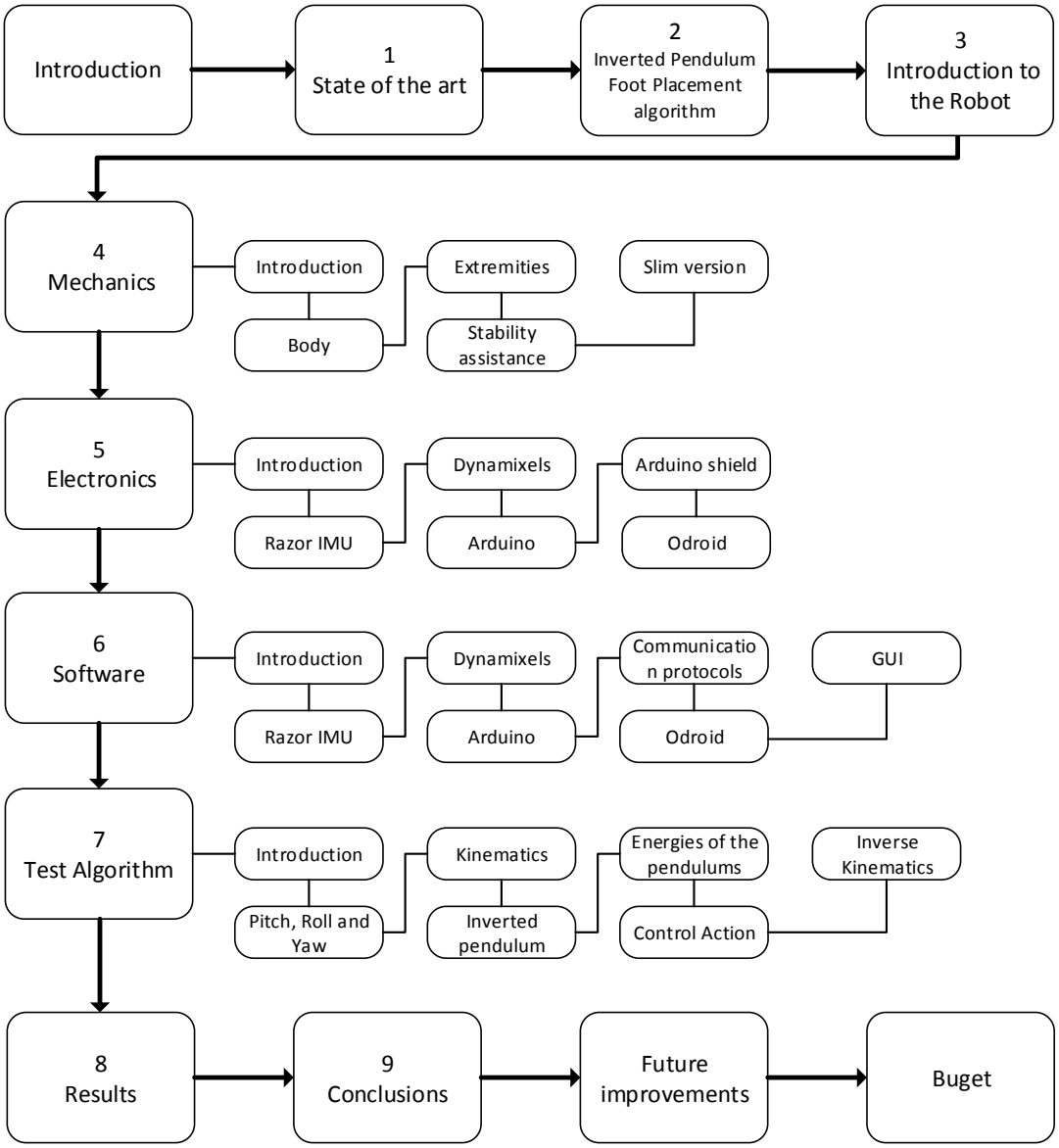

Figure 0.0.1: Structure of the TFM.

Introduction

This project aims to design and manufacture a bipedal robot specifically designed to be controlled with the *Inverted Pendulum Foot Placement* algorithm.

The goal of this algorithm it is to compute the desired stepping point (xd;zd), modeling the robot with an inverted pendulum model (IPM) in order to archive a solution for a lot of body types. The kinetic and potential energy of the IPM Equation 2.0.1 are set as the current state, described by its velocity v and it is height h. then the position of the extremity is calculated, so that the robot does not fall to the ground.

The robot is an autonomous bipedal robot called *Zancobot* which is designed to implement an *Inverted Pendulum Foot Placement* algorithm. To successfully implementation of the algorithm this robot has to resemble as much as possible to two inverted pendulums in 3 dimensions, for this reason the robot will be only designed and built the lower part.

The entire robot parts were deigned in 3d model. The robot is built with plastic, carbon fiber and aluminum. And the plastic parts are printed using ABS and a 3d printer.

The robot is powered by lithium batteries and it uses 6 Dynamixel motors to move. The input signals come from an IMU (Inertial Measurement Unit) and the encoders inside the motors. All the electronics are controlled by a micro controller (Arduino), but the algorithm runs inside of a Microcomputer (Odroid).

The robot has a lot of different electronic parts that can be programmed in different programming languages. The IMU and the Arduino are programmed in Arduino language, and the Odroid is programmed in Python language.

There is a second design of the robot that is connected to a PC, as well as, a power source in order to have infinite autonomy.

In order to test that the robot can be controlled by a *Inverted Pendulum Foot Placement* algorithm, a *Test Algorithm* was created that tests that the robot work correctly. This algorithm is based on the *Inverted Pendulum Foot Placement* algorithm but it is a fast algorithm and will try that the robot stand while being helped by the *Stability Assistance*.

Objective

The objective of this TFM is to create a development platform, specifically created to develop *Inverted Pendulum Foot Placement* algorithms. To do this, the following objectives are needed:

- Design of the Robot:
  - Design of a 3d model of the robot.
  - Design of the electronics of the robot.
  - Development of the software of the robot.
- Construction of the robot:
  - Print of the 3D model pieces.
  - Assembly of all the mechanics.
  - Assembly of all the electronics.
- Creation of a *Test Algorithm* .

Goal

The goal of this project is to design and manufacture a bipedal robot specifically designed to be controlled by the *Inverted Pendulum Foot Placement* Algorithm. That means to create a development platform robot. After this it will be developed a *Test Algorithm* in order to test that the robot can be controlled with this type of algorithm, and test that that the robot's mechanics and electronics work correctly.

Time schedule

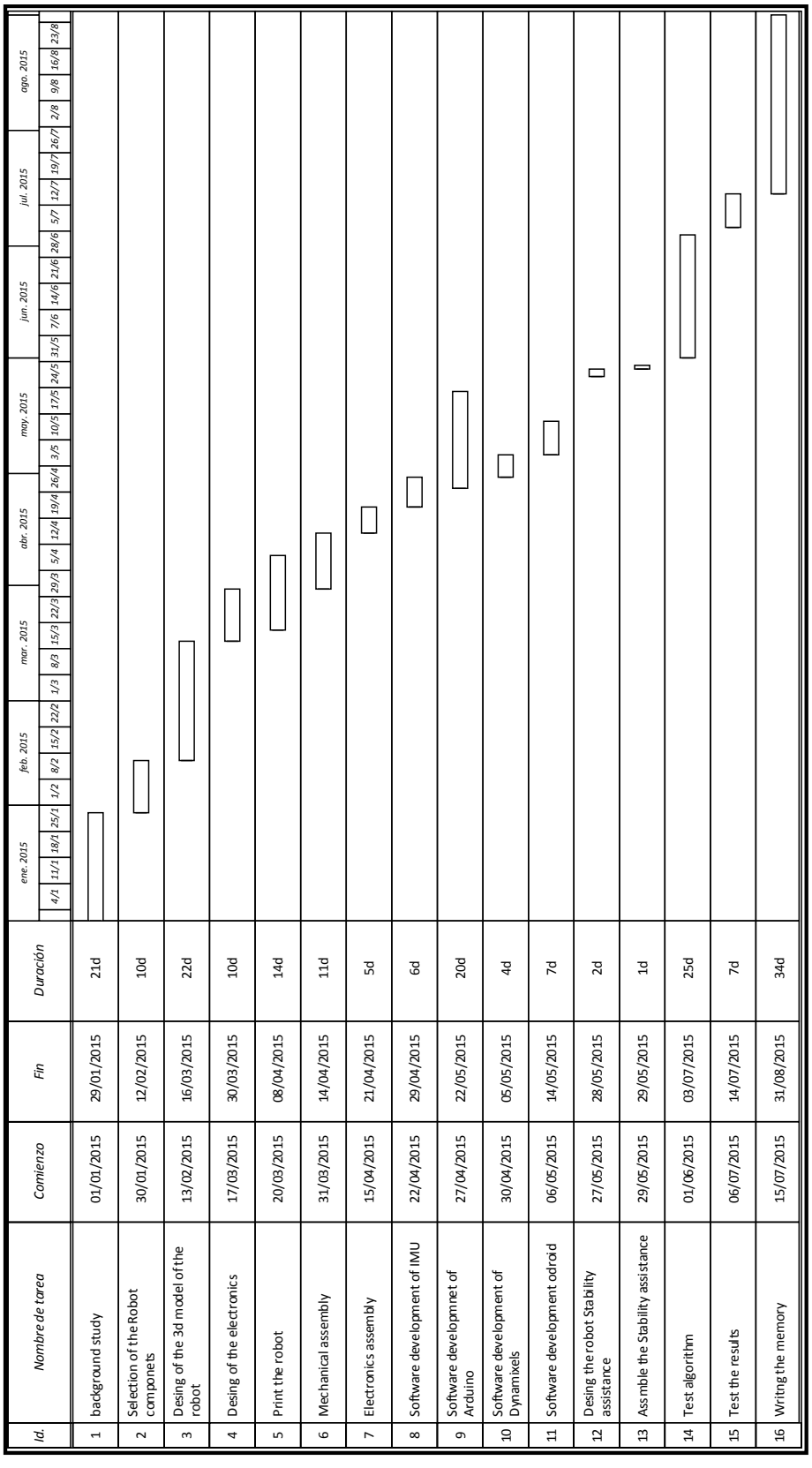

Figure 0.0.2: Gantt diagram of the TFM.

Chapter 1

State of the art

Regarding the construction of a robot there are many parameters that define the robot parameters as the control algorithm used, actuators, sensors and building materials.

1.1 Control Algorithm

There are many proposed ways to define stability for a bipedal walking robot. Here we will refer to eigenvalues of Poincare return maps [12], the Zero Moment Point criterion [15], and change of angular momentum [3, 18].

1.1.1 Eigenvalues of Poincare Maps

In mathematics, particularly in dynamical systems, a first recurrence map or Poincaré map, named after Henri Poincaré, is the intersection of a periodic orbit in the state space of a continuous dynamical system with a certain lower-dimensional subspace, called the Poincaré section, transversal to the flow of the system. Deviations from a periodic limit cycle will return to the cycle and for small deviations, typically follow a linear relation,

$$X_{n+1} = KX_n \tag{1.1.1}$$

Where X is the vector of deviations from the fixed point that the limit cycle passes through and K is a linear return matrix. One of the eigenvalues of K will equal to 1. If the magnitude of the remaining eigenvalues are all less than one, then the limit cycle is stable.

Measuring the eigenvalues of Poincare return maps is commonly used for analyzing Passive Dynamic Walking robots [14, 6, 23] and was used by Miura and Shimoyama [16] to analyse their Biped robots.

However, using this technique assumes periodicity and is valid only for small deviations from a limit cycle. While humans, and hence the bipedal systems trying to mimic the human-like walking, seem to be periodic, there is nothing about the bipedal walking problem that requires periodicity. For example, when a biped walks over discontinuous rough terrain, or when abruptly changes speed or direction, its motion is not periodic. Therefore, while eigenvalue magnitudes of Poincare return maps may be sufficient for analyzing periodic bipedal walking, they are not sufficient for analyzing non-periodic motions, and are not necessary for analyzing bipedal walking in general.

1.1.2 Zero Moment Point

All of the biped mechanism joints are powered and directly controllable except for the contact between the foot and the ground, which can be considered as an additional passive DOF, where the interaction of the mechanism and environment takes place.

The foot cannot be controlled directly but it can be controlled in an indirect way, by ensuring the appropriate dynamics of the mechanism above the foot. Thus, the overall indicator of the mechanism behaviour is the point where the influence of all forces acting on the mechanism, can be replaced by one single force. This point was termed the *Zero Moment Point*. [15].

The *ZMP* when used in control algorithms synthesis for bipedal walking robots typically is computed analytically based upon desired trajectories of the robot's joints. As long as the *ZMP* lies strictly inside the support polygon of the foot, then these desired trajectories are dynamically feasible. During playback of the desired joint trajectories, the actual *ZMP* is measured from force sensors in the foot or by observing accelerations of all the joints [9]. Then deviations between the precomputed and actual *ZMP* are typically used to modify the joint trajectories [8, 26].

The *ZMP stability margin* is the distance from the ZMP to the nearest edge of the convex hull of the support polygon. Typically it measures the room of error for achieving the desired trajectories of the robot. The *ZMP criterion* can be stated as: *Given desired state variable trajectories that are consistent with the dynamics and that predicts the ZMP staying inside the support polygon, a trajectory tracking control algorithm can stably track those trajectories as long as the ZMP does indeed remains inside the support polygon* .

1.1.3 Angular Momentum

From the area of Biomechanics there are some researchers that observed that humans appear to regulate angular momentum about the Center of Mass when standing, walking and running [3, 18]. Their suggestion is that the angular momentum about the Center of Mass (also referred as spin angular momentum) of a biped should be minimized throughout a motion. They also argue that the spin angular momentum should be used when is needed to balance or recover from a push.

Pratt and Tedrake [19] argue that minimizing spin angular momentum is neither necessary or sufficient condition for stable walking. However, they suggest that a biped should *reserve* the limited (by joint angle limits, joint speed limits and joint power limits) spin angular momentum can be used to help recover from a push or other disturbance, as there is a coupling between angular momentum rate change and linear momentum rate change, and hence the speed.

1.2 Actuators

In robotics there are used different actuator power pneumatic, hydraulic, or electronic signals. There are several types of actuators in robotic.

- Synchronous Actuator : The rotor of the motor rotates in synchrony with the oscillating field.
  - Brushless DC Servo : In this motor the rotor is a permanent magnet, and the rotor generate a rotating poles of the opposite magnetic polarity controlled by electronically controlled commutation system creating torque. The advantages are: higher efficiency and reliability, reduced noise, longer life time, low EMI.
  - Stepper : A type of brushless servo motor, moves or rotates in small discrete steps. The advantages are that the maximum dynamic torque occurs at low pulse rates. Drawbacks low efficiency; high heat dissipation energy.
  - Brushed DC Servo : The motor generates an oscillating current in a rotor with a split ring commutator, and either a permanent magnet stator. The advantage to using a brushed motor over a brushless is the cost.
- Asynchronous Actuator : The rotor of the motor doesn't rotate in synchronous with the oscillating field.
  - AC Servo Motors : The motor is powered by Electrical AC signal. The stationary stator has coils supplied with alternating current to produce a rotating magnetic field, and the rotor produces a second rotating magnetic field.

- Pneumatic : Uses the compressed air to pressure force. They are compact and light actuators, but they are less energy efficient than electric motors and need an air compressor.
- Hydraulic : Converts hydraulic pressure and flow into torque and rotation, they can generate a large amount of force but they are expensive, they need an hydraulic generator.

1.3 Construction materials

The robot can be made of many materials, but the most used materials are those with good strength to weight ratio. The most used materials are plastic, aluminum, carbon fiber, titanium and other metal alloys. Some examples can be seen in the Figure1.3.1.

Some robots use printed plastic instead of injected plastic because although it have worse properties the manufacturing cost is much lower and it allows many more forms. One example it is the Poppy robot, Figure 1.3.1.

Figure 1.3.1: Darwing(left), Atlas (central) and Poppy(left) Robot.

1.4 Sensors

There are a lot of different sensors that can be used in robotics. The most common are IMU, encoder and force sensors in the rotation joints, sensor pressure or contact sensor in the feet, range sensor, stereo vision and microphones. The use or not of specifics sensors determines the design of the robot.

Figure 1.4.1: Some example of the sensors in robotics.

For example the utilization of the range sensor to detect the space need that the robot use wheels to be enough stable. the pressure sensors need that the foot have a special shape. Normally the utilization of stereo version need a mobile platform to orientate the camera, for dis reason are usually situated in the head of the robot.

Chapter 2

Inverted Pendulum Foot Placement algorithm

In a robot walking the position where to place its feet is very important in order to not fall down. Depending on the support point, the robot movement can vary a lot. Ranging for a minimal distance can be a good choice for a idle stance, but for a fast walking are needed large steps in order to have large push.

The target of this algorithm is to compute the desired stepping point (x_d, z_d), modeling the robot with Inverted Pendulum Model (IPM) in order to achieve a solution for a lot of body types.

The kinetic and potential energy of the IPM Equation 2.0.1 are set as the current state, described by its velocity v and its height h.

$$KineticEnergy = \frac{1}{2}mv^2 \qquad PotentialEnergy = mgh \tag{2.0.1}$$

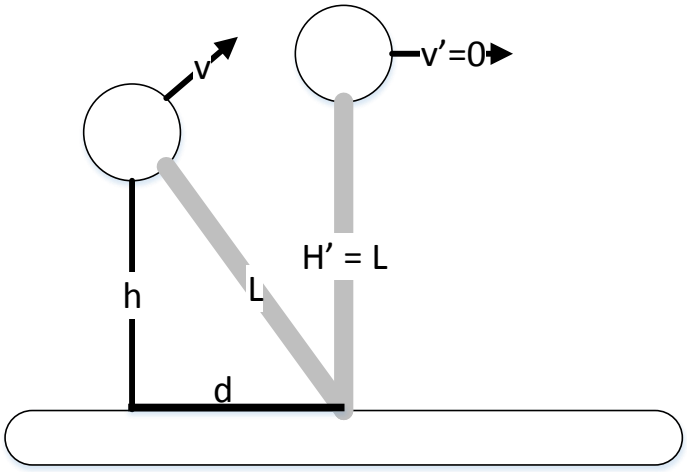

Figure 2.0.1: Inverted pendulum model.

The future support point is balanced with the current state Equation 2.0.2 in order to in the future state, the robot has 0 velocity and vertical position.

$$\frac{1}{2}mv^2 + mgh = \frac{1}{2}mv'^2 + mgh'v' = 0 \qquad h' = L = \sqrt{h^2 + d^2} \tag{2.0.2}$$

Solving this relation, d has the form of the Equation 2.0.3. The value of d is desired to have zero velocity in the next step.

$$d = v\sqrt{\frac{h}{g} + \frac{v^2}{4g^2}} \tag{2.0.3}$$

But taking a minor value of d will make a short step that will achieve a positive, and higher value of d will make a short step that will achieve a positive velocity, so computing the Equation 2.0.4 where V_d is the magnitude of the desired velocity and α is a constant. This V_d can be used in order to move the robot into a desired position.

$$d' = d - \alpha V_d \tag{2.0.4}$$

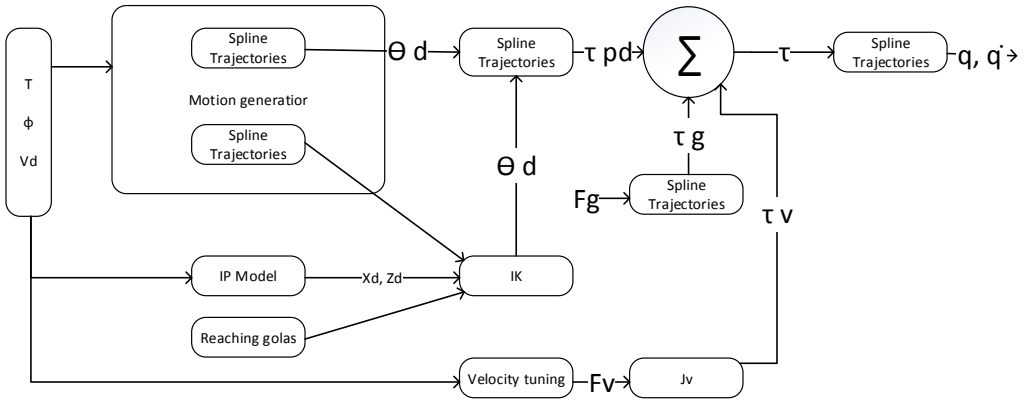

Figure 2.0.2: Implementation of the Inverted Pendulum Foot Placement algorithm.

According to the article [22]. This algorithm can be implemented in a walking robot controller using the controller of the Figure 2.0.2.

Chapter 3

Introduction to the Robot

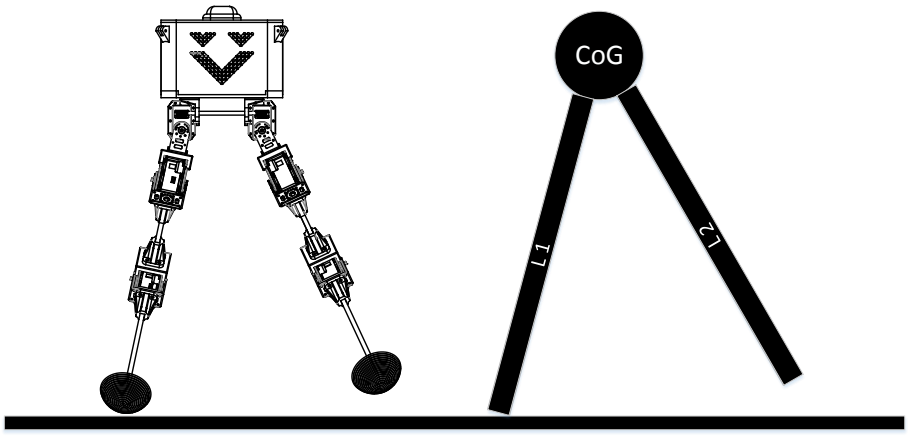

Figure 3.0.1: Two inverted extensible pendulums model.

The robot is an autonomous bipedal robot called "Zancobot" which is designed to implement an *Inverted Pendulum Foot Placement* algorithm. To successfully implementation of the algorithm this robot has to resemble as much as possible to two inverted pendulums in 3 dimensions as in Figure 3.0.1, for this reason the robot will be only designed and built the lower part.

This section of the project it's going to be a brief explanation about the different parts and a small description of the main components of the robot. An extended explanation of the different parts of the robot, how it works and the reason why we have chosen the different components will be made in the chapters 4, 5 and 6.

The robot is a biped robot, that only has 2 lower extremities, those extremities attempt to be as similar as possible to two extensible inverted pendulum. Therefore the electronics

will be located in the robot's body.

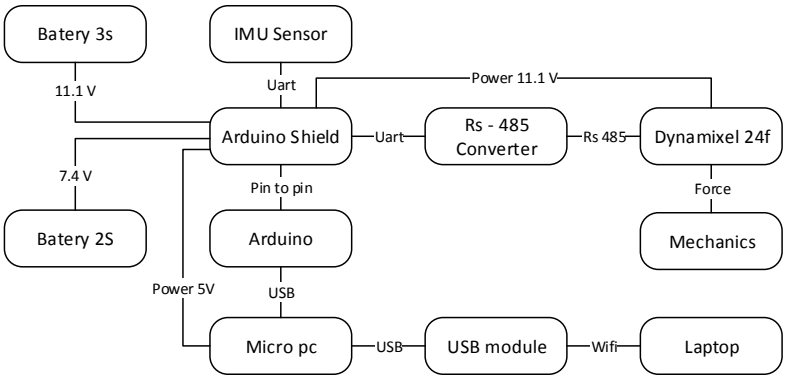

Figure 3.0.2: Schematics of the Robot.

As it can be seen in the Figure 3.0.2 the robot's engine are rotational motors, in particular the Dynamixel RX-24F, these motors are powered by a 3-cell lithium battery. The rest of the electronics are powered by another 2-cell lithium battery. The robot has 2 batteries because the Dynamixel has a huge consumption and that can make that battery's drop a lot and it can cause a lot of electronics's errors.

The electronics of the robot are based in a microcomputer as a center computational cost of the algorithm helped by Arduino an Arduino shield, which is in charge of the power management, it makes the communication between all the components and checks the correct performance of all the electronics.

The sensors of this robot are an inertial measurement unit (IMU) and encoders in all of the robot's 6 engines. Also the temperature of the motors, the voltage of the two batteries and the voltage of the supply power of the microcomputer.

The main electronics of the robot are: one micro PC for the calculation and one micro controller for the management of the electronics.

The interaction with the robot will be made via Wi-Fi using an USB Wi-Fi Adaptor. The robot will create a Wi-Fi network and the user will be able to connect it with any device with Linux, Android or Windows.

After some considerations it was decided to design two configurations of the robot: the standard one and the slim version. The main difference between the two configurations is that the standard has all the electronics inside of the robot, this means that it is autonomous but consequentially is heavier than the slim configuration. This one has only the motors and the IMU inside the robot, and the rest outside of the robot, in this case robot is lighter. The slim configuration will be explained below.

Chapter 4

Design and manufacture of the mechanics of the robot

4.1 Introduction

In this Chapter is going to be explained the mechanics of the robot, the different parts of the robot Figure 4.1.1, the construction materials, the CoG, the different parts of robot and finally the stability assistant.

The first think try to replicate two extensible inverted pendulums can be used linear motors to implement an extensible linear legs, but for this type of motors is difficult to implement because it is heavy and slower, for this reason it has decided to use only rotational motors, the Dynamixels RX-24f.

The robot needs to have a good weight/strength ratio. To do that the robot will be built with plastic, carbon fiber and aluminum, and the design has to be the most simply as possible.

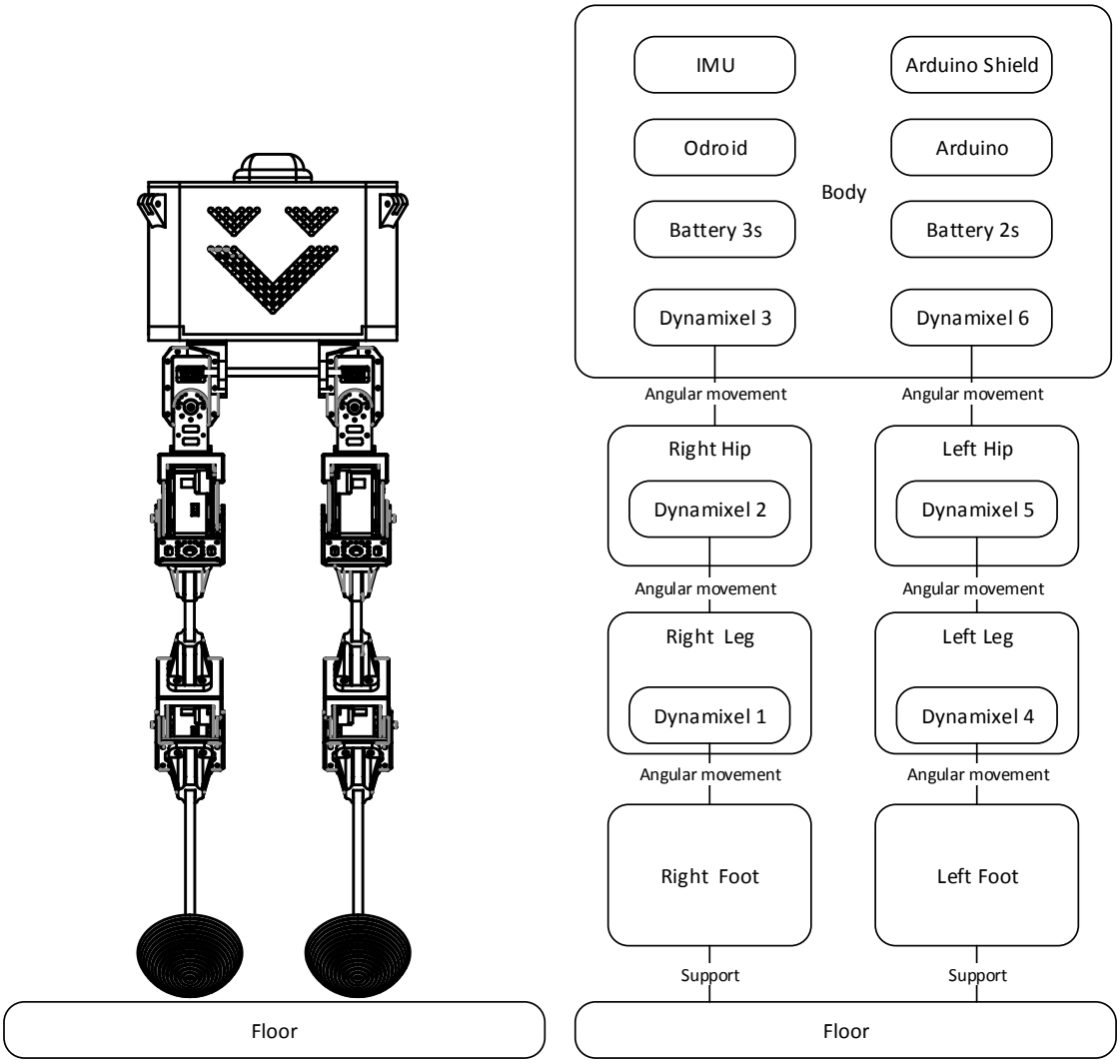

Figure 4.1.1: Schematics of the mechanic of the robot.

The robot is divided in seven parts as it can be seen in the Figure 4.1.1. All these parts are connected by the Dynamixels, it means that it can change this configuration with rotational moments. The most important part is the body and also the heaviest part too because it has inside the batteries, the IMU and all the electronics. The rest of parts are those ones that compose the two extremities: there are two hips, two legs and two foots.

Figure 4.1.2: Rendering of the front part of the robot.

The 3D model is designed with the 3D editor software SolidWorks. The 3D model Figure 4.1.2 and Figure 4.1.3 have all the parts of the robot, the 3D model of the bought parts as Dynamixels, Dynamixel frame, Odroid and Arduino were extracted from sellers website, and others like the batteries are modeled with a fast model. The rest of parts are modeled in order to be printed and make the robot fit correctly.

The robot has been made with a 3D printer, a printer designed and built by the author which is based in a Prusa Iteration 3. This printer extrude the parts with ABS plastic which it is a good material of construction because it has low density and enough strength for this purpose, these parts will be joint with nuts and hexagonal head screws in order to make sure

that the pieces are fixed correctly. Also some of these pieces are treated with steam bath of acetone in order to increase the cohesion between the layers of the printed parts.

The frames of the Dynamixels are built with aluminium. The square bars that connect the two parts of the leg and the foot are built with carbon fiber because are extremely light and at the same time are strong, the carbon and aluminum are used for these parts because they are the more stressful pieces and they need to be more stronger and less flexible than the printed parts.

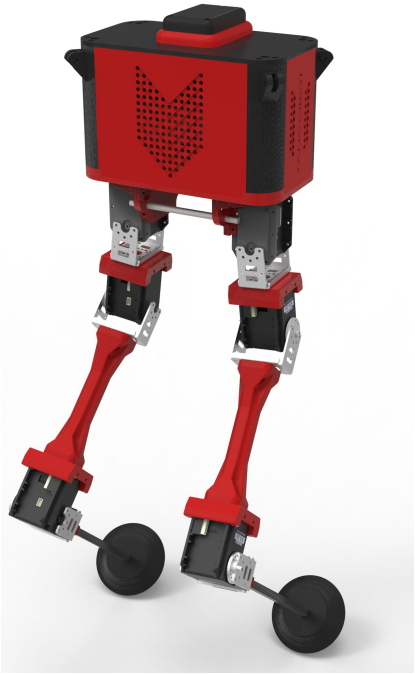

Figure 4.1.3: Rendering of the back part of the robot.

After the print of all pieces, they are weighted separately in order to find the center of gravity of each fixed part. As we can see in the Table 4.1.1 the total weight for standard version of the robot is 1338.64 grams and 753.64 grams for the slim version. We can see that the slim configuration is 33 percent more lighter than the standard configuration. This low weight can make that in the future will be more easy to implement the control algorithm in the slim version than in the standard one.

Other important parameter to know about the robot is the location of the CoG of the robot, the CoG changes with the configuration of the robot, but the CoG of each rigid part of the robot (body, hips, legs and foots) can be extracted. To extract this CoG the weight of all pieces is introduced into the 3D model of the robot, then this 3D editor is able to calculate it numerically. The values can be seen in the Table 4.1.1. The CoG of the robot depends on the configuration of the engine and the orientation of the robot will be explained in the Chapter 7.

Parts	Weight (g)	Cog X Axis (mm)	Cog Y Axis) (mm)	Cog Z Axis (mm)
Body	815	-1.89	0	-75.42
Body (Slim)	365	-1.50	0	-38.47
Hips	94.5	0	0	21.91
Legs	120	0	0	54.74
Foots	65	0	0	60.2
Total	1374	—	—	—
Total (Slim)	924	—	—	—

Table 4.1.1: Table of Weight and CoG of the different parts of the robot

4.2 Body

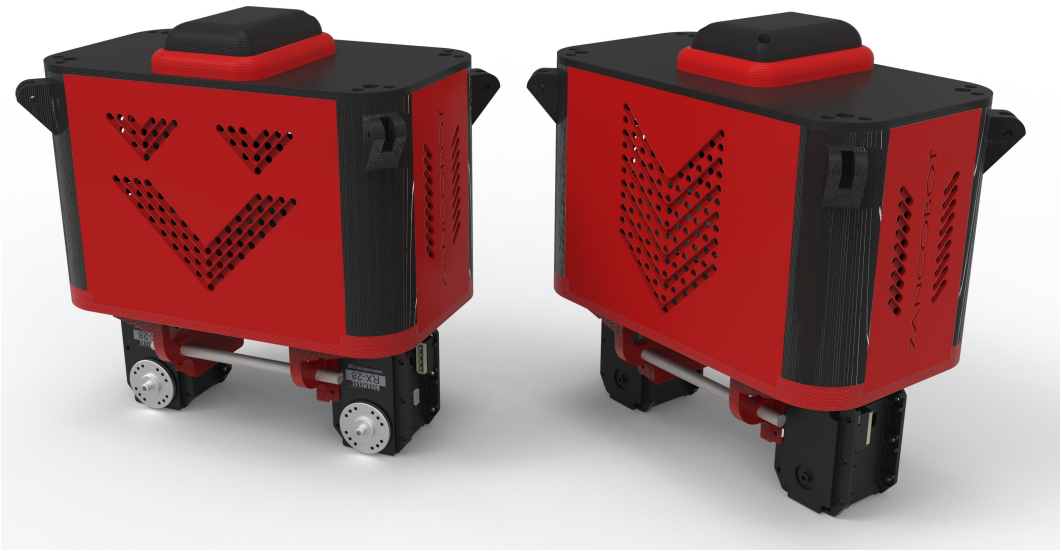

Figure 4.2.1: Chasis of the robot.

The body is the part of the robot that has the function of carry all the electronics except four of the motors. This part is designed with interchangeable parts in order to increase and reduce the shape of the robot, depending in the fitting of the different possible electronics.

The bottom part of this piece is the most robust because it carries all the stress of the extremities; to make it strong it was used an M5 threaded rod that link the motors 3 and 6. The upper part is built with two flat peaces into the top and with four columns in to the corners, this configuration maximises the internal capacity to make the body more light

and small. The four columns have a rope grabs that can be used to help to walk the robot connecting it to the *Stability Assistance* .

Other important factor is that the robot has been designed with the possibility to change the distances between the extremities, this fact makes that the model of the robot can be changed with the required configuration, it is possible because the body have two rails where the Motors 3 and 6 can be moved along them, to move the motor's position the only thing that is needed do it's to loose two screws and two nuts per engine.

The electronics inside the body are distributed by weight, the most heavy are the two batteries which fit in the lower part, them the Odroid, the Arduino and the Arduino Shield are put above the batteries fixed with a plastic piece.

The IMU sensor is located on the top of the robot in order to increase the accuracy in the lectures.

4.3 Extremities

The extremities are composed by three parts: the foot, Figure 4.3.1, the leg, Figure 4.3.2 and the hip, Figure 4.3.3

The robot have two extremities with three actuated rotation joints each one, which make that each extremity have 3 DoF, that makes possible to situate the tip of the leg in any possible position in 3D inside the work space; but the angular position of the tip only has two maximum possible locations.

The motor of the joints will be located in the previous joint in order to make the robot the robot and make an easy future calculation of the robot dynamics.

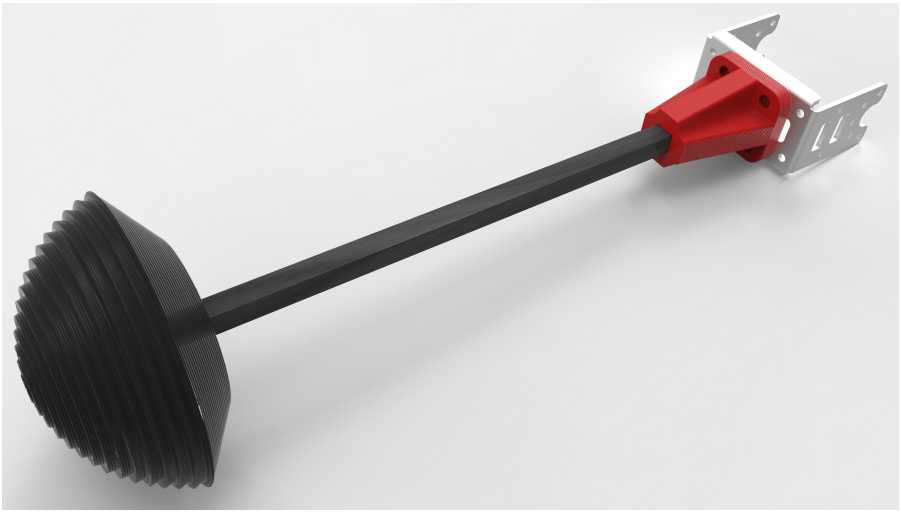

Figure 4.3.1: Foot of the robot.

The foot seen in the Figure 4.3.1 is the lower part of the extremity, the optimum shape of the stump should be a point in order to help to calculate where is it the contact point of the robot leg with the ground, but the problem is that the grip of a tip is very low, for this reason the stump of the robot have a semi-spherical shape to increase the grip and also to help to measure the contact point, as can be seen the stump have grooves in order to increase the grip between the robot and the floor of the Stability Assistant which is make with liquid silicone and soft mat.

Figure 4.3.2: Leg of the robot.

The leg part of the extremity, Figure 4.3.2 connects the foot with the hip, it has in one endpoint a frame that will be connected with the Dynamixel of the hip and in the other side has a Dynamixel that will connect to the foot and will be able to move it. The two endpoints will be connected with a rectangular carbon fiber bar with a cavity inside in order to wire the Dynamixel.

The last part of the extremity of the robot is the hip, Figure 4.3.3 it has the minimum possible distance to the leg in order to try to be the most similar to a inverted pendulum.

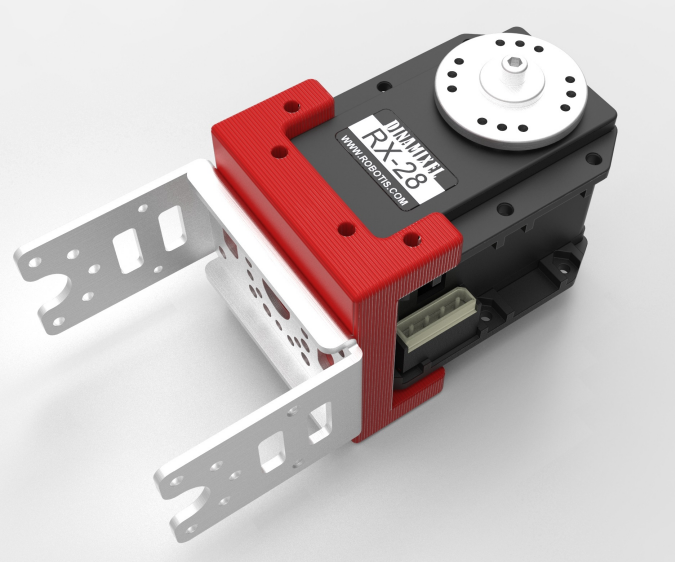

Figure 4.3.3: Hip of the robot.

4.4 Stability Assistant

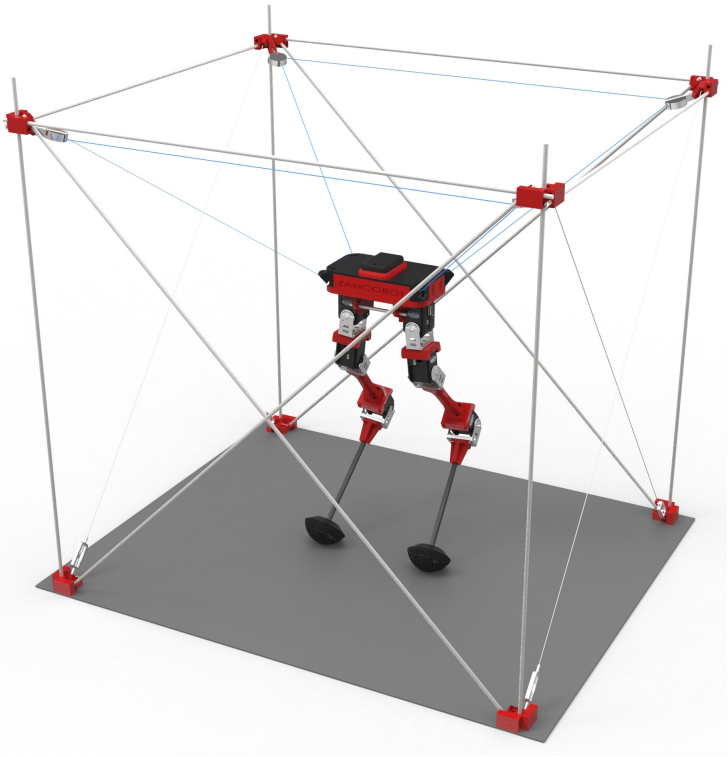

Figure 4.4.1: Stability assistance.

In order to help at the development of the robot, the Stability Assistant has been created, Figure 4.4.1, this device has different functions. One of them is to anchor the robot when it falls down, also it creates a good grip surface using a liquid silicone and soft mat to remove the possible slide. With a rope and the pulleys of the box it is possible to transform the 3-dimensional problems to a near 2-dimensional problem, this simplification will be explained in the Chapter 7.

The Stability assistant have the dimensions of (800 , 800, 650) millimeter width, height and depth respectively, leaving a useful workspace of (700 ,700, 550) millimeter which is enough for the robot to walk.

This device is very similar to a cubic box, the base is a wooden panel, the vertex of the box are 3D printed pieces which are designed to connect all the parts, the edges of the box are M5 threaded rod; a box with bars in the edges don't have the required rigidness, so, for this reason an auxiliary bar has been added in the front and rear of the box and wired with tensor in the laterals. The final result is that the *Stability Assistance* is light and enough rigidness in the top vertexes, which are the subjection points of the rope that hold the robot.

4.5 Slim version

Figure 4.5.1: Chassis of the slim version.

The slim configuration is a lighter version where are moved out all possible parts of the robot, which are the two batteries, the Odroid, the Arduino and Arduino shield, it means that all the electronics except the IMU and the motors has been removed. After take off all of this electronics the chassis of the robot can be reduced as it can be seen in the Figure 4.5.1.

The removal of all of these pieces have the direct consequence that the weigh of the robot has a reduction of the 33 percent of the robot's weight, other consequence is that the extraction of the batteries enable that the robot can be powered by power supply, this fact make easier the continued use of the robot.

All the Python code that runs in the Odroid can be used in a PC, this make easy and fast to turn on the robot, and the data collection.

For this version it will be do a GUI in order to see all the data in real time.

Figure 4.5.2: Slim version of the robot.

Chapter 5

Design and manufacture of the electronics of the robot

5.1 Introduction

In this robot as in all robots is very important how will be the responsibility of: move the robot, collect information from the world, to compute the algorithm, the handling of the batteries and the communication of all the electronics. Is very important to assure the correct section of the electronics part to make the robot being full functional.

The first important thing before to start with the design of the robot's electronics is to estimate the computational cost of the algorithm, and decide where this algorithm will be running and the programming language, after a some estimations the results show that the computational cost is high because the system have to handle the computational of a lot of matrices involved in the calculation of the CoG...

After some searching it was considered two possible options. The first one it was to run it in a 32 bits ARM microcontroller configuration in C language. The second one was to run it in a microcomputer like Raspberry Pi or Odroid in Python language. The second option was finally considered because it has more computational potential and it has more libraries with the use of a microcomputer.

The use of a microcomputer with Linux as an operation system have very goods features like: fast configure, amount of library's and documentation, etc. But a Linux system has the problem that the time is not precise as it is required, for this reason the microcomputer does not control the motors directly and it reads the values of the sensors, this will be managed with a board with a 8 bit microcontroller (Arduino Mega R3) which will handle all the information with very accurate timing and will be communicated with the microprocessor via UART-USB.

In the following sections it is gone to be explained which and why components are used and their specifications.

5.2 Inertial Measurement Unit (IMU)

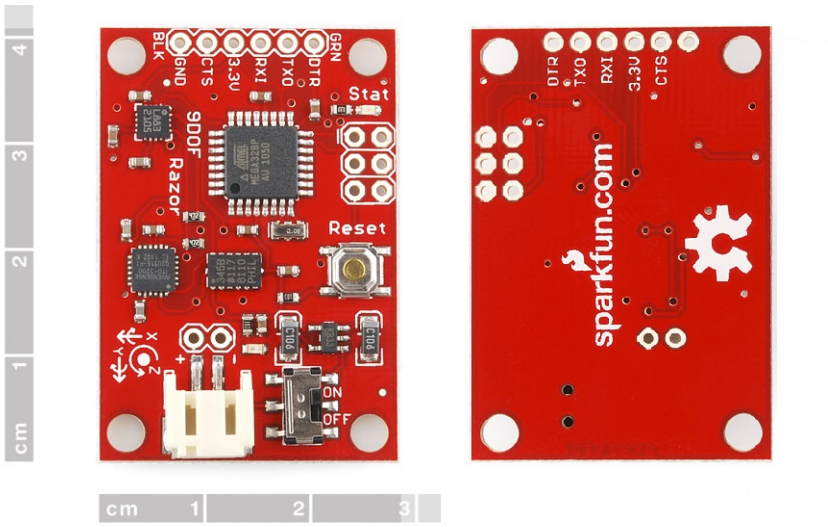

Figure 5.2.1: Razor 9 DOF.

This algorithm need as an input information the angular position and the angular velocity of the inverted pendulums. It means that is needed the configuration of the robot and its orientation; to extract the last one it is used an IMU the 9DOF Razor IMU.

The 9DOF Razor IMU is a board that incorporates three sensors: an ITG-3200 (MEMS triple-axis gyro), an ADXL345 (triple-axis accelerometer), and a HMC5883L (triple-axis magnetometer). To give at the user nine degrees of inertial measurement. The outputs of all sensors are processed by an on-board ATmega328 and output over a serial interface.

The good point of this IMU it's that incorporates a 8 bit microcontroller, the ATmega328, this means that a filter to process the signal can be programed in order to extract directly the output filtered.

The board comes programmed with the 8 MHz Arduino bootloader (stk500v1) it mean that it can be programmed simply by connecting to the serial TX and RX pins with a 3.3V FTDI Basic Breakout, and also can be used the Arduino IDE to program own filter to process the signal.

5.3 Dynamixels

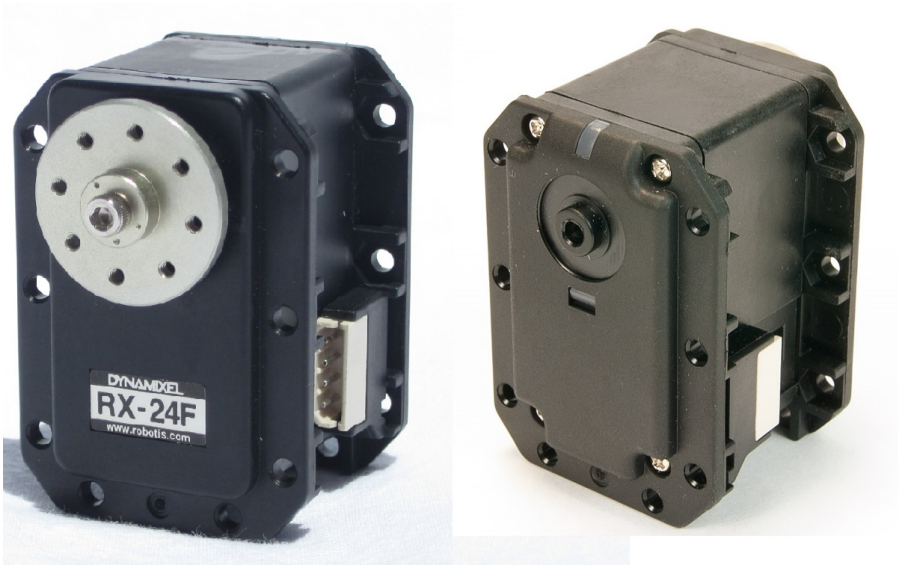

Figure 5.3.1: Dynamixel RX-24f.

For this robot will be used a Dynamixel motor because it is a line-up high performance networked actuator. This motor are manufactured by the Korean company ROBOTIS. It incorporates all features needed to move and control the robot: gears, DC motor, DC driver, encoders and an 8-bit microcontroller.

The microcontroller receives the action control from serial communication that can vary depending on the model, there are a lot of possible actions but for this robot it will used the position, then the microcontroller with a PID controller will control the position of the robot.

This motor have another features and also can be read other information's like current temperature, voltage, strength (not accurate)...

After design the mechanical parts of the robot the motors have to have at least 2 Nm and 60 RPM, as we see there are a lot of possible models to chose as is showed in the Figure 5.3.2, after some consideration the RX-24F is the choosen one because it have enough stall torque, good speed and good relation price/cost.The specifications of RX-24F Dynamixel are showed in the Table 5.3.1.

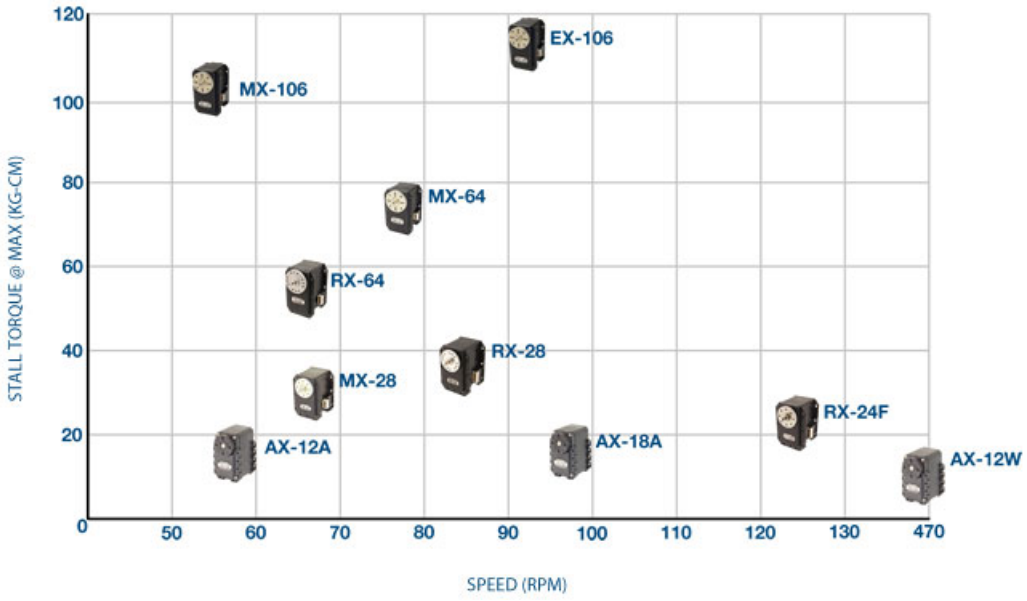

Figure 5.3.2: Dynamixel comparison.

Concept	**Specifications**
Weight	67g
Dimension	35.6mm x 50.6mm x 35.5mm
Resolution	0.29 C
Gear Reduction Ratio	193 : 1
Stall Torque	2.6N.m (at 12V, 2.4A)
No load speed	126rpm (at 12.V)
Running Degree6	0° 300°
	Endless Turn
Running Temperature	-5° +80°
Voltage	9V 12V (Recommended Voltage 11.1V)
Command Signal	Digital Packet
Protocol Type	RS485 Asynchronous Serial Communication (8bit,1stop, No Parity)
Link (Physical)	RS485 Multi Drop Bus
ID	254 ID (0 253)
Communication Speed	7843bps 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Material	Full Metal Gear, Engineering Plastic Body
Standby current	50 mA

Table 5.3.1: Dynamixel 24f Specifications

5.4 Arduino

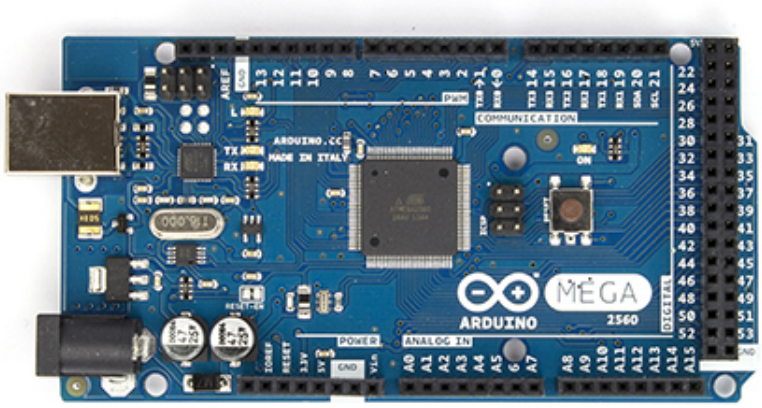

Figure 5.4.1: ArduinoMega2560 R3.

To control and read the IMU and the motors with accurate timing it is used and 8-bit microcontroller because to do this actions it is needed a low computational cost but it has to be managed a lot of serial communications, with small data but a lot of time every seconds.

It was decide to employ The Arduino Mega 2560 because it is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (15 of them can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button as can be seen in the Table 5.4.1. It contains everything needed to fulfill the needed specifications.

There are different reasons to choose it in front to use other 8-bit microcontroller. The main reason is that it is no need to print the PCB and mount it because the manufacturer seal it with the PCB mounted, the other good thing is that the integrated development environment (IDE), has a fast implementation.

Concept	Specifications
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Table 5.4.1: Arduino Mega Specifications.

5.5 Arduino shield

In order to connect all the electronics to the Arduino, has been designed and constructed a board, also called Arduino shield in order to physically connect it with all the robot electronics, or an additional electronics needed too.

The main functions of the Arduino Shield are:

1. Power the IMU(3.3V), the Dynamixels(11.1V), the Odroid (5V), the integrate MAX485 (5V) and the Arduino (5-20V).
2. Communicate the IMU with the Arduino.
3. Communicate the Dynamixels with the Arduino.
4. Transform the Voltage of the Battery 2s of range (6.4V - 8.2V) to (3.2V - 4.1V) and sent to Arduino.
5. Transform the Voltage of the Battery 3s of range (9.6V - 12.4V) to (3.2V - 4.1V) and sent to Arduino.
6. Transform the Odroid supply voltage of range (4V - 6V) to (2V - 3V) and sent to Arduino.
7. Show with led, the state of the batteries (Full, Low, Empty) and the robot state (On, Off and Error).

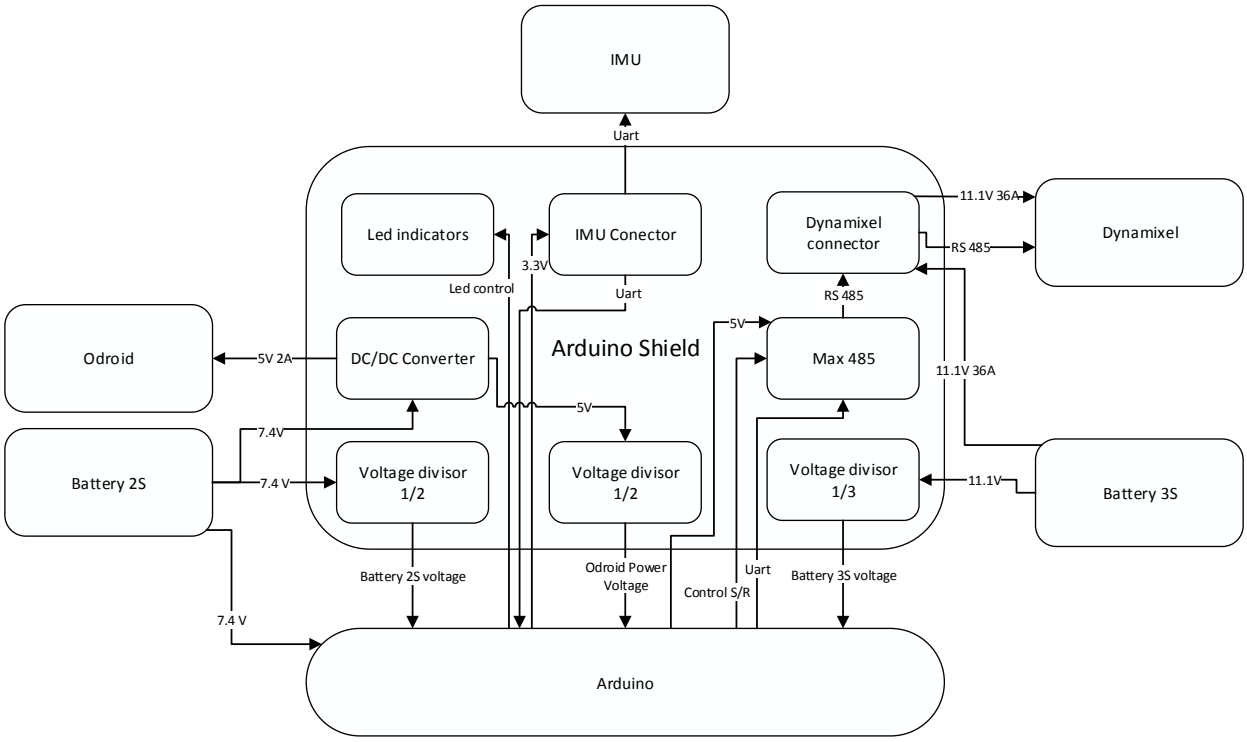

Figure 5.5.1: Schematic of Arduino shield.

The electric power of the Battery 3S is directly connected to the Dynamixels connector with no any transformation because the voltage of the battery is the optimum for the Dynamixels, and because the maximum power supplied to the motors is 446.4 W and to transform this amount of energy it will required a huge electronics and probably a refrigeration system. All other electronics are powered with the other battery, the Battery 2S have a nominal voltage of 7.4 V and the Odroid need a supply voltage of 5V and 2A, to convert the voltage it is used the integrate circuit LT1084CP-5, the battery also powers directly the Arduino; although the Arduino works at 5V, it as a 7805 voltage regulator which allows to connect the battery directly to the Arduino, the other electronics are supplied by the Arduino. The Arduino can be supplied using the battery or the USB connector, this allows to power the electronics using a computer instead the Battery 2S, this powering method will be used in the slim configuration.

The communication between the Arduino and the IMU is the UART communication so it is not needed additional electronics and they no need to be connected directly. For the communication with the Dynamixels it has to be have in mind that the physical link is a RS-485 Multi Drop Bus, and the Arduino MEGA 2560 R3 do not have this type of lins, to fix this it is used the chip MAX485 to convert UART to RS-485.

The lithium battery have a lot of great features as: the good relation weight/capacity and volume/capacity and a high electric power supply. But it has very sensitive operation

temperature and voltage, if it has high voltage it can burn and can get lost if the voltage is too low, to take care of the battery it is sensed, but the voltage of the battery is too high for be read with the ADC of the Arduino, which have a range of 0V - 5V. To fix this issue the exit voltage is reduced with a simply voltage divider. The supply voltage of the Odroid also is sensed in order to assure the correct operation voltage, because the Odroid is very sensible to the Voltage drop.

The voltage of the batteries is displayed by 6 LED, 3 of them for each battery, the green LED means that the battery is between 33 and 100 percent, the yellow LED turns on when the battery is between 33 and 0 percent and the red one shows when the battery is empty. When the battery is empty the Arduino will stop the motors, there are also 2 LED more, the green one indicates when the Arduino is on and the red one when there is an error.

The electric schematic of the Arduino Shield can be seen in the Figure 5.5.2 and its PCB in the Figures 5.5.3, 5.5.4, 5.5.5 and 5.5.6.

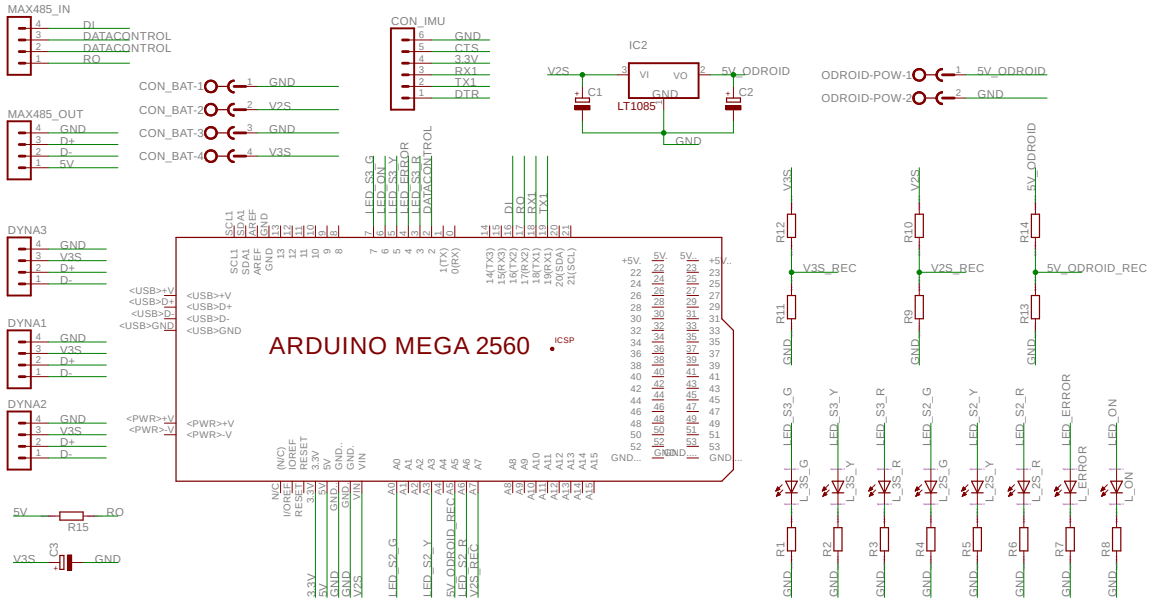

Figure 5.5.2: Electric schematic of Arduino shield.

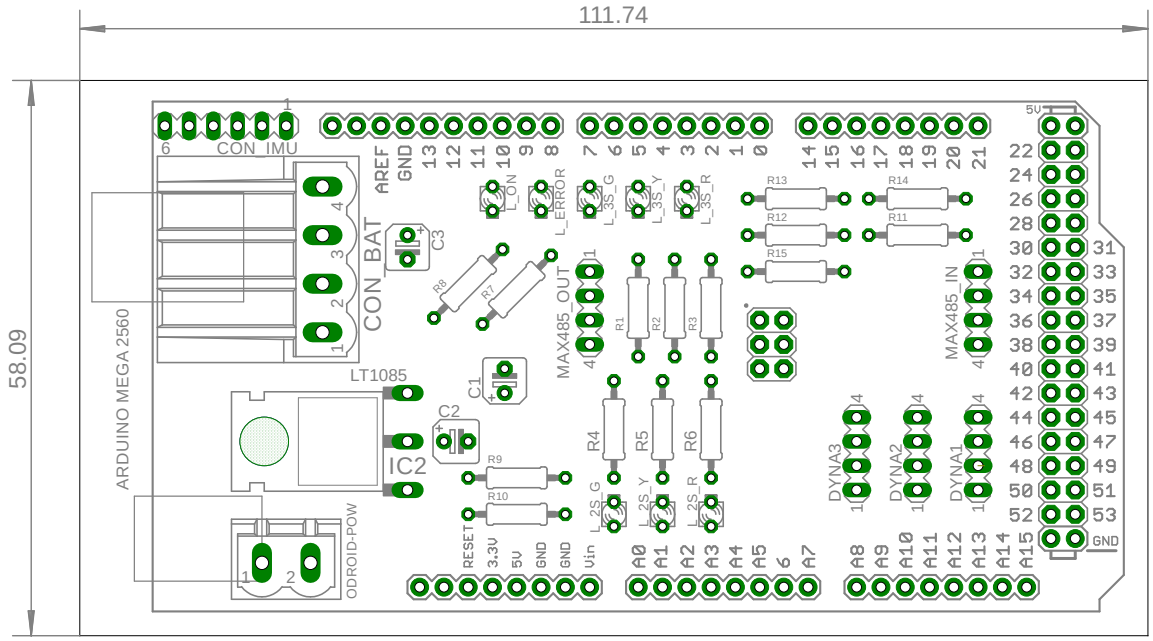

Figure 5.5.3: Top Components and names of the PCB of the Arduino shield.

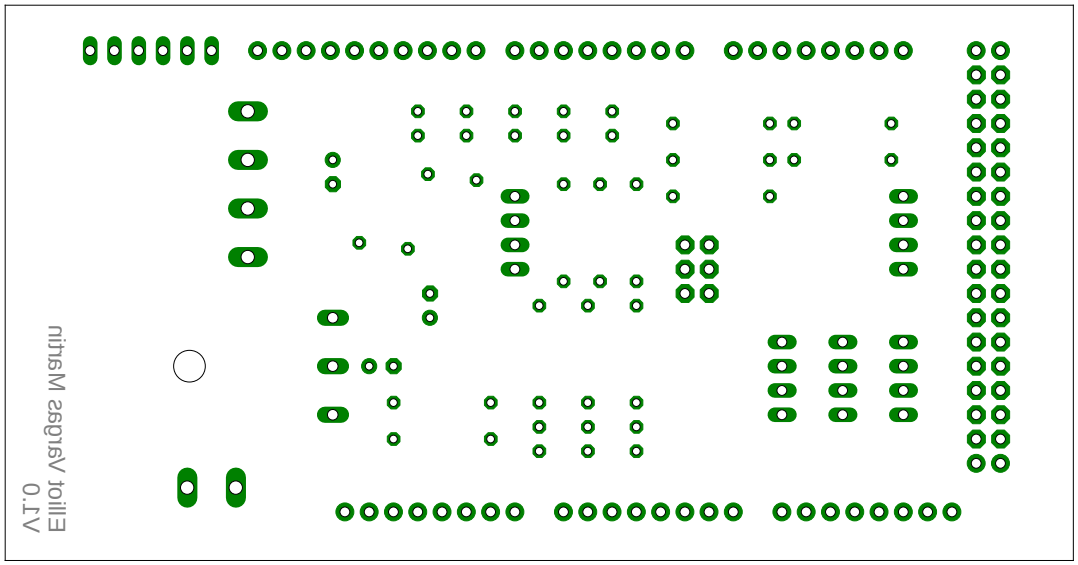

Figure 5.5.4: Bottom Components and names of the PCB of the Arduino shield.

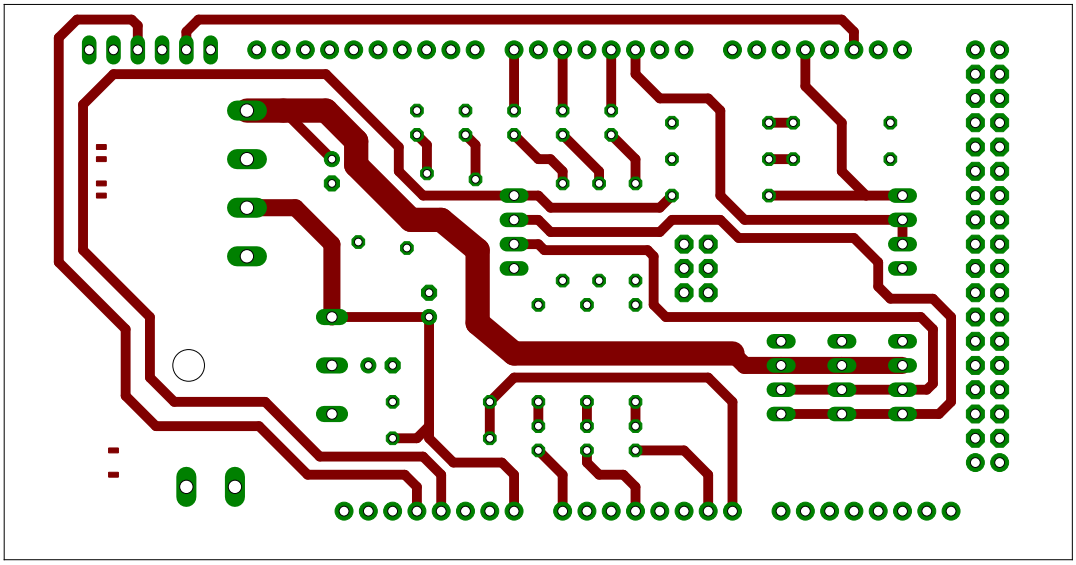

Figure 5.5.5: Top cooper layer the PCB of the Arduino shield.

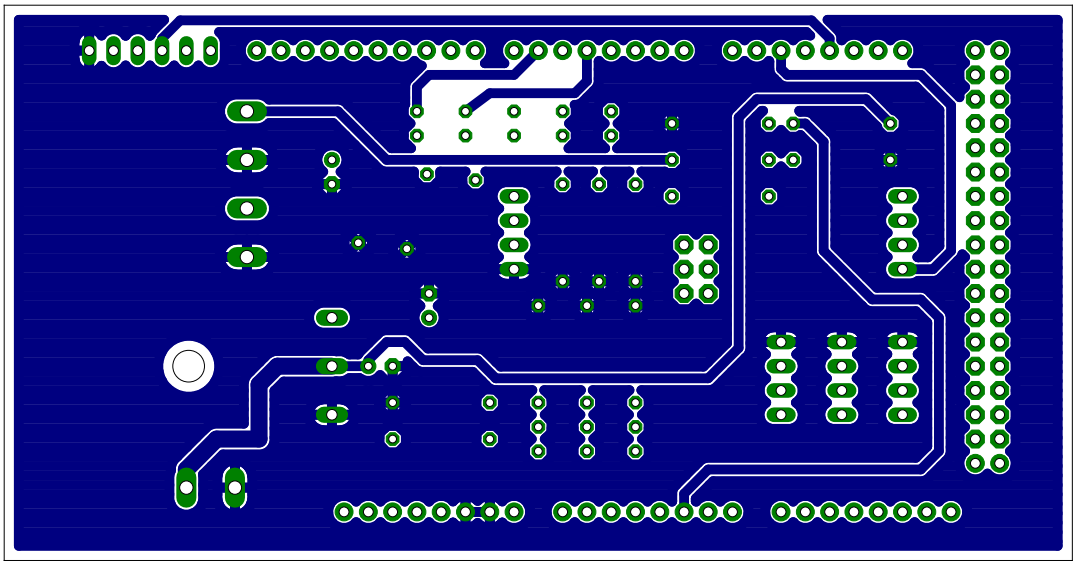

Figure 5.5.6: Bottom cooper layer the PCB of the Arduino shield.

5.6 Odroid

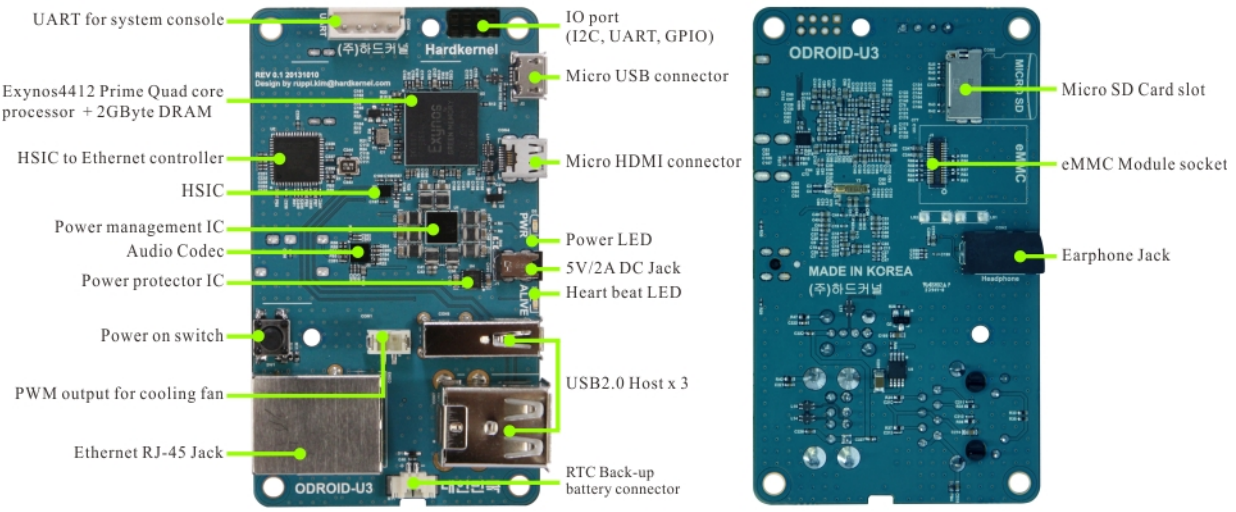

Figure 5.6.1: Odroid u3.

In the market there are a lot of microcomputers, and is hard to decide which is the best to use, the Raspberry Pi has a good community and workgroups behind it, but it has a slow microprocessor, after some research it was decides to use Odroid U3, which it has a fast processor, small size, runs Xubuntu 14.04 and it is relatively cheap.

The features to decide to use Odroid U3 are that it uses Exynos4412, it is a fast quad core that comes with a 2GB LP-DDR2, this specification is very suitable for the size of this device, its size is very small, only 83 mm per 48 mm. It also have 3 USB ports which are enough for this robot.

A favourable thing by using this microcomputer is that it uses Ubuntu, and it can be used a laptop or Odroid with the same Python code without any change.

Concept	**Specifications**
Processor	Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.7Ghz with 1MB L2 cache
Memory	2048MB(2GB) LP-DDR2 880 Mega data rate
3D Accelerator	Mali-400 Quad Core 440MHz
Video	Supports 1080p via HDMI cable(H.264+AAC based MP4 container format)
Video Out	micro HDMI connector
Audio	Standard 3.5mm headphone jack HDMI Digital
LAN	10/100Mbps Ethernet with RJ-45 Jack (Auto-MDIX support)
USB2.0 Host	High speed standard A type connector x 3 ports
USB2.0 Device	ADB/Mass storage(Micro USB), Host mode is possible if the PCB Rev is 0.5 or higher.
Display	HDMI monitor
IO Port	GPIO, UART, I2C, SPI(Board Revision 0.5 or higher)
Storage (Option)	MicroSD Card Slot eMMC module socket
Power (Option)	5V 2A Power
System Software	Linux : Xubuntu 14.10
PCB Size	83 x 48 mm
Weight	48g including the heat sink

Table 5.6.1: Odroid U3 Specifications.

Chapter 6

Design and manufacture of the software of the robot

6.1 Introduction

In this section it will be explained the software related with all the electronics, the communication of the different parts of the robot, and the programs that will run into the Arduino and the Odroid, but not about the *Inverted Pendulum Foot Placement* or the control of the robot.

The robot have a lot of different electronics that can be programmed in different programming languages. For the IMU and the Arduino it will be programmed in Arduino language, and the Odroid will be programmed in Python.

6.2 Razor IMU

To extract the information from the sensors it is used the software provided by the manufacturer Sparkfun, this software is an Arduino code to be programmed in the integrated circuit ATmega328. This software will extract the information from the sensors, will apply the DCM and then it will send the information to the Arduino.

6.2.1 Direction-cosine-matrix (DCM)

The accelerometer, magnetometer and gyroscope data extracted from the 3 sensor has a lot of noise, to extract a good rotation position it is used a Direction Cosine Matrix (DCM)

algorithm. This algorithm takes care of handling sensor noise and numerical errors. It is based on this paper by William Premerlani [24]. More information can be find in the papers by Robert Mahony [20] [13] [7] [21].

6.2.2 Sensor calibration

The calibration is needed in order to fit the signals from the sensor to a real ones. The procedure to calibrate each sensor is the following :

- Calibrating the accelerometer: Find the minimum and maximum output values for the earth gravitation on each axis.
- Calibrating the magnetometer: Move it away from magnetic distortions introduced by computers and other electronic devices and metal objects.
- Calibrating the gyroscope : Collect the average of the noise of the gyroscope on all three axes. Keep the equipment in rest.

Then with this parameters Table 6.2.1 the software is able to find the offset and the gain of each sensor and compensate it to have more accurate estimation of the angular position.

Sensor	Values	X axis	Y axis	Z axis
Acceleration	min/max	-277.00/265.00	-271.00/274.00	-285.00/256.00
Magnetometer	min/max	-677.00/115.00	-517.00/451.00	-139.00/910.00
Gyroscope:	average	-43.43	-106.40	26.96

Table 6.2.1: Values of the calibration of the IMU sensors.

6.2.3 Sensor Output

Finally the information that the IMU sento to the Arduino every 50 ms is the following:

- Acceleration (X-Y-Z) axis
- Magnetometer (X-Y-Z) axis
- Gyroscope (X-Y-Z) axis
- Pitch
- Roll
- Yaw

6.3 Dynamixel

Area	Address	Name	Access	Initial Value
	0 (0X00)	Model Number(L)	R	24 (0X18)
	1 (0X01)	Model Number(H)	R	0 (0X00)
E	2 (0X02)	Version of Firmware	R	-
	3 (0X03)	ID of Dynamixel	RW	1 (0X01)
E	4 (0X04)	Baud Rate	RW	34 (0X22)
	5 (0X05)	Return Delay Time	RW	250 (0XFA)
P	6 (0X06)	CW Angle Limit(L)	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	RW	0 (0X00)
R	8 (0X08)	CCW Angle Limit(L)	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	RW	3 (0X03)
O	11 (0X0B)	The Highest Limit Temperature	RW	80 (0X50)
	12 (0X0C)	The Lowest Limit Voltage	RW	60 (0X3C)
M	13 (0X0D)	the Highest Limit Voltage	RW	190 (0XBE)
	14 (0X0E)	Max Torque(L)	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	RW	3 (0X03)
	16 (0X10)	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	RW	36 (0X24)

Table 6.3.1: Dynamixels Control Table (EEPROM).

The Dynamixel has its own controller inbuilt which handles the control of the motor and the communication, the Dynamixels are controlled, configured and readed by accessing in to their register, it can be seen in the Table 6.3.1 and Table 6.3.2.

The parameters of the EEPROM of each motor, see Table 6.3.1 will be configured with the RoboPlus software that it is provided by the manufacturer. The parameters configured are :

- The maximum angle of each motor.
- The minimum angle of each motor.
- Id of each motor from 1 to 7.
- Baud Rate of 1000000 baud.

The maximum and minimum angle of each motor are set in order to assure that the different parts of the robot can not collide.

The parameters of the RAM, see Table 6.3.2 will change with the Arduino, the register that will be used are:

- Torque Enable.

- Goal Position(L) and Goal Position(H).

- Present Position(L) and Present Position(H).

- Present Temperature.

Area	Address	Name	Access	Initial Value
	24 (0X18)	Torque Enable	RW	0 (0X00)
	25 (0X19)	LED	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	RW	1 (0X01)
	27 (0X1B)	CCW Compliance Margin	RW	1 (0X01)
	28 (0X1C)	CW Compliance Slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	RW	32 (0X20)
R	30 (0X1E)	Goal Position(L)	RW	-
	31 (0X1F)	Goal Position(H)	RW	-
	32 (0X20)	Moving Speed(L)	RW	-
	33 (0X21)	Moving Speed(H)	RW	-
	34 (0X22)	Torque Limit(L)	RW	ADD14
	35 (0X23)	Torque Limit(H)	RW	ADD15
A	36 (0X24)	Present Position(L)	R	-
	37 (0X25)	Present Position(H)	R	-
	38 (0X26)	Present Speed(L)	R	-
	39 (0X27)	Present Speed(H)	R	-
	40 (0X28)	Present Load(L)	R	-
	41 (0X29)	Present Load(H)	R	-
M	42 (0X2A)	Present Voltage	R	-
	43 (0X2B)	Present Temperature	R	-
	44 (0X2C)	Registered	R	0 (0X00)
	46 (0X2E)	Moving	R	0 (0X00)
	47 (0X2F)	Lock	RW	0 (0X00)
	48 (0X30)	Punch(L)	RW	32 (0X20)
	49 (0X31)	Punch(H)	RW	0 (0X00)

Table 6.3.2: Dynamixels Control Table (RAM).

6.4 Arduino

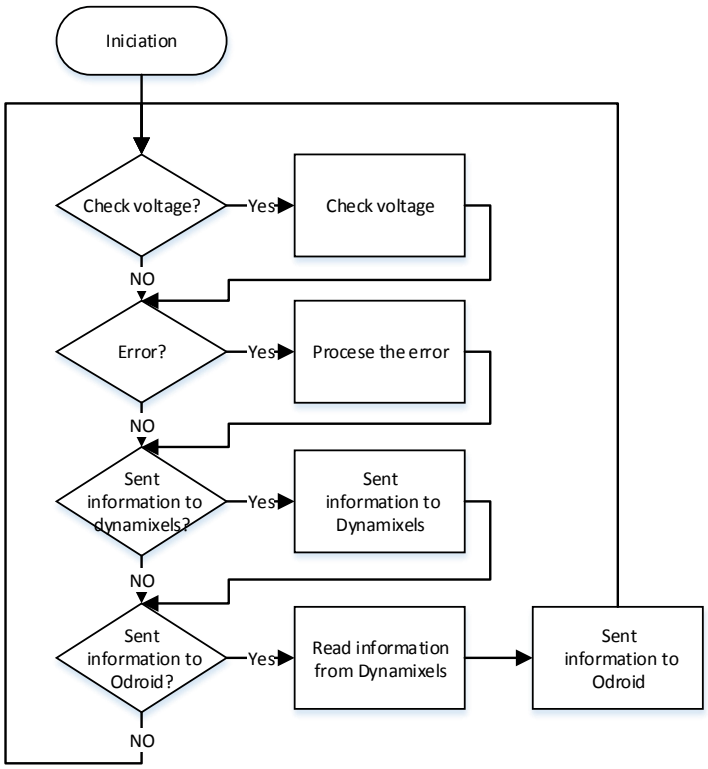

Figure 6.4.1: Flowchart of the main program of the Arduino.

The Arduino is the part of the robot's electronics which controls, reads and communicates all electronics parts, specifically the tasks that it has to manage are:

- Read the information from IMU using a serial port 1.
- Read the information from Dyanmixels using a serial port 2.
- Sent the commands from Micro PC to the Dyanmixels using a serial port 2.
- Sent the information of the IMU and Dynamixels to the Micro PC using the serial port 0.
- Read the commands of Micro PC using the serial port 0.
- Check the voltage of batteries and Odroid.
- Check the voltage, temperature and communication problem of the Dynamixels.
- Show if there are an error.

The software of the microcontroller Arduino is programmed in Arduino language using the open-source Arduino Software (IDE).

As it can be seen in the Figure 6.4.1 the main part of the Arduino software start initializing the variables and the configuration of the microcontroller, after that, the program enters in a loop in which the program check continuously, it checks the batteries voltage, if there are and error and if it has to sent information to the Dynamixels or Odroid, if one of this condition is fulfilled the software make the proper action.

This conditions are activated depending on different conditions, for example the Arduino check every second the voltage of the two batteries and Odroid power supply, the system detects if there are and error if any of the voltage or temperature inputs are out of range, or if there are and communication problem.

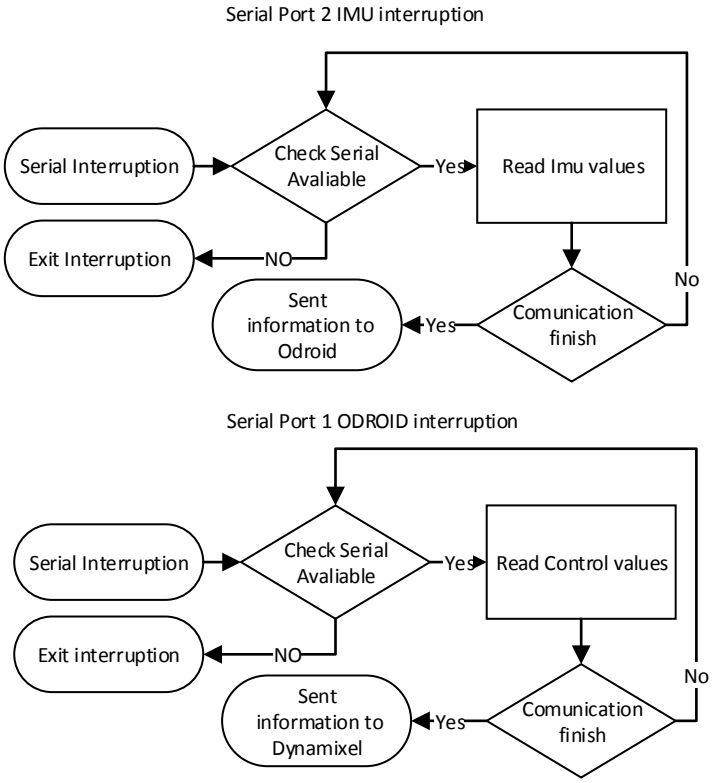

Figure 6.4.2: Flowchart of the serial interruptions of the Arduino.

The condition that activates if the system has to sent the information to the Dynamixel's or Odroid are triggered when arrives information from Odroid and IMU respectively as can be seen in the Figure 6.4.2

6.5 Communication protocols

All the electronics are communicated with the Arduino and all the information passes through to the Arduino. A fast communication protocols between the Dynaminxels, IMU and Odroid with the Arduino is needed to have a fast execution time. In the following sections the differences will be explained.

6.5.1 Communication Dynamixels-Arduino

Instructions	Function	R/W	A	B
self.begin(A,B)	Serial Configuration	W	Baud	Control pin
self.move(A,B)	Set Target Position	W	Dynamixel id	Target position
self.readTemperature(A)	Read Temperature	R	Dynamixel id	—
self.readVoltage(A)	Read Voltage	R	Dynamixel id	—
self.readPosition(A)	Read Position	R	Dynamixel id	—

Table 6.5.1: Dynamixels-Arduino communication protocol.

In order to manage this communication protocol it is used the library DynamixelSerial3 of Savage Electronics, in the Table 6.5.1 there are the most important function used to the control of the Dynamixels.

The communication with the Dymanixel is by sending the instruction with the id of the Dyamixel, the address of register and new data if it is wanted to write register readed. We can see the registers in Table 6.3.1 and Table 6.3.2

The control pin is needed because the serial port of the Dynamixel is duplex, the control converts the duplex signal to a half duplex. A control signal is needed to switch between send and receive.

6.5.2 Communication IMU-Arduino

Position	nº bytes	Information	Code	Value
1	1	Start	Unsigned Byte	0x24
3-5-7-9-11-13	1	New data	Unsigned Byte	0x2C
15-17-19-21-23				
25	1	End	Unsigned Byte	0x23
2-4-6	1-6	Acceleration (X-Y-Z) axis	ASCII	[-99999,99999]
8-10-12	1-6	Magnetometer (X-Y-Z) axis	ASCII	[-99999,99999]
14-16-18	1-6	Gyroscope (X-Y-Z) axis	ASCII	[-99999,99999]
20	1-6	Pitch	ASCII	[-18000,18000]
22	1-6	Roll	ASCII	[-18000,18000]
24	1-6	Yaw	ASCII	[-18000,18000]

Table 6.5.2: IMU - Arduino communication protocol.

The Communication between IMU and Arduino is made only in one direction from IMU to Arduino. The information is sent with a frequency of 20Hz. All the data is sent only in one message with the sequence of the Table 6.5.2. It is selected this type of protocol because it is fast, easy to implement with low parameters, and because it is not needed to sent information from Arduino to IMU.

The communication starts with one start byte the ”0x24”, then data is sent in ASCII code inserted with one byte, the ”0x2C” to indicate new data, when the all the data is sent one byte of End the ”0x23” is sent.

6.5.3 Communication Odroid-Arduino and Arduino-Odroid

Position	nº bytes	Information	Code	Value
1	1	Start	Unsigned Byte	0x24
3-5-7-9-11	1	New data	Unsigned Byte	0x2C
13-15-17-19				
21-23-25.27				
29-31-33-35				
37	1	End	Unsigned Byte	0x23
2-4-6-8-10-12	1-4	Motor (1-2-3-4-5-6) Position	ASCII	[0,1023]
14-16-18	1-6	Acceleration (X-Y-Z) axis	ASCII	[-99999,99999]
20-22-24	1-6	Magnetometer (X-Y-Z) axis	ASCII	[-99999,99999]
26-28-30	1-6	Gyroscope (X-Y-Z) axis	ASCII	[-99999,99999]
32	1-6	Pitch	ASCII	[0,36000]
34	1-6	Roll	ASCII	[0,36000]
36	1-6	Yaw	ASCII	[0,36000]

Table 6.5.3: Arduino-Odroid communication protocol.

The Communication protocol Odroid-Arduino and Arduino-Odroid is very similar to the IMU-Arduino, the only difference is that different data is sent as it can be seen in the Table 6.5.3 and 6.5.4.

Position	nº bytes	Information	Code	Value
1	1	Start	Unsigned Byte	0x24
3-5-7-9-11	1	New data	Unsigned Byte	0x2C
13	1	End	Unsigned Byte	0x23
2-4-6-8-10-12	1-4	Motor (1-2-3-4-5-6) set point	ASCII	[0,1023]

Table 6.5.4: Odroid-Arduino communication protocol.

6.6 Odroid

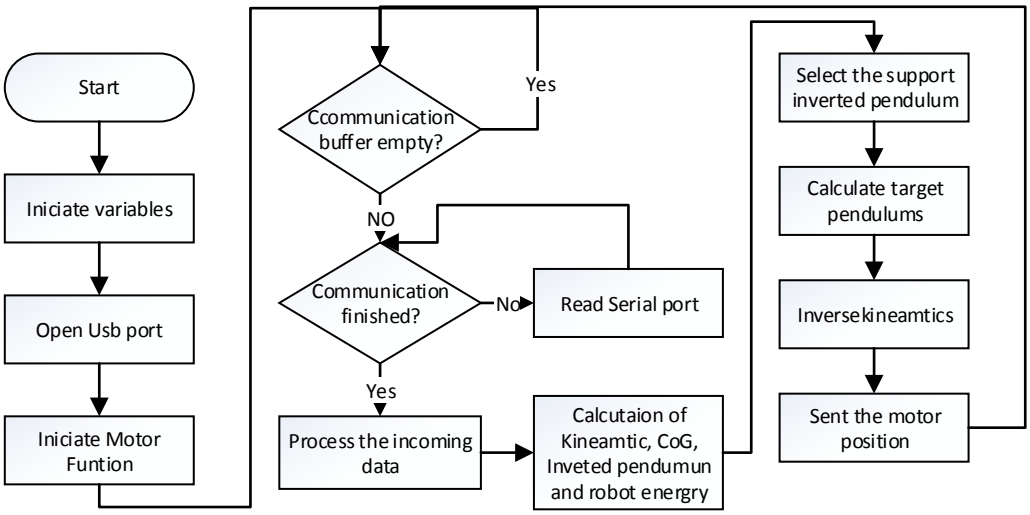

Figure 6.6.1: Flowchart of Odroid.

The software of the Odroid is made in Python language, it means that the code can be run in any system with Linux and one serial port, this fact make easy to change the microcomputer with any change in the software. Other important thing to use Python is that it makes easier that in the future other people can implement another algorithm to control the robot without having good programming skills.

The flowchart of the robot starts initializing all the global variables, then the Odroid search the USB port where the Arduino is connected and it opens the socket.

The *Inverted Pendulum Foot Placement* algorithm have as an output a target position of the robot that it is translated in motor position, it means that it is needed an auxiliary control of the motor to control the maximum velocities of the robot. So the algorithm need to initiate the motors controller.

After all the initialization the software start a loop, this loop checks if has arrived information from the Arduino, when the USB buffer has at least 30 bytes, the system starts to read and assigns the information until all the data it is read. By using this kind of algorithm it means that algorithm starts when the information of the Arduino arrives to the computer, and it happens every 50 ms.

Then, all the received data is processed to be into a correct measurement units, and also it is filtered to be more robust to the error lecture and disturbance.

Since this point the *Inverted Pendulum Foot Placement* starts. Now it will be a short summary about how it works, it will be a further explanation in the next section.

With all the incoming information from the Arduino all the position of all joints are calculated. All the CoG, the two pendulums, and all the energies of the robot. Then with the energies and the two pendulum the target pendulums are calculated; those target pendulums will be transformed into motors position with inverse kinematics. Finally the target position is computed in the motors controllers and it is send to the Dynamixels.

6.7 Graphic user interface

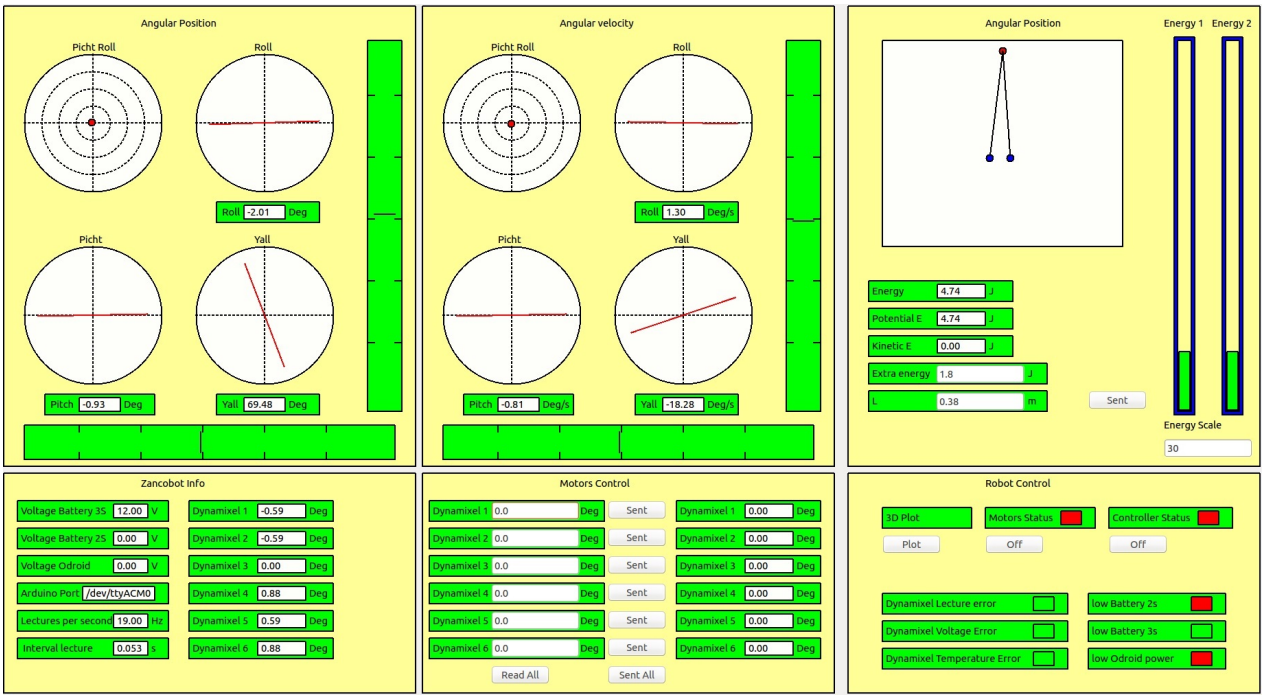

Figure 6.7.1: Graphic user interface.

The GUI is designed in order to show the information of the robot and controlling some parameters to help the developing of the *Test Algorithm* and future algorithms. This GUI only works in the slim version, the GUI is divided in 6 parts as can be seen in the Figure 6.7.1:

- Angular Position: It shows the Roll, Pitch and Yaw position. numerically and also with angular indicator, the vertical bar show more precisely the Roll and in the horizontal

bar the pitch. Figure 6.7.2.

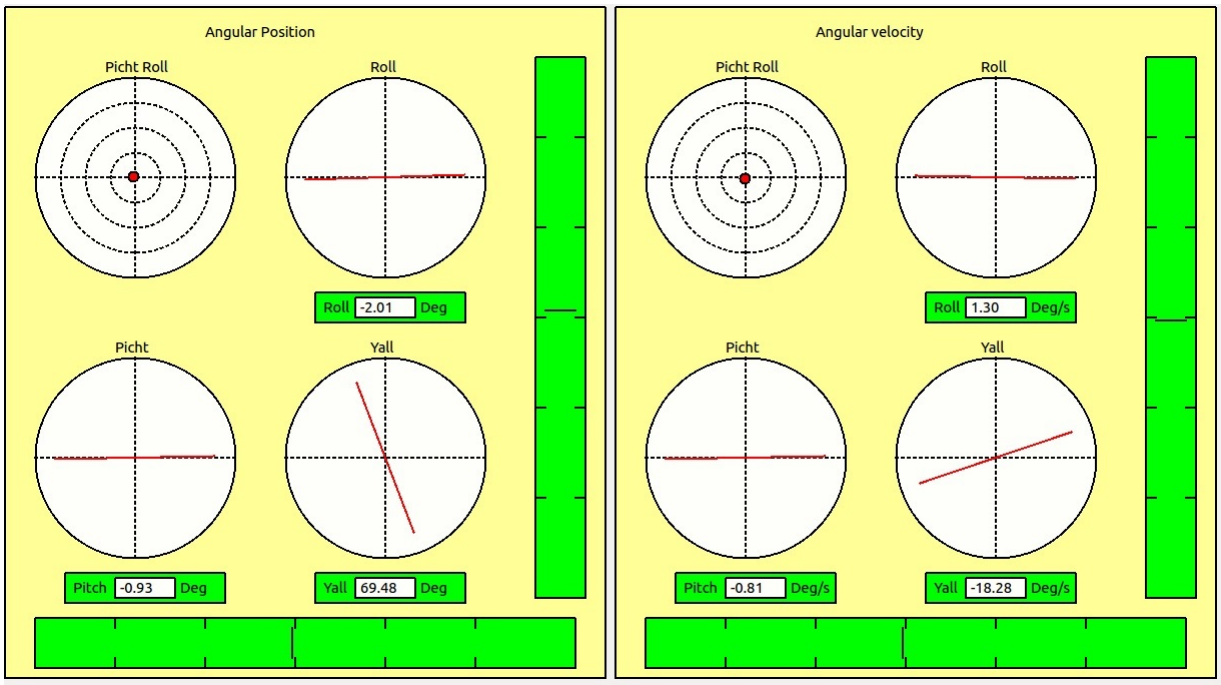

Figure 6.7.2: Angular position and velocity part of the GUI.

- Angular Velocity: It shows the Roll, Pitch and Yaw velocity. numerically and also with angular indicator, the vertical bar show more precisely the Roll and in the horizontal bar the pitch. Figure 6.7.2.

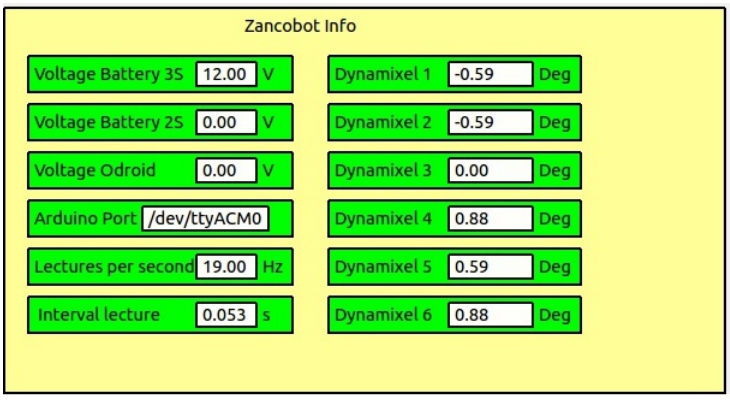

Figure 6.7.3: Zancobot information part of the GUI.

- Zancobot information:. In this part it is showed the actual position of all motors, all voltages, the port that the Arduino is connected, the actual operation frequency and the interval value of lecture of the robot. Figure 6.7.3.

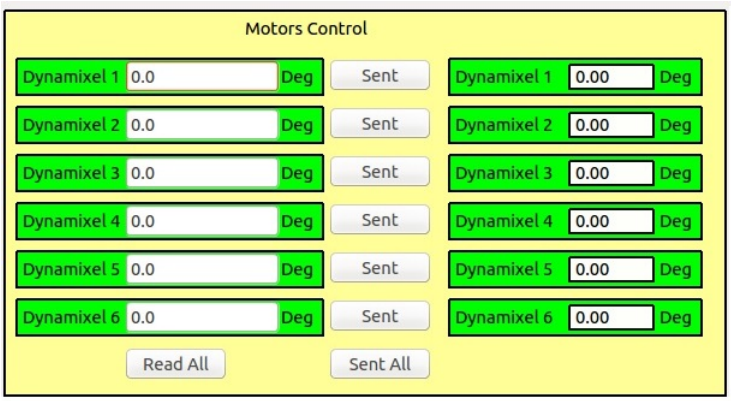

Figure 6.7.4: Motors Control part of the GUI.

- Motors Control: In the left column of this part of the GUI can be introduced manually the target position of the motors, pressing the ”Sent All” button the robot archive the desired position, one motor position can be changed using his ”sent” button: the ”Read All” button put in the left column the actual target position of the servos. And in the right column the target position of the motors are showed. Figure 6.7.4.

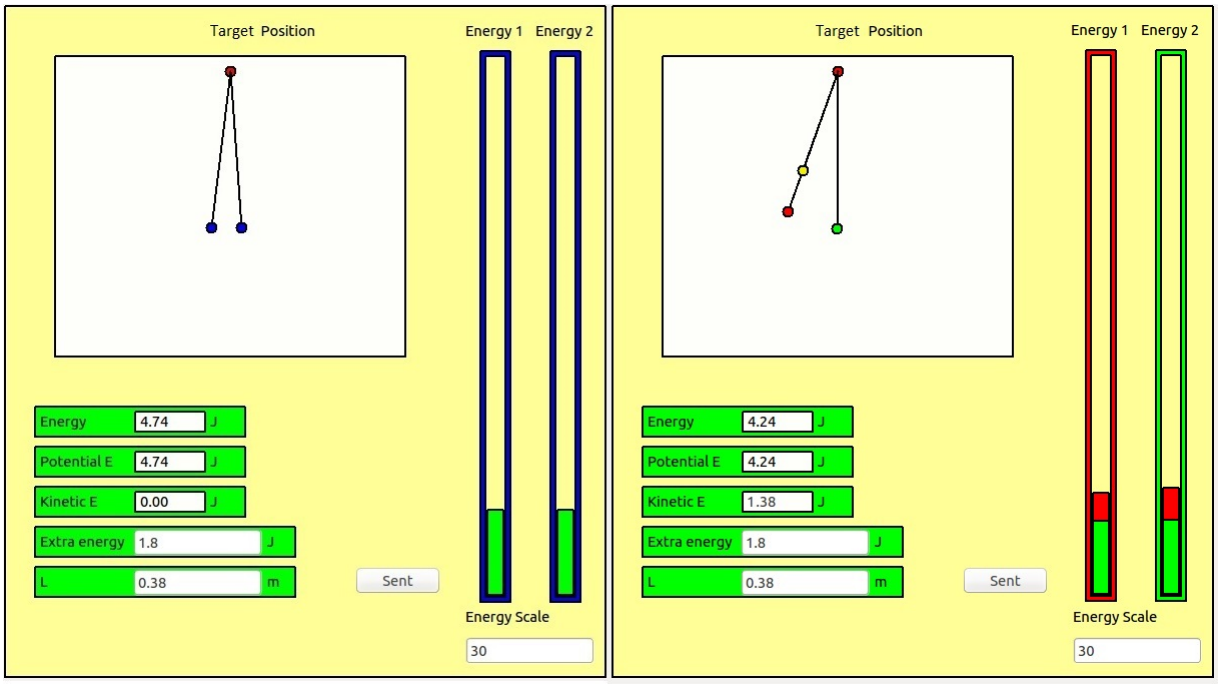

Figure 6.7.5: Target Position part of the GUI.

- Target Position. In the graphic is showed the two pendulums of the robot in two dimensions, these pendulums change the color depending of their status: Blue the two extremities are touching the flour, red the extremity it is free and green the extremity it is supporting the robot. In the left bars it is showed the energy of the two pendulums, in green the potential energy and in red the kinetic energy. The scale of the energy bars can

be changed the *EnergyScale* number. The *Energy*, *PotentialE* and *KineticE* boxes show the energies of the support pendulum. Finally the *PotentialE* and *KineticE* parameters can be changed in order to modify the target position of the free pendulum. The *extraenergy* parameter and the L that it is the longitude of the target inverted pendulum can be changed filling the boxes and pushing "sent" button. Figure 6.7.5.

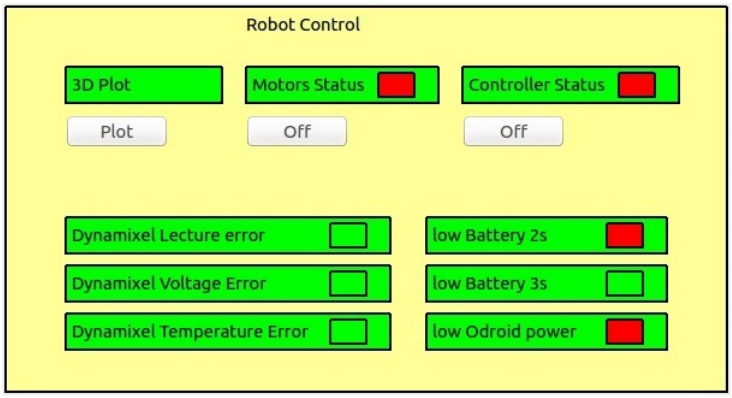

Figure 6.7.6: Robot control part of the GUI.

- Robot control. In the last part of the GUI are showed all the errors of the robot, the green color indicate no error, the red color error and the yellow color indicates a warning. Pushing the "plot" button is showed a 3d representation of the robot, as in the Figure 7.3.1, using the other 2 buttons the motors can be stopped and also the controller of the robot. Figure 6.7.6.

Chapter 7

Test Algorithm

7.1 Introduction

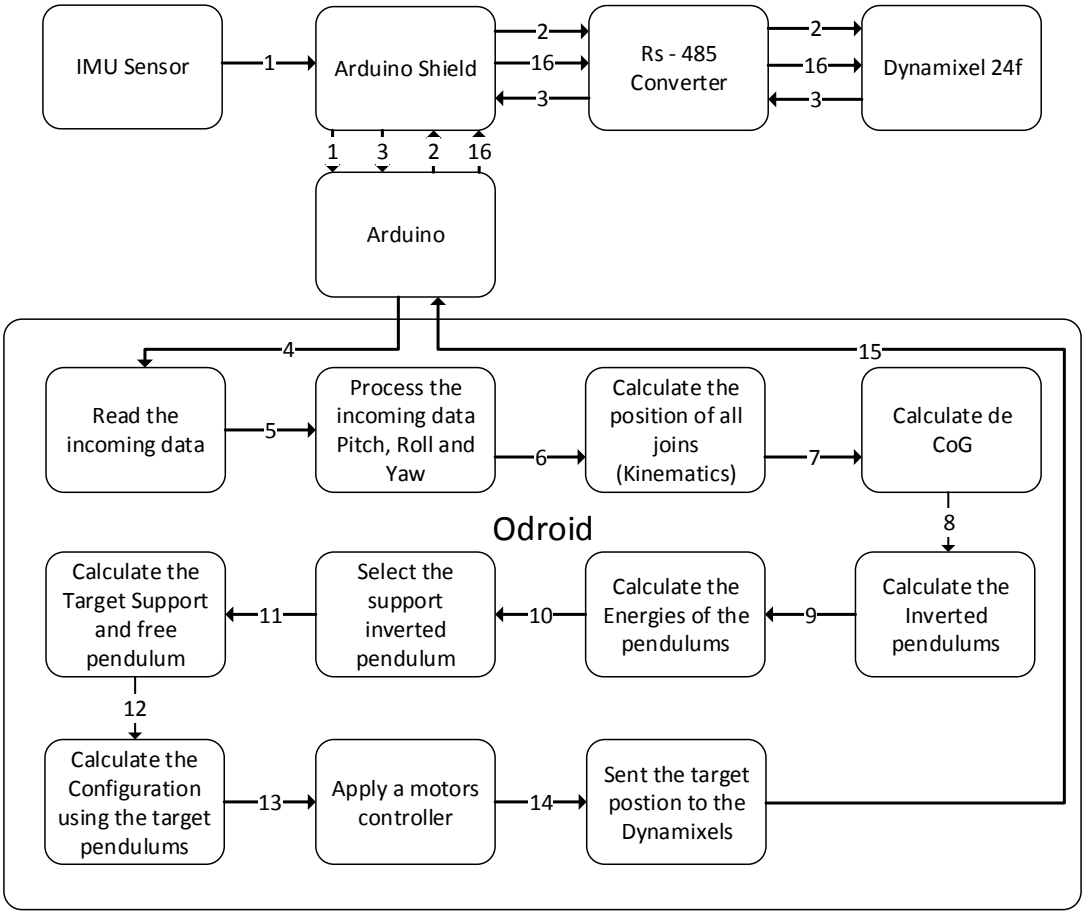

Figure 7.1.1: Schematics of the test algorithm.

In order to test that the robot can be controlled by an *Inverted Pendulum Foot Placement* algorithm it has been created a *Test Algorithm* that will test if the robot works correctly. This algorithm is based in an *Inverted Pendulum Foot Placement* model, but it is a fast algorithm and will try to make the robot to stand, helped by the Stability Assistant.

This is a fast implementation algorithm. The parameters of this algorithm are that the robot will be in the *Stability Assistance* which helps the robot to not fall down and also limits the robot to move in the X axis.

The main steps of this *Test Algorithm* can be seen in the Figure 7.1.1 and they are the following:

1. The IMU send the angular position of the robot and the IMU sensors output to the Arduino.
2. The Arduino ask to all the Dynamixels they angular position, temperature and operation voltage.
3. The Dynamixels send they position, temperature and operation voltage.
4. The Arduino send the the angular position of the robot, the IMU sensors output, the Dynamixels angular position, temperature and operation voltage, the batteries voltage and the supply voltage of the Odroid.
5. The Odroid receives the data sent from the Arduino.
6. Process the incoming data as Pitch, Roll and Yaw.
7. Calculate the position of all joints (Kinematics) and the partials CoG.
8. Calculate the CoG.
9. Calculate the Inverted pendulums Position and Velocity.
10. Calculate the Inverted pendulums Energies.
11. Select the support inverter pendulum and the free inverted pendulum.
12. Calculate the target Support and free inverted pendulum.
13. Calculate the Target rotation position of the servomotors using the target pendulums.
14. Apply the output to the motors controller.
15. The Odroid send the target rotation position of the servomotors to the Arduino.
16. The Arduino send the target rotation position of the servomotors to the Dynamixels.

7.2 Process the incoming

The incoming data of the robot is not in useful units and for this reason needs to be processed. The robot's encoder are of 10 bits and it represents the range of [-150, 150] degrees and

it is converted to Radians. The rotation of the robot is in centidegrees and they will be transformed to radians. After that, a filter is applied. Also, the rotation velocity is extracted from the angular position.

7.3 Kinematics

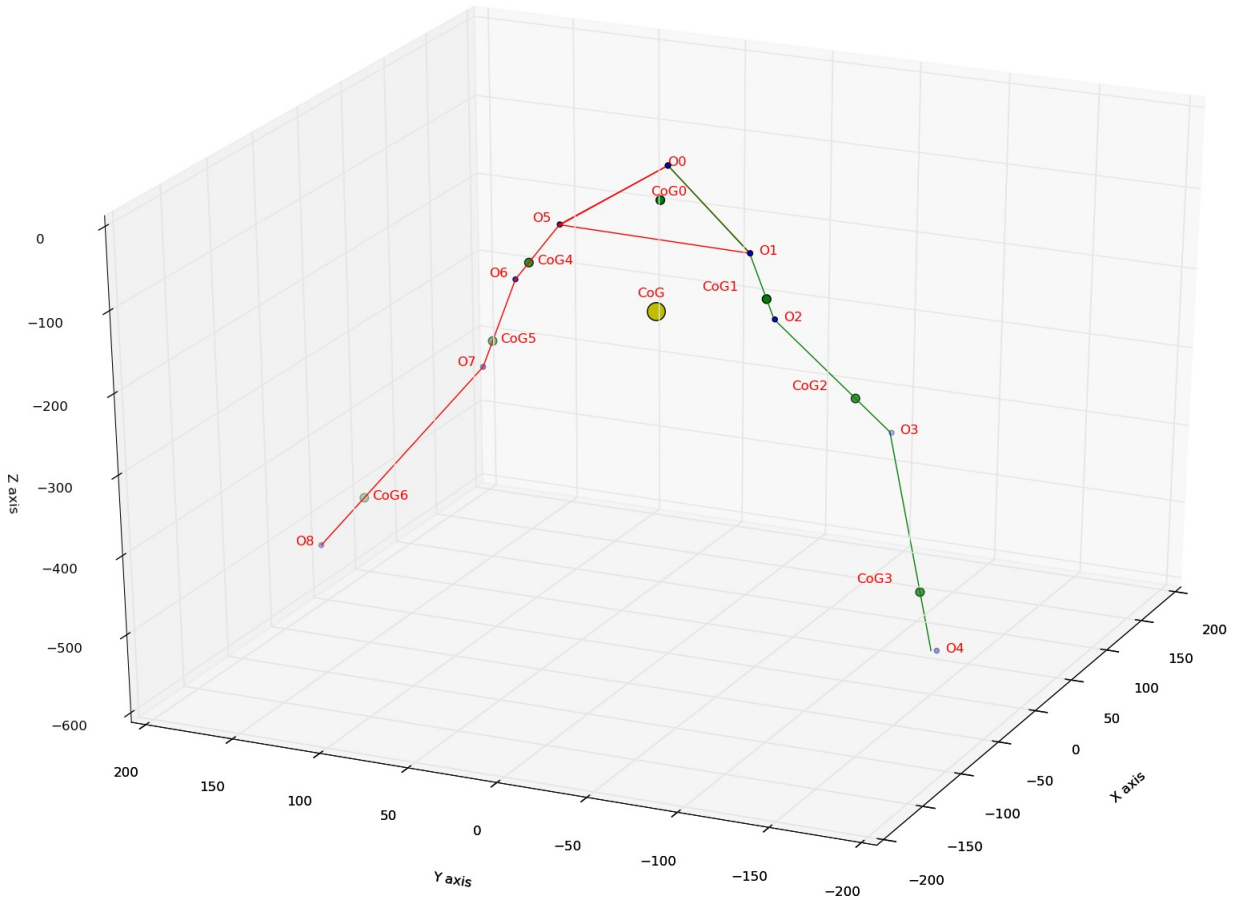

Figure 7.3.1: Position of the frames and the CoG

To know were it is each part of the robot it is important to implement the IPM. To extract this, transformation matrices are used in order to transform one frame to another. The robot has 7 frames $O_0, O_1, O_2, O_3, O_4, O_5, O_6, O_7, O_8$. As we can see in the Figure 7.3.1. The O_0 is located in the IMU, but the orientation is the real world. The $O_1, O_2, O_3, O_5, O_6, O_7$ are situated in the rotation axis of the Dynamixels (3, 2, 1, 6, 5, 4) respectively, and the O_4, O_8

are situated in the contract point of the right foot and left foot respectively.

$$T_{01} = \begin{bmatrix} C_R & S_P\,S_R & C_P\,S_R & -L1\,C_P\,S_R - L2\,S_P\,S_R \\ 0 & C_P & -S_P & -L2\,C_P + L1\,S_P \\ -S_R & C_R\,S_P & C_P\,C_R & -L1\,C_P\,C_R - L2\,C_R\,S_P \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.1}$$

In order to make the transformation from the frame O_0, which has the orientation of the real world, to the motors 3 and 6, it is needed the orientation of the chassis respect to the world. To extract it, the Pitch, Roll and Yaw are used that they are provided by the IMU. But Yaw of the IMU is not accurate, for this reason it is considered Yaw = 0. Then the resultant transformation matrices are Equation 7.3.1 and Equation 7.3.2, where C_R is the cosine of Roll, C_P is the cosine of Pitch, S_R is the sine of Roll, S_P is the sine of Pitch.

$$T_{05} = \begin{bmatrix} C_R & S_P\,S_R & C_P\,S_R & -L1\,C_P\,S_R + L2\,S_P\,S_R \\ 0 & C_P & -S_P & L2\,C_P + L1\,S_P \\ -S_R) & C_R\,S_P & C_P\,C_R & -L1\,C_P\,C_R + L2\,C_R\,S_P \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.2}$$

In the Equations 7.3.3, 7.3.4, 7.3.5, 7.3.6, 7.3.7 and 7.3.8 it can be seen the rest transformations where $Theta_1, Theta_2, Theta_3, Theta_4, Theta_5, Theta_6$ are the rotation position of the motors.

$$T_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(Theta_1) & -Sin(Theta_1) & L3\,Sin(Theta_1) \\ 0 & Sin(Theta_1) & Cos(Theta_1) & -L3\,Cos(Theta_1) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.3}$$

$$T_{23} = \begin{bmatrix} Cos(Theta_2) & 0 & Sin(Theta_2) & -L4\,Sin(Theta_2) \\ 0 & 1 & 0 & 0 \\ -Sin(Theta_2) & 0 & Cos(Theta_2) & -L4\,Cos(Theta_2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.4}$$

$$T_{34} = \begin{bmatrix} Cos(Theta_3) & 0 & Sin(Theta_3) & -L5\,Sin(Theta_3) \\ 0 & 1 & 0 & 0 \\ -Sin(Theta_3) & 0 & Cos(Theta_3) & -L5\,Cos(Theta_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.5}$$

$$T_{56} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(Theta_4) & -Sin(Theta_4) & L3\,Sin(Theta_4) \\ 0 & Sin(Theta_4) & Cos(Theta_4) & -L3\,Cos(Theta_4) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.6}$$

$$T_{67} = \begin{bmatrix} Cos(Theta_5) & 0 & Sin(Theta_5) & -L4\,Sin(Theta_5) \\ 0 & 1 & 0 & 0 \\ -Sin(Theta_5) & 0 & Cos(Theta_5) & -L4\,Cos(Theta_5) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.7}$$

$$T_{78} = \begin{bmatrix} Cos(Theta_6) & 0 & Sin(Theta_6) & -L5\,Sin(Theta_6) \\ 0 & 0 & 0 & 1 \\ -Sin(Theta_6) & 0 & Cos(Theta_6) & -L5\,Cos(Theta_6) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.8}$$

And finally multiply the transformations to have the transformations matrix from all frames to O_0 as it can be seen in the Equations 7.3.9, 7.3.10, 7.3.11, 7.3.12, 7.3.13 and 7.3.14.

$$T_{02} = T_{01}T_{12} \tag{7.3.9}$$

$$T_{03} = T_{01}T_{12}T_{23} \tag{7.3.10}$$

$$T_{04} = T_{01}T_{12}T_{23}T_{78} \tag{7.3.11}$$

$$T_{06} = T_{05}T_{56} \tag{7.3.12}$$

$$T_{07} = T_{05}T_{56}T_{67} \tag{7.3.13}$$

$$T_{08} = T_{05}T_{56}T_{67}T_{78} \tag{7.3.14}$$

$$T_{00'} = \begin{bmatrix} C_R & S_P\,S_R & C_P\,S_R & 0 \\ 0 & C_P & -S_P & 0 \\ -S_R & C_R\,S_P & C_P\,C_R & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.15}$$

7.4 Center of Gravity (CoG)

Then with all the transformation matrices of the frames, the localization of all the CoG of the fixed parts can be found. Using the local position of the partials CoG Equation 7.4.1 and 7.4.2 which are extracted from the Table 4.1.1. Then, it will be multiplied the transformation matrices to put the partial CoG in to O_0 frame as in the Equations 7.4.3.

$$CoGSlim_0 = \begin{bmatrix} -1.5 \\ 0 \\ -38.47 \\ 1 \end{bmatrix}, \quad CoG_0 = \begin{bmatrix} -1.89 \\ 0 \\ -78 \\ 1 \end{bmatrix} \tag{7.4.1}$$

$$CoG_1 = CoG_4 = \begin{bmatrix} 0 \\ 0 \\ 21.91 \\ 1 \end{bmatrix}, \quad CoG_2 = CoG_5 = \begin{bmatrix} 0 \\ 0 \\ 54.74 \\ 1 \end{bmatrix}, \quad CoG_3 = CoG_6 = \begin{bmatrix} 0 \\ 0 \\ 60.21 \\ 1 \end{bmatrix} \tag{7.4.2}$$

$$CoG'_0 = T_{00'}\,CoG_0, \quad CoG'_1 = T_{01}\,CoG_1, \quad CoG'_2 = T_{02}\,CoG_2, \quad CoG'_3 = T_{03}\,CoG_3, \\ CoG'_4 = T_{04}\,CoG_4, \quad CoG'_5 = T_{05}\,CoG_5, \quad CoG'_6 = T_{06}\,CoG_6 \tag{7.4.3}$$

Finally to extract the position of the CoG, Equation 7.4.4 is applied.

$$CoG = \frac{CoG'_0\,W_0 + CoG'_1\,W_1 + CoG'_2\,W_2 + CoG'_3\,W_3 + CoG'_4\,W_4 + CoG'_5\,W_5 + CoG'_6\,W_6}{W} \tag{7.4.4}$$

7.5 Inverted pendulums Position and Velocity

$$ContactPointRight = T_{04} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad ContactPointLeft = T_{08} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{7.5.1}$$

To extract the IPM of the robot, the localization of the CoG as well as the contact point of the robot with the ground are needed. The robot has two possible contact points. To extract these contact points, Equation 7.5.1 is used. One of the contact points has to be selected to apply the IPM, but for the moment both pendulums are calculated. These two pendulums are the distances in three axis between the contact points and the CoG.

To extract the energies, the velocity of the pendulum is needed. The velocity in each axis is extracted from the difference between the current time pendulum and the last time pendulum. One filter is needed in order to be robust.

7.6 Energies of the inverted pendulums

$$KineticEnergy = \frac{1}{2} m\, v^2 \qquad PotentialEnergy = m\, g\, h \tag{7.6.1}$$

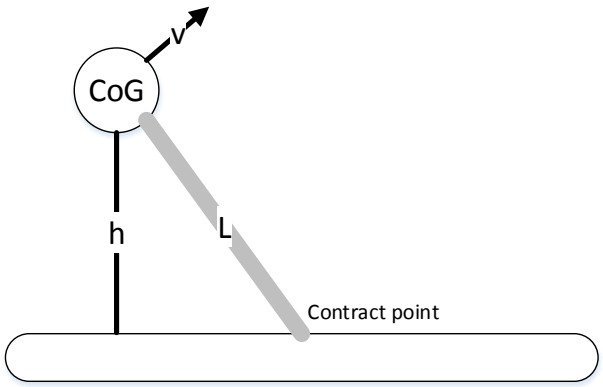

Figure 7.6.1: Inverted pendulum model.

The input for the control action are the kinetic and the potential energy of the robot. To compute this energy using the Equation 7.6.1 it is needed the v and h of the Figure 7.6.1.

7.7 Selection of the support and free inverted pendulum

The position of the end of the extremity is known through the kinematics, but it is also necessary to know which is the extremity that is in contact with the floor. One way to know which of the extremities are in contact with the ground is to use a contact sensor, but the good ones are very expensive. In case of a flat floor, there is another way: using the position of the end of the extremities that are referenced to the IMU. The lower extremity in the Z axis will be the contact extremity.

When the robot is full supported by one extremity this method have good results, but when it is near to the change point, the robot results are not reliable. Therefore, a filter has to be implemented. The output for this selection is that the robot is supported by the extremity 1, extremity 2 or both. When the support change is triggered,there are some parameters that will be used in the selection of the target inverted pendulums.

7.8 Target Inverted pendulums

This is the most important part of the algorithm, because it is the part which select the target position of the end of the extremities. This part has to specify the inverted pendulums, that will designate the robot support extremities and the free extremity.

The support inverted pendulum will have a 0 value for the X axis and the Y and Z axis are described by a $SupportAngle$ angle and $SupoortDistance$ distance. The $SupoortAngle$ has a fixed value and the $SupportDistance$ start with the distance before the change of support feed until arrives to a fixed value $MaxDistaceSupport$, the increase of this distance depends on $TimetoLastChange$ that it is the time from the robot change the status.

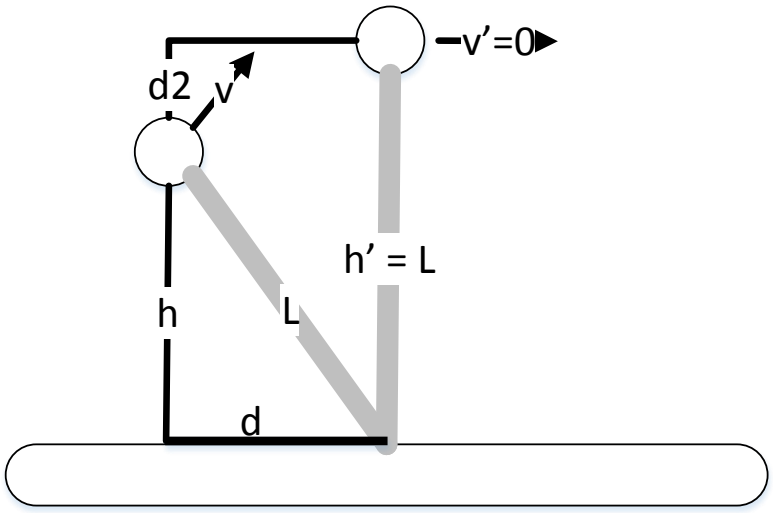

Figure 7.8.1: Target Inverted pendulums model.

In order to find the free inverted pendulum the value d_2 is calculated Figure 7.8.1 and Equation 7.8.5. This parameter is the height that the robot has to go up in order to have a $v = 0$. This increase of height can be transformed in energy Equation 7.8.3 and 7.8.4. This Target increase of the height is a fusion between the height needed to reduce the kinetic energy to 0 Equation 7.8.3 and the lost energy with respect the energy at the vertical position Equation 7.8.4.

The distance $Lpendulum$ has a maximum and minimum value that depends on the d_2 and the $TimetoLastChange$.

$$TargetE_1 = EKinetic + ExtraE * K_1 \tag{7.8.1}$$

$$TargetE_2 = Lpendulum * W * G - EPotential + ExtraE * K_2 \tag{7.8.2}$$

$$Targetheight_1 = \frac{TargetE_1}{W * G} \tag{7.8.3}$$

$$Targetheight_2 = \frac{TargetE_2}{W * G} \tag{7.8.4}$$

$$d_2 = K_3 * Targetheight_1 + Targetheight_2 * (1 - K_3) \tag{7.8.5}$$

7.9 Inverse Kinematics

When the target pendulums are fixed it is necessary to calculate the configuration of the motors, so that the robot acquires the desired position.

The procedure is the same for the 2 pendulums. Each pendulum will describe the position of the motors, for this reason, only on of them will be described.

The first thing to do is to convert the target inverted pendulum to a position in referenced to a O_0, this conversion is only to subtract the target pendulum to the CoG.

Then, it is time to calculate the inverse kinematics of the robot. The extraction from the kinematics is very complicate. For that reason an approximation used.

This approximation starts to transform from O_0 to O_1 or O_5 the desired position.

The configuration of the robot makes that if the robot is referenced to the O_1 or O_5 the position of the motors 3 or 6 depends of Z and Y. This fact has as result of the Equation 7.9.1.

$$PositionMotor_{3\&6} = \pm atan\left(\frac{|Targetpoint_Z|}{|Targetpoint_Y|}\right) - Pitch \pm \pi \tag{7.9.1}$$

In this moment the remaining can be simplified as a triangle as in the Figure 7.9.1 such that the input of this triangle is the L and the α_1.

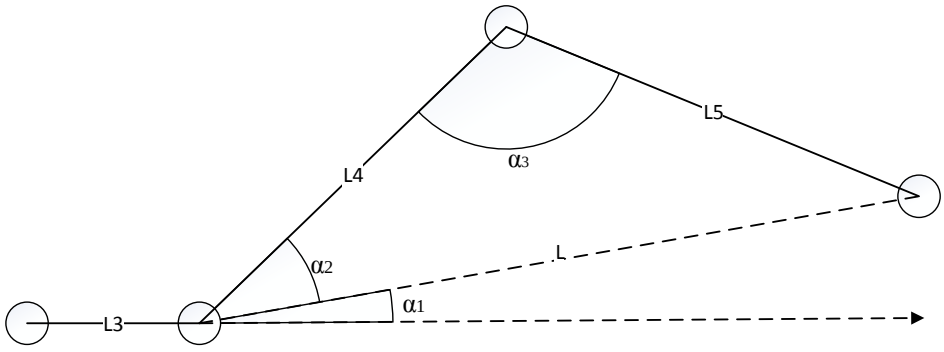

Figure 7.9.1: Inverse kinematiocs of the leg and the foot of the robot.

The L is extracted from the longitude of the pendulum as in the Equation 7.9.2.

$$L = \sqrt{Targetpoint_X^2 + Targetpoint_Y^2 + Targetpoint_Z^2} \tag{7.9.2}$$

$$\alpha_1 = Roll \tag{7.9.3}$$

Then using the inverse cosine theorem the target position of the motors 1 and 2, or 4 and 5 are described in the Equations 7.9.4 and 7.9.5.

$$\alpha_2 + \alpha_1 = PositionMotor_{2\&4} = acos\left(\frac{L^2 + L4^2 - L5^2}{2\,L\,L4}\right) + \alpha_1 \tag{7.9.4}$$

$$\alpha_3 = PositionMotor_{1\&4} = acos\left(\frac{-L^2 + L4^2 + L5^2}{2\,L5\,L4}\right) - \pi \tag{7.9.5}$$

Chapter 8

Results

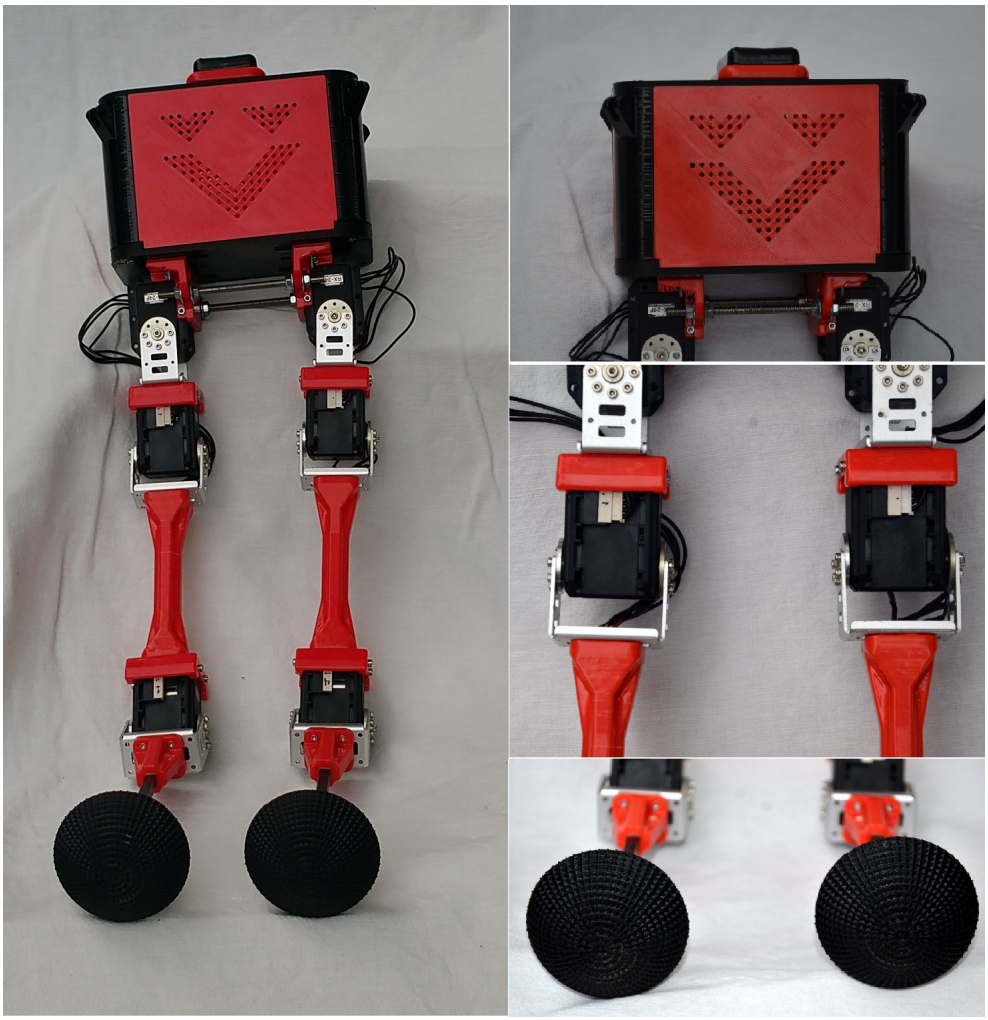

Figure 8.0.1: Photography of the standard version of the robot.

The final result of the construction of the robot standard and slim version can be seen in the Figure 8.0.1 and 8.0.2. As it can be seen, the robot is constructed using a 3d printer. This makes the robot to be light and compact. In all the possible parts a steam bath of acetone has been applied, to improve the appearance and increase the inter layer cohesion.

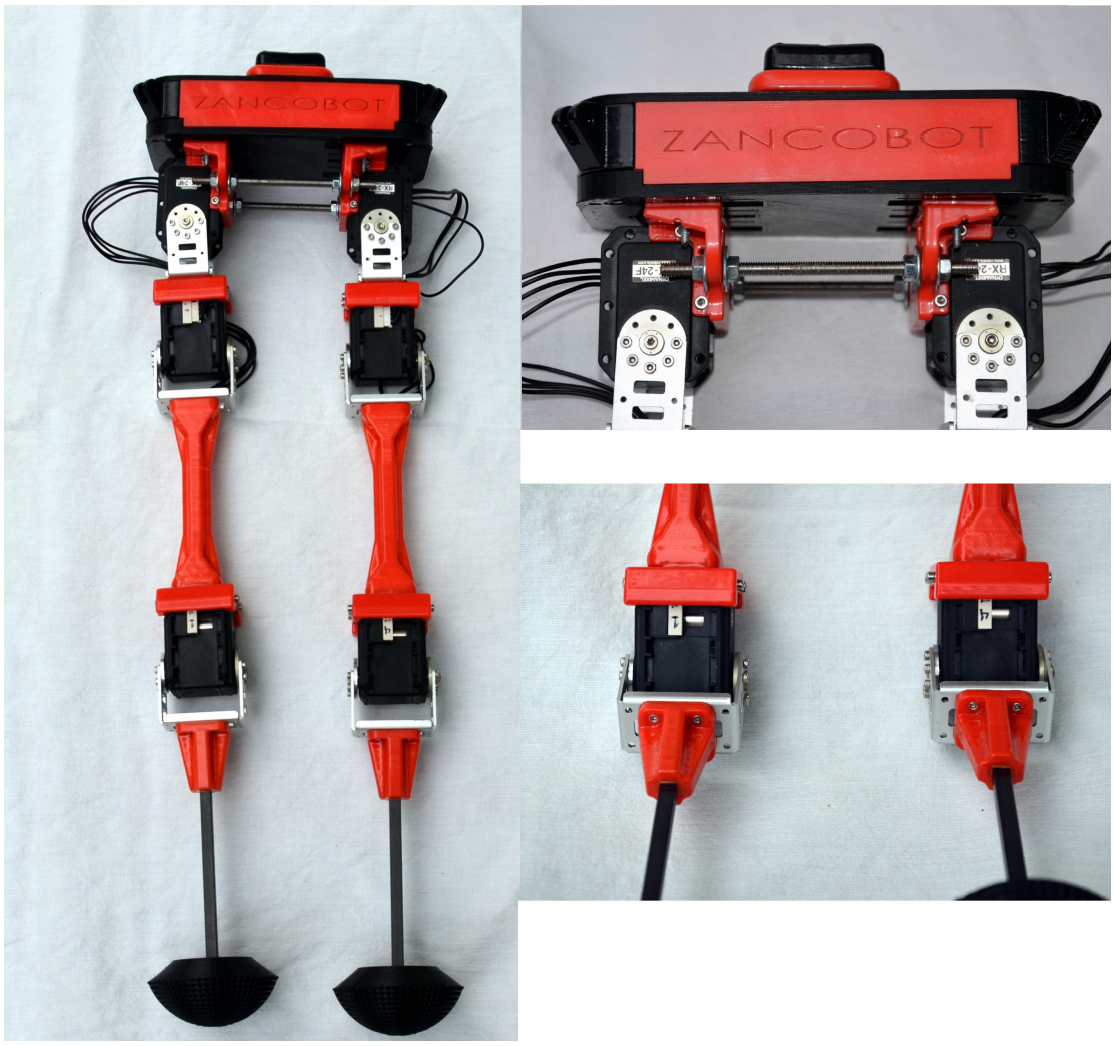

Figure 8.0.2: Photography of the slim version of the robot.

The *Stability Assistance* it is constructed Figure 8.0.3, it is enough robust to hold the robot and also light to be carried easily, the utilization of soft mat increase the grip of the robot that help the robot.

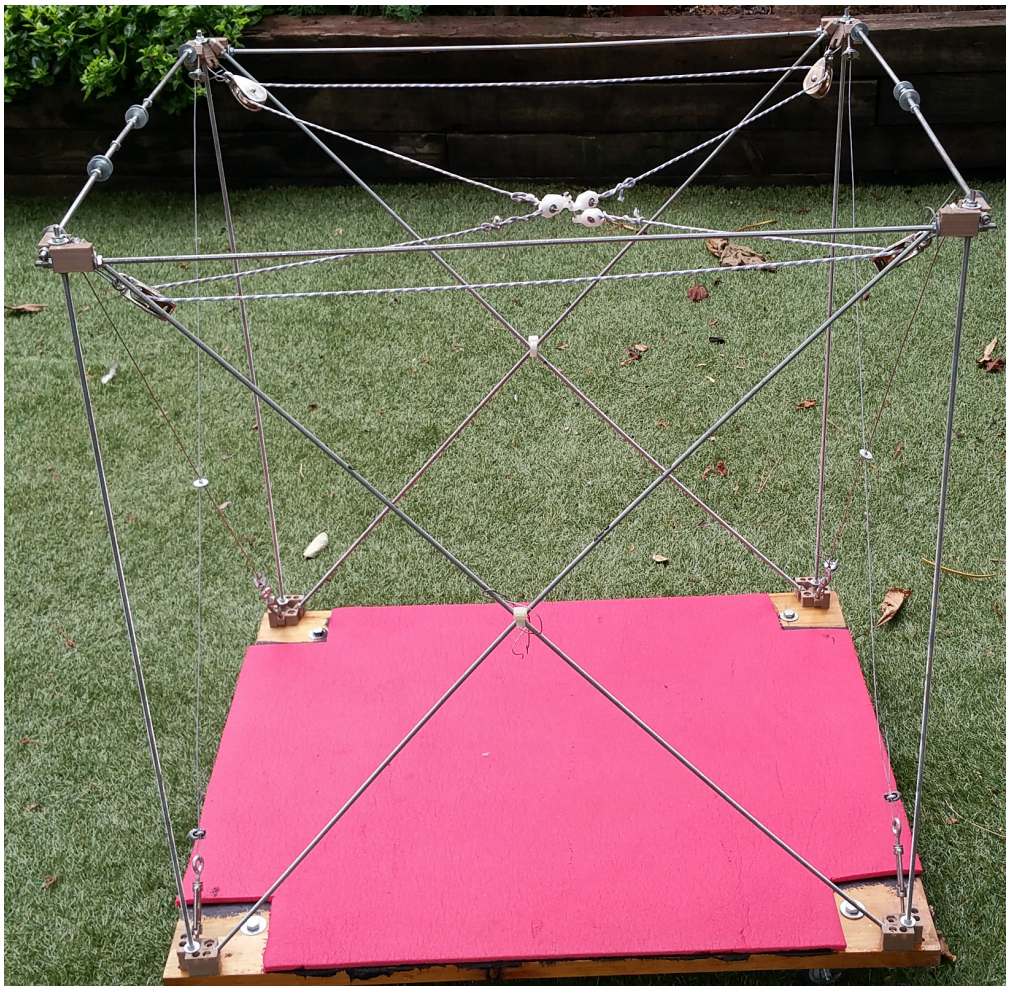

Figure 8.0.3: Photography of the Stability Assistance.

8.1 Mechanics

The mix of different construction materials in the extremities make them strong, light and robust to impacts. The buckling of the rigid parts it is very low and this helps to the control of the robot. But the looseness of the motors is significant and make the tremble when the one motor is under high torque.

The body of the robot is robust enough to carry all the electronics and light enough to not prejudice the behaviour of the robot dynamics.

8.2 Electronics

The electronics of this robot have some goods features such as: fast boot for all the electronics and exception of the Odroid that it is 20 - 30 seconds. The electronics are also electrically robust. Also the communication is fast and the rate of the communication error is lower than 0.01

Regarding the heat generation of the robot it is located in the motors and in the DC/DC Converter for The Odroid power supply and the motors. The heat generated in the DC/DC Converter is dissipated with out any problems, but the dissipation of some motors can be not enough if the robot have prolonged use under high load. More specifically the motors that suffer this problem are the motors 1 and 4 that are the motors located in the Leg because they carry more load than the rest of the motors.

8.3 Test Algorithm

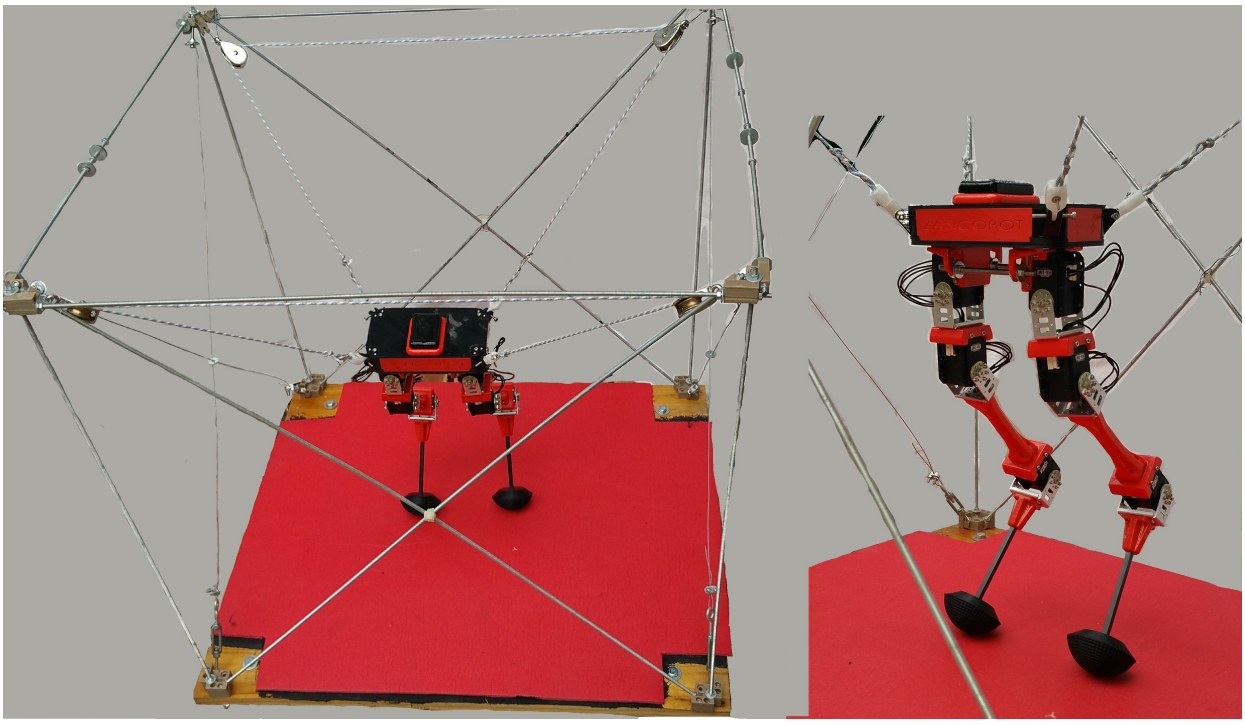

Figure 8.3.1: Photography of the Stability Assistance with the Slim version of the robot.

The *Test Algorithm* is tested using the slim version tied to the *Stability Assistance* . The test proves that the communication between all the parts of the robot, is fully functional. The lecture and actuation of all the motors work perfectly and it can also be seen that the

estimation of the angular position and velocity is good. It is tested with all the possible errors and they are detected correctly.

The estimation of the position of the joint and all CoG the CoG correct in any position. This is checked by measuring the estimation position of the joins compared with the real measurements. The correct estimation shows also that the IPM is computed correctly.

The frequency of calculation of this *Test Algorithm* is at 20hz and it is fast enough to control the robot.

When robot runs this *Test Algorithm* , it can be seen that attempts to put the foot in an area in a distance, such that, to remove the kinetic energy of the robot, which demonstrates that the robot can be used as a platform for development of *Inverted Pendulum Foot Placement* algorithm. The robot needs to be tied to the *Stability Assistance* in order to not fall down. The robot is not reliable to stay up without external help.

Chapter 9

Conclusions

After some tests over various configurations, it can be observed that the robot's mechanics are light and compact which was one of the objectives of the robot. Also after these tests the mechanics remained intact. For all these reasons the mechanics of the robot accomplish the mechanical objectives. Regarding the electronics of the robot, they are very reliable on extracting the information from the sensors and were able to sent the control actions to the motors periodically with out any problem.

Standard configuration is a functional autonomous robot but the slim version has better dynamics and the GUI was helpful on developing the system. This made possible to develop the system better using the slim version and then finally test it in the standard configuration.

The *Stability Assistance* is very usefully, because it provides a good grip and the use of ropes help the development of the system. It is also helpful on developing algorithms as it can change the dynamics of the robot.

When control algorithm runs, it can be seen that the CoG of the robot is calculated correctly as well as the IPM. The robot tries to put the tip, and has the enough force and speed to move the robot with out problems. For all these reasons it is proven that the robot can be controlled by *Inverted Pendulum Foot Placement* algorithms.

This project also helped me to develop and apply the knowledge learned in the master's degree in Robotics and Automatic Control as: Kinematics, Dynamics, Control Robotics, Modelling, Identification, Sensors, Instrumentation, Communication, Planning and Implementation of Robotic Systems, Robotics Optimization, Linear Multivariable Control Systems, Embedded and Real Time Systems, Nonlinear Control Systems and Geometric Foundations for the Design of Robots.

Future improvements

There are a lot of future improvements of the robot. Some of the most important are the following:

- Use the RX-64 Dynamixel motors instead of RX-24f. Although the motor is powerful enough to move the robot, the use of a better motor will improve the robot movement.
- Implement multiple IMU sensors. The use of multiple IMUs will improve the robot rotational position estimation.
- Use plastic injected or carbon fiber instead of printed plastic. The properties of the injected plastic are better that the printed plastic but the cost of use injected plastic is higher.
- Implement an *Inverted Pendulum Foot Placement* algorithm. The implementation of this algorithm will attempt to make the robot walk.
- Implement contact sensor in the tips of the robot. This will improve the detection of tip of the robot.
- Make the tips of the robot of rubber. The rubber have high grip and will make possible to the robot walk in different surfaces.

Budget

This section will talk about the different costs of the project: material cost, software, printing, power consumption, staff cost and depreciation.

Material cost

The total material cost of the robot and all complementary equipment is 1484.16 €, it is divided into electronics cost and mechanics cost Table 9.0.1. The disaggregation can be seen in the Table 9.0.2.

Concept	**Amount(€)**
Electronics	1149.51
Mechanics	334.65
Total	1484.16

Table 9.0.1: Material cost.

Electronics

Concept	**Unit cost (€/u)**	**Units**	**Cost (€)**
Dynamixel rx-24f	129	6	774
9 DOF Razor IMU - AHRS	69	1	69
Arduino	40.5	1	40.5
Odroid U3	56.1	1	56.1
16GB eMMC	25	1	25
MAX485	5.21	1	5.21
Baterry 3S	23.4	1	23.4
Baterry 2S	15.6	1	15.6
LT1084CP-5	11.5	1	11.5
Battery charger	15.2	1	15.2
Topos plate	6.4	1	6.4
Connectors	32	1	32
Wires	21	1	21
Power supply 700W	39.9	1	39.9
Pasive electronics	12.3	1	12.3
Leds	2.4	1	2.4
Others			25.4
Total			1149.51

Mechanics

Concept	**Unit cost (€/u)**	**Units**	**Cost (€)**
Abs coil 1.75mm 1kg	22.5	3	67.5
Dynamixel frame FR07-H101	30.5	6	183
Carbon fiber square bar 1m	11	2	22
M2 screws and nuts	0.1	100	10
M2.5 screws and nuts	0.08	100	8
M3 screws and nuts	0.03	100	3
M4 screws and nuts	0.05	25	1.25
M5 screws and nuts	0.1	25	2.5
M5 screw rod 1m	2.9	7	20.3
Acetone 1L	4.6	1	4.6
Wooden panel	8	1	8
Castors whhels	4.6	4	18.4
Wire tensioner	2.1	4	8.4
Metal wire 1m	0.5	10	5
Nylon cord 1m	1	5	5
Liquid silicone 1kg	6.1	1	6.1
Pulley	4.5	4	18
Others			11.1
Total			334.65

Table 9.0.2: Mechanics and electronics cost.

Software cost

The Software cost is 0€ because all software used is free.

Concept	Amount (€)
Ubuntu 14.04.5 LTS	0
XUbuntu 14.04.5 LTS	0
Pycharm	0
SolidWorks Student Edition	0
Proteus Design Free Version	0
Sharelatex	0
EAGLE Free Version	0
Arduino Ide	0
Microsoft Visio Student Edition	0
Repetier-Host	0
Python 2.7	0
RoboPlus	0
Total	0

Table 9.0.3: Software cost.

Printing cost

The Printing cost of the 3 copies is 195€.

Concept	Quantity	Unit cost ((€)/u)	Amount(€)
Printing and binding TFM	3	65	195
Total			0

Table 9.0.4: Printing cost.

Power consumption cost

The use of the equipment has associated power consumption Table 9.0.5. Then in the Table 9.0.6 is calculated the monetary cost related to this power consumption.

Concept	Power (W)	Total hours (h)	Energy (KWh)
Lighting	200	500	100
PC	300	400	120
Odroid	10	200	2
Battery charger	25	40	1
3d Printer	250	100	25
Robot supply	300	120	36
Total			284

Table 9.0.5: Power consumption cost.

Concept	Calculation	Amount(€)
Hired power	10kW x 8 months x 4.103364 €/kW	328.26912
Cost consumption	284kWh x 0.145562€/kWh	41.339608
		369.6
Electricity tax	369.6 x 1.05113 x 4.864%	18.89697183
Total		388.50

Table 9.0.6: Calculations of the power consumption cost.

Staff cost

The staff cost is divided depending on the work required and it has an amount of 15795€.

Concept	Price hourly (€/h)	Total hours(h)	Amount (€)
Feasibility study	25	100	2500
Hardware Development	20	130	2600
Software Development	20	210	4200
Equipment assembly	15	80	1200
Memory writing	15	110	1650
Before SS			12150
SS (30%)			3645
Total			15795

Table 9.0.7: Staff cost.

Depreciation

In the total cost of the project it also has to be taken into account the cost associated to equipment depreciation as can be seen in the figure 9.0.8.

Concept	V_0 **(€)**	V_r **(€)**	**Useful life** ***(Years)***	**Use** ***(Years)***	**Depreciation (€)**
PC	1100	350	5	0.66	99.00
3D pritner	900	300	2	0.66	198.00
Dynamixels	774	200	5	0.66	75.77
Dremel	70	10	5	0.66	7.92
Multimeter	80	10	5	0.66	9.24
Drill	75	10	5	0.66	8.58
Electronic welder	60	30	2	0.66	9.90
Batterys	39	0	2	0.66	12.87
Power supply	39.9	10	10	0.66	1.97
Total					423.25

Table 9.0.8: Depreciation cost.

Total budget

Finally the total cost of the project taken out the staff cost and the depreciation before taxes is 2067.66 €. The taxes have a cost of 434.20 €. And adding the staff cost and the depreciation the total cost of the project it is 18720.12 €.

Concept	**Subtotal(€)**
Material cost	1484.16
Software cost	0
Printing cost	195
Power consumption cost	388.50
Total before taxes	2067.66
IVA (21%)	434.20
Total after taxes	2501.86
Staff cost	15795
Depreciation	423.25
Total	18720.12

Table 9.0.9: Total budget.

Acknowledgements

I thank all those friends and family who supported and helped me during the development of this project, also I appreciate the advices of my tutor of the project Manel Velasco.

Bibliography

[1] Aaron D. Ames, Robert D. Gregg, Eric D.B. Wendel and Shankar Sastry: *Towards the geometric reduction of controlled three-dimensional bipedal robotic walkers.* University of California, 2006.

[2] Aaron D. Ames, Robert D. Gregg and Mark W. Spong: *A geometric approach to three-dimensional hipped bipedal robotic walking.* 46th IEEE Conference on Decision and Control, 2007.

[3] Abdallah, Muhammad and Ambarish Goswami: *A biomechanically motivated two-phase strategy for biped upright balance control.* IEEE International Conference on Robotics and Automation (ICRA), 2005.

[4] Arthur D. Kuo, J. Maxwell Donelan and Andy Ruina: *Energetic consequences of walking like an inverted pendulum: Step-to-step transitions.*

[5] Christine Chevallereau, J. W. Grizzle, Fellow IEEE and IEEE Ching-Long Shih, Member: *Asymptotically stable walking of a five-link underactuated 3-d bipedal robot.* IEEE TRANSACTIONS ON ROBOTICS, VOL. 25, NO. 1, FEBRUARY 2009.

[6] Goswami, A., B. Espiau, and A. Keramane: *Limit ycles and their stability in a passive bipedal gait.* IEEE International Conference on Robotics and Automation (ICRA), pages 246–251, 1996.

[7] Grant Baldwin, Robert Mahony, Jochen Trumpf Tarek Hamel Thibault Cheviron: *Complementary filter design on the special euclidean group se.*

[8] Hirai, K., M. Hirose, Y. Haikawa, and T. Takenaka: *The development of honda humanoid robot.* In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1998.

[9] Huang, Qiang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko, Hirohiko Arai, Noriho Koyachi, and Kazuou Tanie: *Planning walking patterns for a biped robot.* IEEE Transactions on Robotics and Automation, 17(3):280–289, June 2001.

[10] Jerry E. Pratt, Russ Tedrake2: *Velocity-based stability margins for fast bipedal walking.* Florida Institute for Human and Machine Cognition, Massachusettes Institute of Technolog.

[11] Jung-Yup Kim, Ill Woo Park and Jun Ho Oh: *Walking control algorithm of biped humanoid robot on uneven and inclined floor*. 2005.

[12] Khalil, Hassan K.: *Nonlinear Systems*. Prentice Hall, 3rd edition, December 2001.

[13] Mark Euston, Paul Coote, Robert Mahony Jonghyuk Kim and Tarek Hamel: *A complementary filter for attitude estimation of a fixed-wing uav*.

[14] McGeer, Tad: *Passive walking with knees*. IEEE International Conference on Robotics and Automation (ICRA), pages 1640–1645, 1990.

[15] Miomir Vukobratovic, Branislav Borovac: *Zero moment point - thirty five years of life*. International Journal of Humanoid Robotics, 1(1):157–173, 2004.

[16] Miura, H. and I. Shimoyama: *Dynamic walk for a biped*. International Journal of Robotics research, 3(2):60–74, 1984.

[17] Morris, B. and J.W. Grizzle: *A restricted poincar´e map for determining exponentially stable periodic orbits in systems with impulse effects: Application to bipedal robots*. 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, 2005.

[18] Popovic, Marko, Amy Englehart, and Hugh Herr: *Angular momentum primitives for human walking: Biomechanics and control*. In *Proceedings of the IEEE/RSJ International Conference of Intelligent Robots and Systems*, 2004.

[19] Pratt, Jerry E. and Russ Tedrake: *Velocity-based stability margins for fast bipedal walking*. In Diehl, Moritz and Katja Mombaur (editors): *Fast Motions in Biomechanics and Robotics*, chapter 14, pages 299–325. Springer.

[20] Robert Mahony, Senior Member, IEEE Tarek Hamel Member IEEE and Jean Michel Pimlin: *Nonlinear complementary filters on the special orthogonal group*. JUNE 2008.

[21] Robert Mahony, Sung-Han Cha, Tarek Hamel: *A coupled estimation and control analysis for attitude stabilisation of mini aerial vehicles*. November 20, 2006.

[22] Stelian Coros, Philippe Beaudoin, Michiel van de Panne: *Generalized biped walking control*. University of British Columbia, 2010.

[23] Tedrake, Russ, Teresa Weirui Zhang, Ming fai Fong, and H. Sebastian Seung: *Actuating a simple 3d passive dynamic walker*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[24] William Premerlani, Paul Bizard: *Direction cosine matrix imu: Theory*. 2010.

[25] W.Whittle, Michael: *Multiple View Geometry in Computer Vision*. ISBN 10: 0 7506 8883 1, fourth edition, 2007.

[26] Yamaguchi, J., N. Kinoshita, A. Takanishi, and I. Kato: *Development of a dynamic biped walking system for humanoid - development of a bipedal walking robot adapting to the humans' living floor*. IEEE International Conference on Robotics and Automation (ICRA), 1996.