

CHABOT

A Humanoid Robot for Educational and Exhibitional Purposes

Bachelor's thesis in Automation & Mechatronics Engineering, Electrical Engineering

KARIN DANKIS

ALEXANDER DAVIDSSON

ERIK HARDSELIUS

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2014
Bachelor's thesis 2014:05

BACHELOR'S THESIS IN AUTOMATION & MECHATRONICS ENGINEERING, ELECTRICAL
ENGINEERING

CHABOT

A Humanoid Robot for Educational and Exhibitional Purposes

KARIN DANKIS
ALEXANDER DAVIDSSON
ERIK HARDELIUS

Department of Applied Mechanics

Division of Vehicle Engineering and Autonomous Systems

CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2014

CHABOT

A Humanoid Robot for Educational and Exhibitional Purposes

KARIN DANKIS

ALEXANDER DAVIDSSON

ERIK HARDESELIUS

© KARIN DANKIS , ALEXANDER DAVIDSSON, ERIK HARDESELIUS, 2014

Bachelor's thesis 2014:05

ISSN 1654-4676

Department of Applied Mechanics

Division of Vehicle Engineering and Autonomous Systems

Chalmers University of Technology

SE-412 96 Göteborg

Sweden

Telephone: +46 (0)31-772 1000

Cover:

3D model of CHABOT with outer shell

Chalmers Reproservice

Göteborg, Sweden 2014

CHABOT

A Humanoid Robot for Educational and Exhibitional Purposes

Bachelor's thesis in Automation & Mechatronics Engineering, Electrical Engineering

KARIN DANKIS

ALEXANDER DAVIDSSON

ERIK HARDESELIUS

Department of Applied Mechanics

Division of Vehicle Engineering and Autonomous Systems

Chalmers University of Technology

ABSTRACT

The CHABOT project entailed the construction of a humanoid robot torso. The robot was to be used in the master program course TIF160 Humanoid Robotics as a laboratory material to be utilized in student group projects. The robot was given the name CHABOT by the customer together with a specified list of functional requirements and a request of an aesthetically pleasing exterior.

The main goal of the design was to provide an expandable platform that could be altered with additional functions in the future. This was achieved by using a CAN bus for communication and a combination of 3D printed parts, metal plates and threaded rods creating the supportive skeletal structure. A USB connection between the robot and a computer, allows students to control the entire structure. A polycarbonate plastic shell was crafted to create an engaging and attractive appearance.

The final CHABOT prototype fulfilled the stated goals of being both modular, re-creatable and aesthetically pleasing.

Keywords: Humanoid Robotics, Electromechanic, CAN, Robot Aesthetics

ACKNOWLEDGEMENTS

While working on CHABOT we recieved help form several different people. We would like express our gratitude to the following:

Krister Wolff & Ola Benderius for their guidance throughout the project.

Hans Odelius for his help in manufacturing the prototype PCBs.

Göran Stigler for his invaluable assistance with rapid prototyping and his enthusiasm for 3d-printing.

Reine Nohlborg & Jan Bragée for thier assistance and engineering guidance in the Prototype Lab.

NOMENCLATURE

API (Application Programming Interface) Is a set of functions that are implemented in the interface of a library or device.

CAD (Computer Aided Design) A virtual tool for modelling and simulating purposes.

CAN (Control Area Network) A serial bus for reliant communication between nodes.

ECAD Electronic CAD. A software suite used for designing circuits and PCB designs.

MCAD Mechanical CAD. A software suite for designing mechanical constructions.

MCU (Microcontroller Unit) A microprocessor with built in memory and io devices.

PCB (Printed Circuit Board) A physical circuit board.

RAM (Random Access Memory) A volatile memory for storing temporary data.

SPI (Serial Peripheral Interface Bus) A bus intended for connecting a MCU with external devices.

TTL (Transistor–Transistor Logic) A digital logic level standard for 5 V logic.

USART (Universal Synchronous/Asynchronous Receiver/Transmitter) A specialised digital unit intended for serial communication.

CONTENTS

Abstract	i
Acknowledgements	iii
Nomenclature	v
Contents	vii
1 Introduction	1
1.0.1 TIF160 Humanoid Robotics	1
1.1 Project Definition	1
1.2 Project Management	1
1.3 Objectives	2
1.4 Limitations	2
2 Design	3
2.1 Visualization Methods	3
2.2 Design & Aesthetics	3
2.3 Cost analysis	7
2.3.1 Component Evaluation Method	7
3 Procedure	8
3.1 Skeleton	8
3.2 Shell	9
3.2.1 Polycarbonate Vacuum Shaping Experiment	9
3.2.2 Mold Construction	9
3.3 Mechanics	12
3.3.1 Equations	12
3.3.2 Theoretical values	16
3.3.3 Component Sizing	17
3.4 Communication Protocol	17
3.4.1 Communication busses	18
3.4.2 Communication	18
3.4.3 CAN	19
3.4.4 Dynamixel servomotors communication protocol	20
3.5 Electronic	20
3.5.1 Architecture	21
3.5.2 Controller	21
3.6 Software	22
3.6.1 Firmware	22
3.6.2 Communication library	22
4 Results	23
4.1 Objectives	23
4.2 Cost Analysis	24
4.2.1 Component Evaluation Method	25
5 Discussion	26
5.1 Project Management & The Value Model	26
5.2 Cost Analysis	26
5.2.1 Component Evaluation Method	26
5.3 Construction/Design Methods	27
5.4 Solid mechanics	27
5.5 Skeleton	27
5.6 Shell	28

5.7 Electronics	28
6 Conclusion	29
6.1 Further work	29
Bibliography	31
A Component list	33
B Budget	37
C Reproduction costs	40
D Servomotor Component-to-Component Matrix	43
E Claw Component-to-Component Matrix	46
F Construction manual	48
G Controller manual	69

1 Introduction

The field of robotics is one of constant improvement. Since the first automated machines seen in the beginning of the industrial revolution, robots have steadily increased both in number and in complexity. Today there are robots capable of performing tasks seen only in science fiction films a decade ago. Yet, one of the biggest challenges is still ahead of us. That of creating a viable human sized machine that is convincingly able to mimic human movement and versatility. The humanoid robots are a technology whose proposed utility lies within the variety of tasks they can perform and their ability to work and operate in an environment primarily built for humans. In order to achieve this, additional studies are required in the field, and the CHABOT will serve as the base for many such studies.

The goal of this project is the creation of a fully functional humanoid robotic torso as well as a detailed manual to be used to recreate and mass-produce said robot for a fraction of the cost. The robot and its copies are to be used in educational purposes for experimentation and examination of student's knowledge in the field of humanoid robotics. It should also be able to be used in exhibitions to promote the university. The machine envisioned serves as an upgradeable platform created as to be easily expanded and improved upon in the future.

1.0.1 TIF160 Humanoid Robotics

The course TIF160 Humanoid Robotics is a part of the curriculum of the master program Complex Adaptive Systems given at Chalmers University of Technology. The goal of the course is for the students to achieve a higher understanding of the construction and implementation of humanoid robotics. TIF160 includes a group project utilizing a robot. The students must decide on a task of their own choice for the robot to perform. Thereafter they must program, redesign or change the robot to fit the specifications of their objective so that the task can be fulfilled.

1.1 Project Definition

The project is defined as such to further develop and improve upon an existing humanoid robot used for educational purposes. The robot will be utilized as a part of the course TIF160 where the students participate in a group project. The project consists of a humanoid robot used as a platform to solve a problem or manipulated to perform a certain task. The tasks may range from fairly simple to relatively complex, therefore the robot should be of general and modular design, which will enable the students to freely laborate. The design should also facilitate the possibility of further development of the robot in the future. It should have an aesthetically pleasing appearance so that it could be used as an advertisement tool for Chalmers at exhibitions and similar events.

1.2 Project Management

Taking into consideration the small size of the team executing a large project, there must be a solid project management objective. The team planned and executed the project management strategies in compliance to the teachings of the Value Model[1].

According to the value model planning should be divided into three different parts; defining the project (the red phase), establishing a plan (the amber phase) and managing the project until completion (the green phase). The teachings and strategies were then presented in a project plan in the very beginning of the project to the customers so that there were no misunderstandings of when and what was to be expected as the result of the project.

1.3 Objectives

The different parts of the project is divided into the following objectives that are requested by the customer.

- The robot will be of modular design to enable future additional work and allow the students to manipulate the construction easily.
- The robot will consist of a torso with a head and at least one arm.
- The robot will be compatible with a Microsoft Kinect device.
- The arm will be equipped with an exchangeable manipulator of some kind, such as a gripper or a humanoid hand.
- The robot will have an aesthetically pleasing design.
- The final cost of the prototype will be under a budget of 10 000 SEK and will be able to be reproduced for under 5 000 SEK.
- The head's camera, microphone and speaker will be handled by a computer through a USB connection and not by control electronics in the robot.
- The robot's weight will not exceed the weight that an average build student can carry the robot in a double lined paper bag.
- The head will be equipped with a LED matrix as a mouth.

1.4 Limitations

A humanoid robot is a very advanced construction. For the aforementioned objectives to be completed in the allotted time, a number of restrictions are required.

- The robot will only be equipped with one camera, though it will be possible to add stereo vision in the future.
- The head will only be able to tilt up and down.
- The robot will not have any legs, only the possibility to add them in the future.
- There will be no outer shell constructed for the arms.
- The robot will only be equipped with an API for C++.
- The robot will not be equipped with batteries.
- The instruction manual and assembly guide found in the appendix of this report are reserved for changes due to the shell not being able to be completed by the due date of the report.

2 Design

As stated in the project definition, the central objective of the project is to design and construct a humanoid robot that has the ability to mimic both the appearance and the movement of humans. The overall design of the robot must include all parts of the robot ranging from the construction of the skeleton to the software program.

2.1 Visualization Methods

To visualize the prototype before the construction begins, sketches and computer generated virtual models are made using various CAD-programs. To create the models of the electronic parts, ECAD-programs are used that are specifically designed to generate PCBs. While all the mechanical parts are created in a MCAD-program.

Both the construction of the mechanical and the electrical parts are first sketched on paper and then translated into a 2 or 3 dimensional model on the computer in the aforementioned programs. This has a number of advantages over regular schematics, as it is possible to utilize the programs features to simulate various manipulations to avoid construction mistakes.

The MCAD-program used for the skeleton and the shell is CATIA V5 R19 that is available to students registered at Chalmers. The servomotors and the claw, chosen by the group for the prototype, already have finished downloadable 3D models of the components. This makes it possible to size the neighboring components exactly after those components as well as creating a complete assembly of the different pieces into a virtual machine. The virtual machine is then manipulated in different ways to simulate movement in the entirety of the model and find clashes between parts reducing or eliminating the need of creating physical models for testing. An example of simulating the assembly can be found in Fig. 2.1.

To visualize the circuit board, an ECAD application is used. The chosen program for CHABOT is Altium designer. Altium is chosen for it's powerful and easy to use interface. A good ECAD program will provide features that assist in transferring a schematic drawing into a final PCB layout. With the help of rules, the ECAD application can provide help in indicating potential errors such as violated wires and room constraints. These rules also provide help in designing PCB layouts such as indicating missing connections, but also to help with differential pair routing and path length matching.

The end results of an ECAD suite is a set of PCB layers that can be exported to gerber files for mechanical milling or printed for chemical milling.

2.2 Design & Aesthetics

There are many different qualities that come into play when it comes to the design of the robot. The desired outcome is achieving an appealing look of the exterior that has a sound foundation, all recreating a resemblance of the human form. These two qualities are to be divided between the role of the skeleton and the role of shell. The skeleton is designed to carry the entire structure and resemble the shape of a human upper body. The shell covers the skeleton and all the electronic components at the same time conveying an aesthetically pleasing appearance.

An important aspect of humanoid robotics is for the robot to engage with and replicate the movement patterns of humans. For a successful interaction between humans and robots, humans have previously been shown to be more comfortable with robots that have an appealing appearance. According to the results of an experiment conducted at the University of Texas [2], people respond positively to a wide range of robots from simple abstract robots to extremely realistic human-like robots. The experiment was conducted on a group of 25 participants aged between 18 and 77. The partakers were shown images of humanoid robots ranging from the abstract to uncanny realistic silicon masked robots. They were then asked to rate the robots on a scale from 1-10 on their perception of the robots "human-like", appealing, familiar and eerie qualities. The data recovered

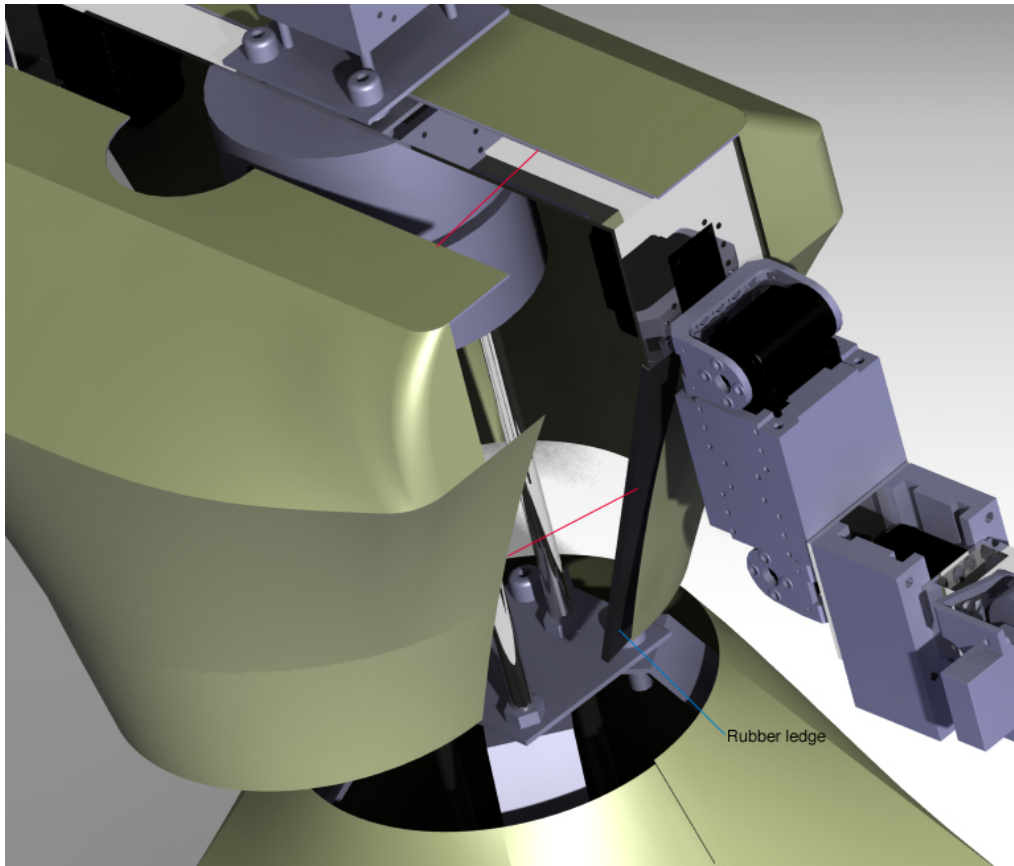


Figure 2.1: *Simulated assembly of torso in Catia v5*

showed that although the participants found that the more realistic robots were more “human-like” and slightly more familiar, they found that the more abstract robots were more appealing.

The study is related to a previous paper by Masahiro Mori discussing how to avoid the “uncanny valley”. The “uncanny valley” is a vernacular of the sudden dip in a plot derived of a study plotting the experience of familiarity of humanoid robots on the y-axis to the human-like resemblance on the x-axis. The study shows a sudden dramatic dip in familiarity when the robot has around an 80% human-like resemblance. This is called the “uncanny valley”. The robots are then experienced as uncomfortable or zombie-like. Mr. Mori recommends robot designers to avoid this valley by creating an exterior appearance that mimics the human form without recreating every single detail such as fingerprints and eyelashes. [3]

CHABOT was intended to have an abstract almost toy-like appearance with influences from both sci-fi and films from the 1950’s, with high familiarity with an appealing design. Thereby, in relation to both the study conducted by Mr. Mori and the further work of administered at the University of Texas, several design aspects were taken into consideration. An example of how the team opted to avoid the “uncanny valley” was to design the head with a resemblance of the human head’s shape but disregard any facial features. Another instance of this would be in regards to the shape of the torso. The torso consists of a breastplate and ”backplate” that do not take the shape of a human torso precisely, but the proportions are measured after a human that gives the illusion of the form. This can be seen in the sketches in Fig. 2.2 and Fig. 2.3.

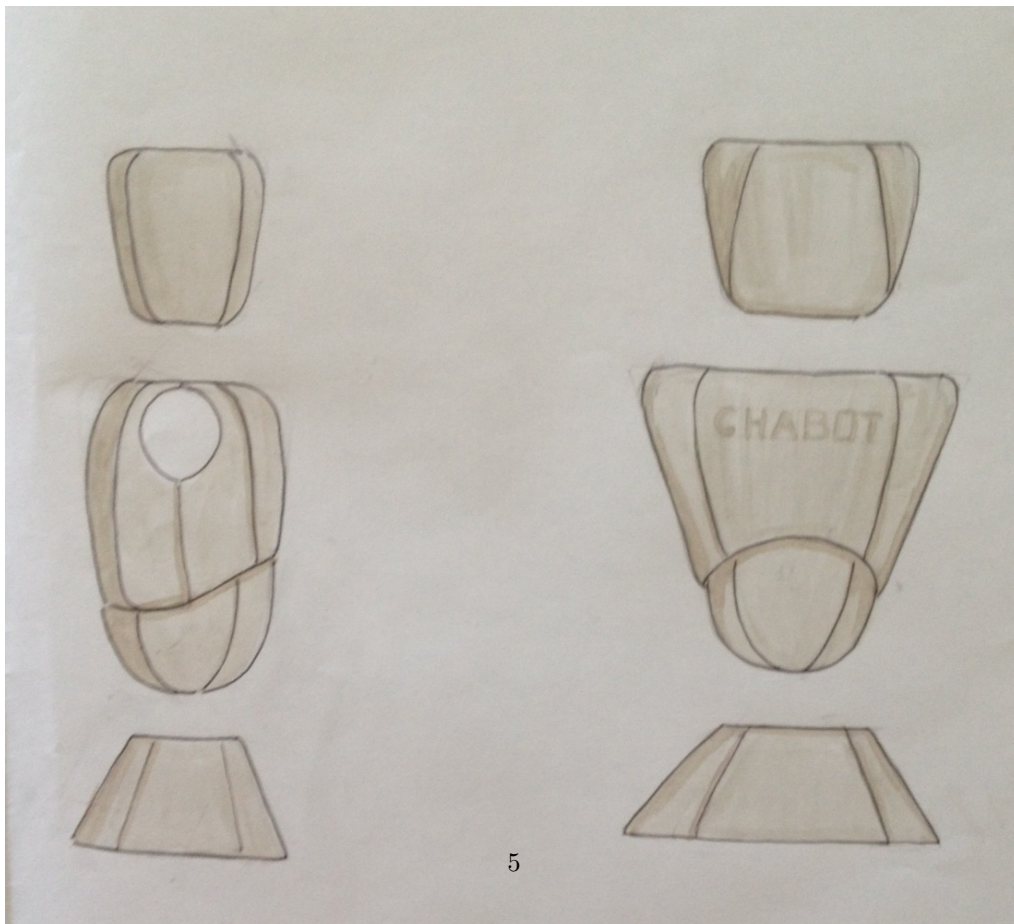


Figure 2.2: *First sketches of CHABOT*

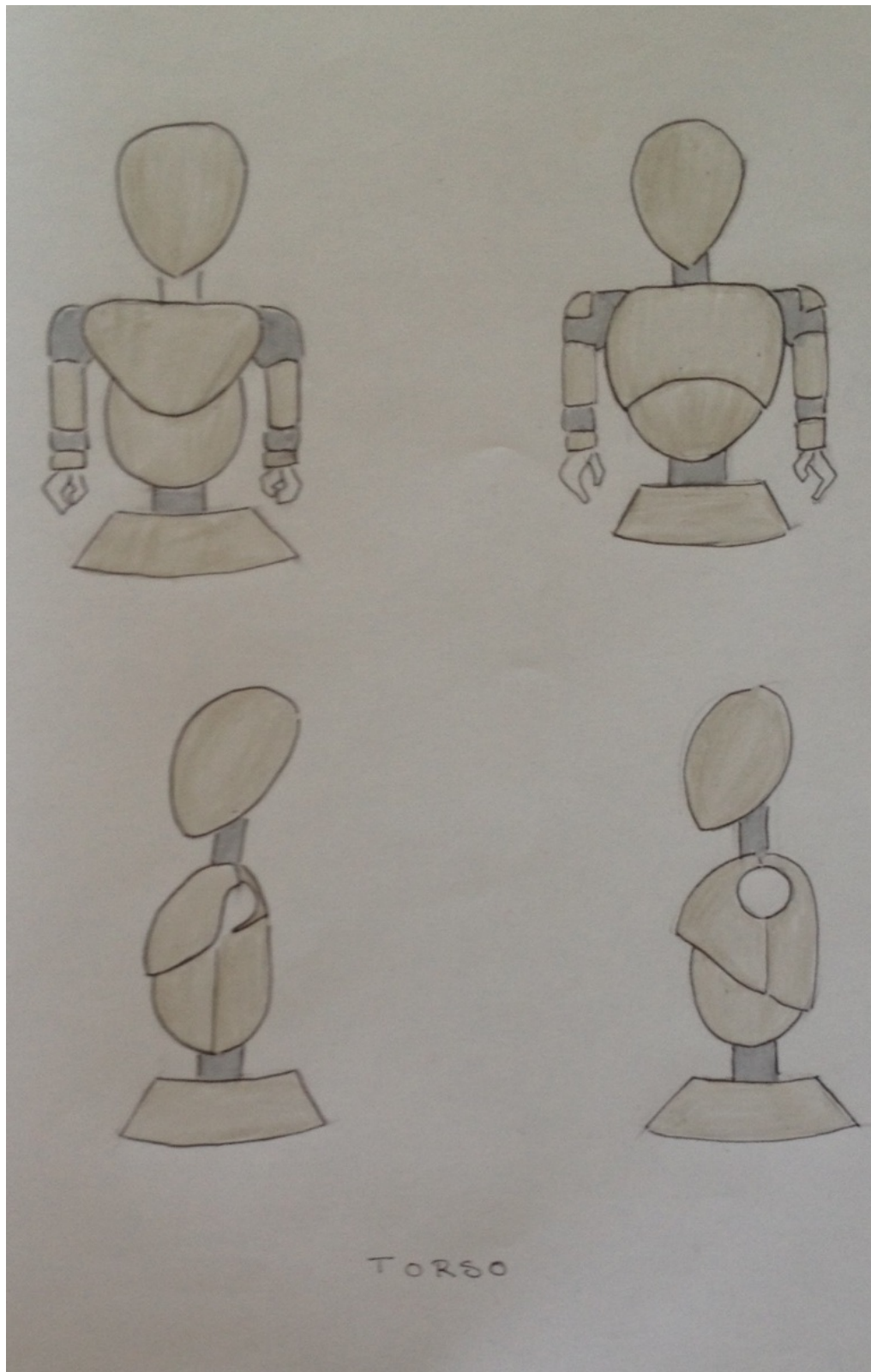


Figure 2.3: *Additional sketches of CHABOT*

2.3 Cost analysis

An important part of the project is to construct a prototype that is possible to recreate on a larger scale for a fraction of the cost per robot. The directive from the customer is to calculate the cost to reproduce six to eight robots for the price of 5 000 SEK per piece. The prototype robot is given a budget of 10 000 SEK.

To achieve this goal, it is imperative to consider every component and material that the robots will consist of. Since the reproductions are to be produced on a larger scale, there is the possibility to purchase materials in bulk to attain a lower final cost. Different suppliers are taken into consideration that have the possibility to offer discounts if one were to come back with a larger order. To calculate the final costs of the prototype versus the final costs of the reproductions a cost analysis is made.

2.3.1 Component Evaluation Method

To decide on which components to purchase, the appropriate method is to weigh different models of the component against each other before purchasing. The component-to-component matrix entails looking at all the attributes of the different components and evaluating which attributes are relevant to the task in hand and which component fulfills the desired specifications the best. This method was inspired by the Pahl and Beitz method and other general weighted evaluation methods, but was modified to suit the team's need.

The first step is to list all attributes of the components in a matrix to easily visualize the different specifications to thereby compare them. The next step is to name a component as a reference and then compare the remaining components by every attribute in the list to the reference in a separate matrix. If one of the components is superior to the reference on a specific attribute, the component is scored with a plus one point. In the same fashion, if the reference is superior to the compared component, it receives a score of minus one point. After repeating the process for every attribute and making every component the reference for a round of scoring, the points are added up. The component with the highest score is the supposedly superior component.

This evaluation method will not be implemented on all the components, only the ones where there are several attributes that come into consideration when deciding. Certain simpler components will only be evaluated and chosen when it comes to the price point or the performance.

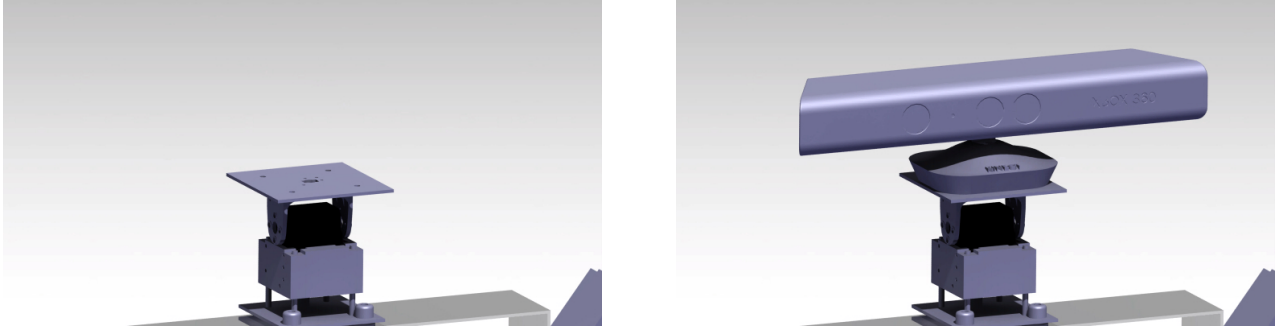


Figure 3.1: *3D Model of CHABOT's neck with and without a Microsoft Kinect*

3 Procedure

To achieve the undertaking of creating CHABOT, a number of different disciplines were involved. Mechanical engineering expertise was needed for tasks such as the skeleton and shell construction. Electronic engineering skills were applied in the form of creating controller cards for the electromechanic components. Extensive software engineering was involved such as creating the firmware for controller cards and software for the communication between CHABOT and a host computer. Beyond the purely technical engineering aspects, there was also a certain level of artistic proficiency required to fulfill the preset goal of an appealing design.

3.1 Skeleton

The design of the skeleton was intended to concentrate all forces working on the CHABOT into the spine and shoulders. The resulting technical drawings from the visualization methods, became the blue prints for the impending structure. The team constructed and assembled the skeleton with the resources offered by the Prototype Laboratory at Chalmers along with additional materials and components acquired.

The skeleton consists of a torso acting as the main load bearing construction, two arms, a head with a flexible neck and lastly a detachable stabilizing platform to serve as a base. The proportions of the CHABOT were inspired by the human body of one of the team members to make sure the CHABOT has a human like posture and reach. The larger parts of the skeleton, such as the torso and head, are constructed out of aluminum, steel plates, and threaded rods, while the arms are assembled out of 3D-printed rapid prototyping parts. The shoulder beam is constructed out of sheet steel due to the realization that an aluminum beam of the suggested size was not be able to carry the load of both arms and the shell without extensive bending.

The 3D-printed parts were created by the rapid prototyping system inherent in the Catia v5 program that enabled the parts to be printed directly from the CAD-drawing and schematics. This guaranteed a high repeating precision of the individual parts even if the parts themselves needed some slight polish to fit securely in the construction due to the inherent roughness of the printing procedure. The 3D-printed parts are both light and strong in relation to their size and foremost they are easy to both reproduce in smaller numbers making it a prioritized material and technique when aiming to replicate CHABOT.

The neck was initially meant to be able to rotate as well as tilt. Unfortunately, time constraints meant it had to be simplified and the rotation was cut from the final product as rotating the waist could solve the rotational movement. It was important that the neck was stable enough to be able to support the weight and size of the Microsoft Kinect. This required a redesign in the top plate of the neck to conform the preexisting screw holes in the Microsoft Kinect. See Fig. 3.1.

The waist of the CHABOT had to be redrawn several times during the construction phase due to the complexity of the construction. The final waist was a compromise of completing the project on time and a robust enough design and is composed of plastic parts revolving around a central axis of steel and brass.

The methods used to create the parts are as followed:

Metal parts:

- Drilling
- Lathe work
- Bench work
- Bending
- Cutting
- Sanding

Plastic parts:

- 3D printed rapid prototyping
- Sanding

3.2 Shell

The goal of CHABOT's design was an abstract, sleek minimalistic design with organic curves to realize an appealing appearance to achieve the desired effect discussed in the previous chapter. After a couple of different materials and construction methods were considered, a polycarbonate plastic shell was decided upon.

The polycarbonate plastic shell was formed through a process of heat and vacuum shaping. The process of shaping polycarbonate plastic entails first heating the plastic in an oven at 220 degree Celsius. When the plastic is soft enough and becomes elastic, the sheet is then removed from the oven and placed over the mold on a vacuum box very quickly before it hardens. The plastic is left to stabilize over the mold to take the shape of the mold. It is crucial that the mold is as close to perfect in shape and texture as possible because every imperfection will show.

3.2.1 Polycarbonate Vacuum Shaping Experiment

An initial experiment of the polycarbonate vacuum forming was conducted to find what shapes gave the best result when used as molds. In certain instances, the plastic creased when formed over sudden topographical changes in the mold. Round shapes with soft transitions proved to give the best outcomes as no folds around the edges were created. These results can be seen in Fig. 3.2 and Fig. 3.3. It was thereby beneficial to make the molds more organic in shape, slowly changing in shape to minimize the potential of unwanted lumps and creases. This also led to the change in the fastening between the shell pieces. The original design was that the shells would overlap and then be fastened with a simple screw or bolt. It showed that it would be much easier to fit the pieces together by adding a rubber ledge on the inside of one of shell pieces and have the corresponding one rest on that ledge. Then it was simple to align them and then fasten with a thin metal clip instead.

3.2.2 Mold Construction

The most important factor when creating the shells was that the molds had to be as close to perfect in shape as possible and the surface roughness had to be minimized. The primary plan was to create molds out of stacking levels of thin wooden planks cut into the outer shape of cross sections taken from the 3D models of the shells. This proved to be extremely time consuming and the time the team had available in the prototype lab at the end project did not allow for this method. It was then suggested by the staff at Chalmers to create the molds out of a solid block of wood and then shape them through sawing and sanding the desired shape of the mold. An image of this can be seen in Fig. 3.4 and a close up of the finish of the plugs can be seen in Fig. 3.5. This resulted in the molds not having the exact form of the model as the molds were shaped free handed by one of the team members.



Figure 3.2: *Polycarbonate vacuum forming initial experiment*

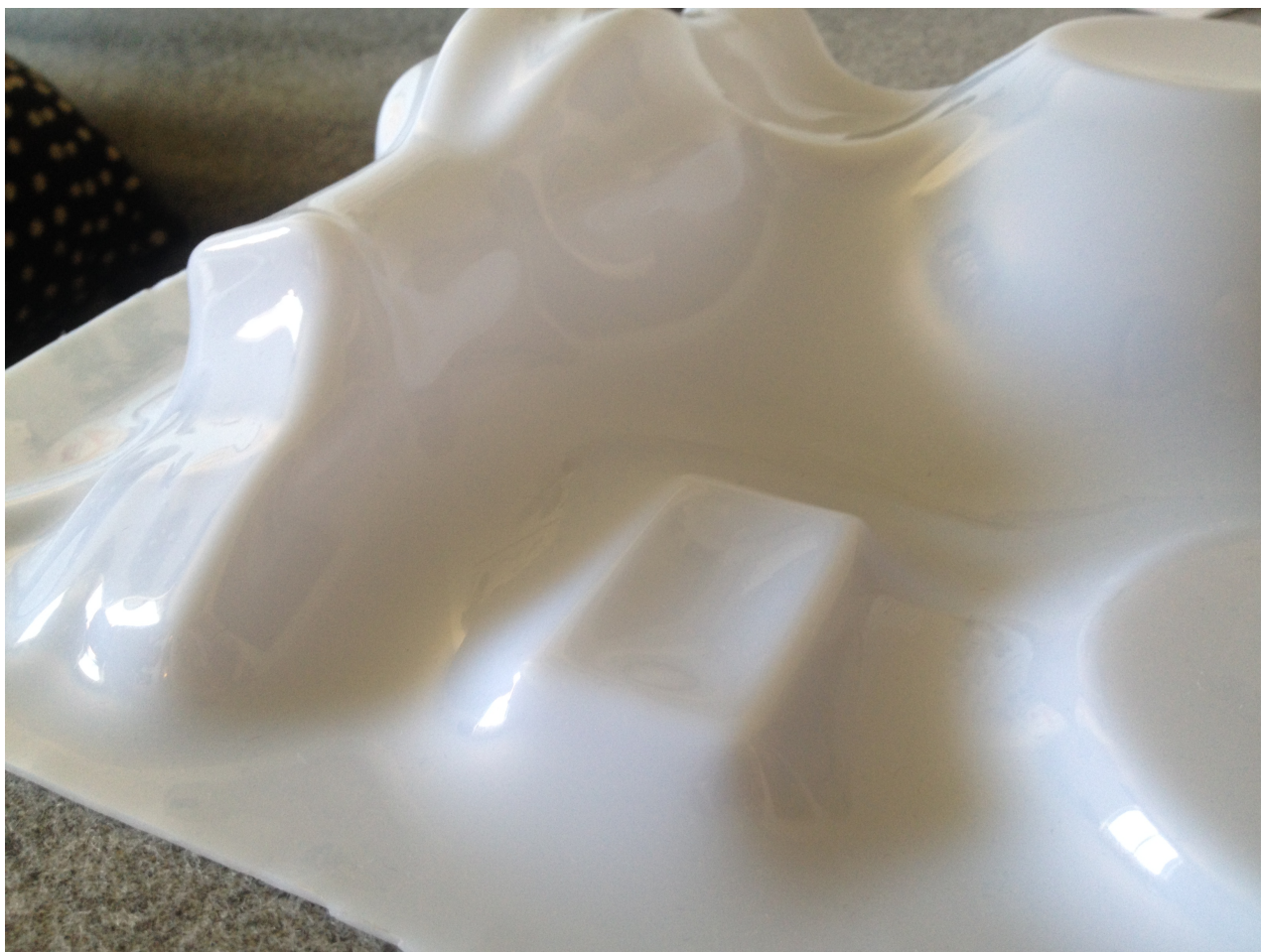


Figure 3.3: *Polycarbonate vacuum forming initial experiment close up*

To reduce the amount of plugs, the head was redesigned to have an identical front and back to be able to construct only one plug for the head that could be used for both sides. The shell covering the support was constructed out of thin aluminium instead of plastic, to not only to reduce the amount of plugs but also to create a more robust support system.

3.3 Mechanics

To make sure the construction does not fall apart due to it's own weight or the stress at the joints induced by the movement of the robot, several different calculations were made for the key parts such as the waist, neck, and shoulder joints. The parts not affected by movement were simply designed after recommendations from the engineering instructors at the Chalmers Prototype lab. To make sure the construction of the arms, neck and waist were structurally sound, the result of the estimates were assumed to lie below or between the mean value and the worst case method. The CHABOT is assumed to move slowly and in controlled movements, thus minimizing forces caused by acceleration and deceleration.

3.3.1 Equations

In order to make predictions on the mechanical properties of the robot, the following equations using Newton's laws of motion were made with the some simplifications. The physical model is reduced to a two dimensional problem which handles the momentum and the force along two axes. As the CHABOT's size is somewhat limited, no equations regarding mechanical fracturing were deemed necessary. Instead, the focus lied on the capabilities of the Dynamixel servomotors. All products are rounded to the third decimal using safe rounding. Load, mass and stress are rounded up and the servomotor's torque and material strength are rounded down. The calculations are performed using both stall torque/footnote The maximum instantaneous and static torque and operational torque.

m_b Mass shoulder bracket	A Waist
m_a Mass arm	B Shoulder
m_v Mass wrist	BC Shoulder bracket
m_c Mass claw	CD Arm
m_x Mass load	DE Wrist
	EF Claw
	GH Neck-Head

Mean value: In these calculations the mass of a given length is assumed to be placed in the center of the given length. The only exception is the work load that is calculated using the worst case method. The mean value method gives a more realistic value than the result of the worst case method.(Fig:3.6)

Worst case: In these equations the entire mass of a given length is assumed to be placed in such a way as to generate the maximum accumulated force possible. This is achieved by placing the mass at the far end of each element. This method is primarily used in calculations as it automatically calculates the maximum stress on the construction and the momentum applied on the servomotors. The maximum work load is calculated using the same method3.6.

By mechanical isolation of the different systems, such as the arm, shoulder and waist, it becomes possible to calculate momentum for each key joint. The momentum around the waist is calculated by doubling the momentum generated from a single arm holding the absolute maximum load in a horizontal position perpendicular to the chest of CHABOT at the stall torque. Thereafter, adding the momentum generated by the head when mirroring the movement of the arm results in the ultimate momentum generated at the waist(Fig:3.8). This result is equivalent to three times the stall torque of the Dynamixel servomotor.

The torques and loads affecting key joints calculated using Newtons second law of motion.

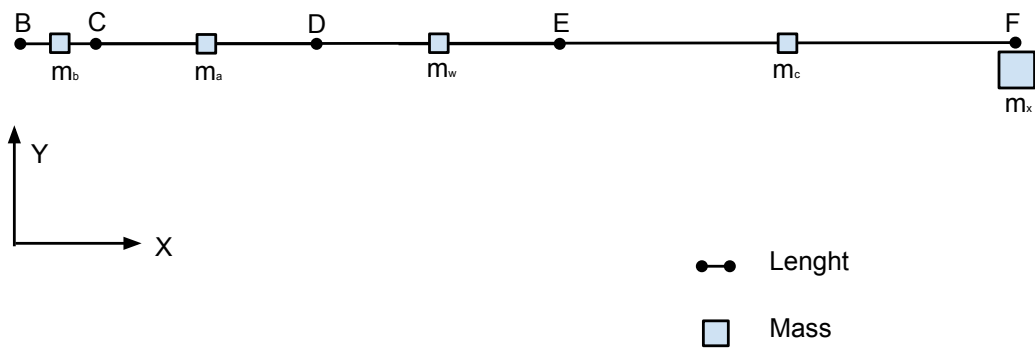


Figure 3.4: *Solid wood plugs of the torso front and head*



Figure 3.5: *Solid wood torso plug close up*

Mean value method



Worst case method

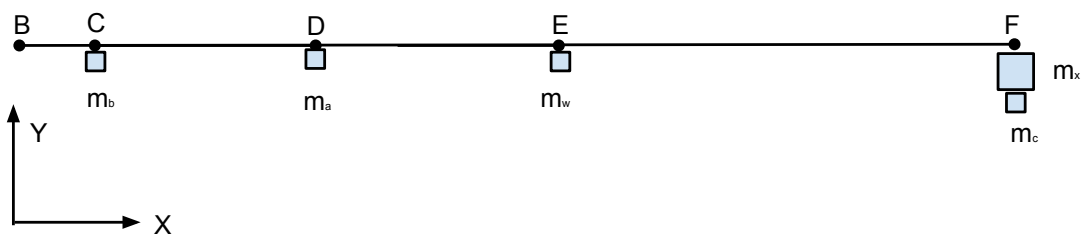


Figure 3.6: *Mean value / Worst case method*

Mechanical forces of the neck using worst case method:

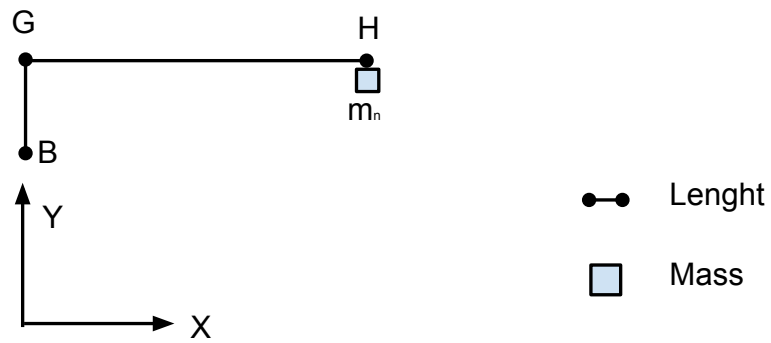


Figure 3.7: *Forces affecting the neck of CHABOT*

Mechanical forces on the waist using the worst case method:

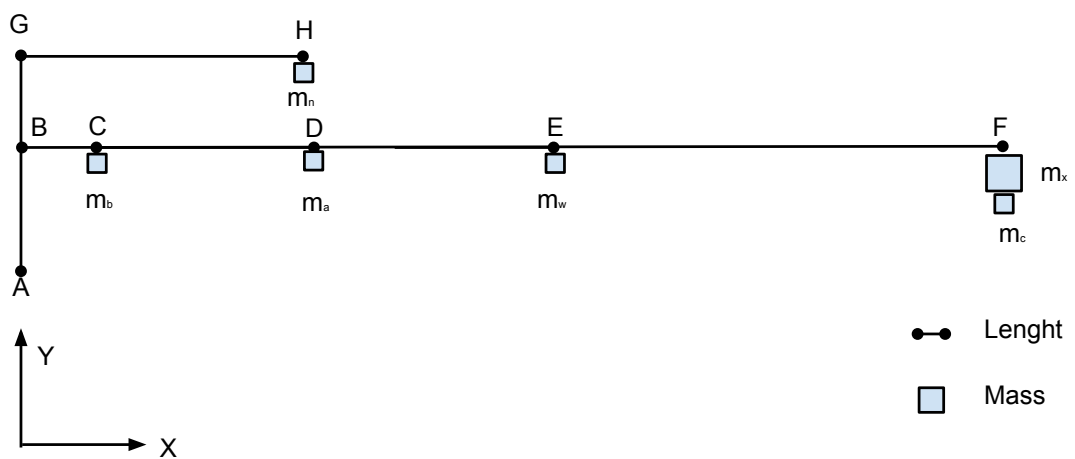


Figure 3.8: *Forces affecting the waist of CHABOT*

Stall torque (M_s): 1.53 Nm
 Operational Torque (M_o): 0.765 Nm
 Acceleration (g): 9.82

Shoulder joint:

The momentum effecting the shoulder joint.

$$M_B = \left[\frac{BC}{2} m_b + \left(BC + \frac{CD}{2} \right) m_a + \left(BD + \frac{DE}{2} \right) m_w + \left(BE + \frac{EF}{2} \right) m_c \right] g \quad (3.1)$$

$$M_{B\langle MAX \rangle} = [(BC)m_b + (BD)m_a + (BE)m_w + (BF)m_c] g \quad (3.2)$$

Operational Load:

Load allowed in normal movement.

$$m_o = \frac{M_o - M_B}{BFg} \quad (3.3)$$

$$m_{o\langle MAX \rangle} = \frac{M_o - M_{B\langle MAX \rangle}}{BFg} \quad (3.4)$$

Maximum Load:

Load allowed when CHABOT is stationary

$$m_m = \frac{M_s - M_B}{BFg} \quad (3.5)$$

$$m_{m\langle MAX \rangle} = \frac{M_s - M_{B\langle MAX \rangle}}{BFg} \quad (3.6)$$

Neck:

Momentum effecting the neck.

$$M_G = \frac{GHmn}{2} \quad (3.7)$$

$$M_G(max) = GHmn \quad (3.8)$$

Waist:

Momentum effecting the waist.

$$M_A = GHmn + 2 [M_{B\langle MAX \rangle} + BF(m_{o\langle MAX \rangle})] \quad (3.9)$$

3.3.2 Theoretical values

The equations above was used to calculate the theoretical required torque for the key joints as well as the lifting power of the CHABOT and the results are presented below.

Shoulder joint

The subsequent joints in the arm are subjected to less force making the shoulder the most important joint of the arm. Fig.(3.6)

The momentum affecting the shoulder using the mean value method: 0. 694 Nm (3.1)

The momentum affecting the shoulder using the Worst case method: 0. 954 Nm (3.2)

Operational Load:

Calculated load allowed in normal movement using the mean value method: 21 g (3.3)

Calculated load allowed in normal movement using the Worst case method: -59 g (3.4)

Maximum Load stationary:

Calculated load allowed when stationary using the mean value method: 258 g (3.5)

Calculated load allowed when stationary using the Worst case method: 178 g (3.6)

Neck

The neck is subjected to different forces depending on the current "head" mounted on the CHABOT. The following calculations assume the Microsoft Kinect being equipped. Fig.(3.7):

The momentum affecting the neck using the mean value method: 0.237 Nm (3.7)

The momentum affecting the neck using the Worst case method: 0.473 Nm (3.8)

Waist

Unlike the previous parts, the forces on the waist is not applied directly on a Dynamixel servomotor but on the support structure and axis used to change the facing of the CHABOT. Fig.(3.8)

The momentum affecting the waist using the Worst case method: 4.59 Nm (3.9)

3.3.3 Component Sizing

Sizing the parts for the required task demanded a balance of weight, material strength as well as an ease of machining and assembly. The main limiting factor in sizing components was the AX-12A Dynamixel servomotor, as the specified torque and size could not be altered. Thereby, the surrounding parts had to be constructed and fitted with their limitations in mind. All pieces printed using rapid prototyping were designed to have a snug fit with the AX-12A. The 3D printed pieces required another way of thinking when designing the parts than with the other mechanical constructions. Instead of adding material to an internal construction, the piece derives from a simple solid shape and then by removing material the piece takes on the desired shape. This process translates the constructing methodology from building parts into something more akin to sculpting.

3.4 Communication Protocol

Since CHABOT is intended to be of a modular design, there is a need for some sort of communication between the different systems. A communication protocol should help with message passing and to choreograph nodes within the design. In CHABOT there is a need for a protocol that can send control information to the different parts of the robot, but also return information such as sensor data and error messages such as a notice of overload. The protocol also needs to be expandable and able to handle additional systems as they are built and integrated into the CHABOT.

3.4.1 Communication busses

Communication can be done over a bus that can be seen as a motorway for information. A bus can either be parallel or serial, both have different uses and applications. A parallel bus is a mean of transferring data over multiple data pins including control and clock wires.

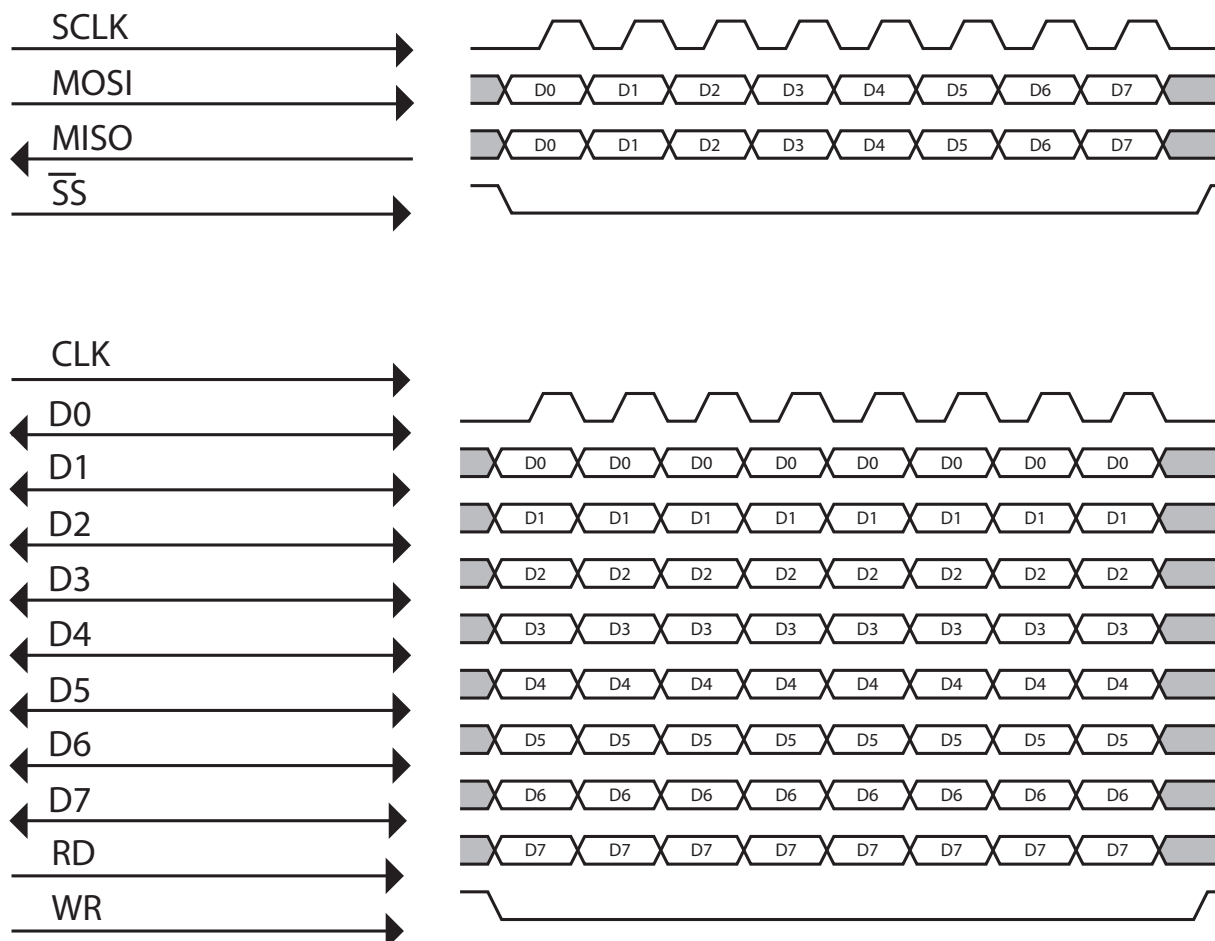


Figure 3.9: *A Serial SPI bus and a simple parallel bus*

A serial bus, contrary to a parallel bus, uses fewer data lines. Instead of one data line per bit, multiple data lines are multiplexed into one or a few data lines. The main benefits with serial buses are that they are easier to wire due to their lower pin count. The Fig. 3.9 compares a serial bus called SPI [4] with a simple parallel bus. This graph shows how higher transfer rates per clock cycles are sacrificed for fewer physical wires.

Communication over a serial bus can conform to different standards with different levels of abstraction such as RS485, Ethernet, CAN, etc.

3.4.2 Communication

In situations where there are requirements of communication between different computers or control units, there is a need of shared standard between the two.

A common way of illustrating these different standards is to use the OSI model.

- 7. Application** The application layer is where standards between different applications are defined. Examples: HTTP, Modbus, FTP

- 6. Presentation** This layer contains how to interpret received packages and in what way to format a package for sending.
- 5. Session** The session layer is the layer where authorization occurs. Examples: PAP, SOCKS
- 4. Transport** Managing flow control and optional error handling occurs in the transport layer. Examples: TCP, UDP
- 3. Network** The Network layer handles addressing of packages between different members in the same network. Examples: IPv4, IPv6
- 2. Data link** The data link layer is where the packages are broken up into frames and addressed to the correct controller. Examples: MAC, CAN
- 1. Physical** This is the wiring used to transport the individual bits in each package meant to be submitted. Examples: RS232, RS485

From this point on, the Presentation and Session layer will be ignored since it's not applicable to this report. [5].

RS485 & Modbus

RS485 is a multipoint serial bus using differential signals. Differential signaling is used to decrease sensitivity to environmental noise. RS485 is a physical layer and does not contain any standard for addressing, routing, collision detection or error correction. This means that the developers are required to add these features themselves. [6].

A solution is to use a Modbus. Modbuses are available in several different versions, the most commons types are Ethernet and a Master / Slave version. The Master / Slave version is the one used on RS485. In the rest of this text only the Master / Slave version will be discussed.

A Modbus has a simple protocol with one master and several slaves, where the master handles all communication. This avoids the problem with collisions in the bus, while prohibiting slaves from sending messages back to the master without a request.

Modbuses simply handle reads and writes to addresses on the slave node. This interface results in a simple yet powerful protocol[7].

3.4.3 CAN

CAN is an abbreviation for Controller Area Network and was designed by BOSCH for the automotive industry. CAN is a differential bus standard which implements both a data link and a physical layer. This means that it implements both an electrical standard and a means for sending blocks of data. CAN provides means to handle addressing and to detect errors in transmissions. CAN buses are restricted to sending up to 8 bytes of data per message, this means that there is a need for a higher level protocol to handle sending larger messages and interpreting what messages mean. An example of such messages can be found in Fig. 3.10.

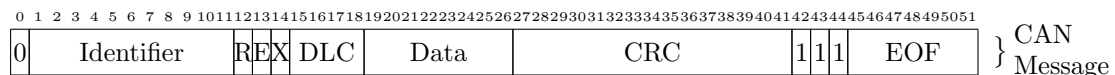


Figure 3.10: *Example CAN package*

CAN uses it's identifier to determine priority, a lower number means a higher priority. Another characteristic of CANs is that all nodes see all messages and the identifier is used by the microcontroller or computer to filter out the message of interest. CAN buses provide several ways of detecting errors to provide a safe communication between nodes [8].

3.4.4 Dynamixel servomotors communication protocol

Each Dynamixel servomotor contains a small memory area that contains all the different information parameters of the servomotor such as position, temperature, torque, etc. These are changed through a half-duplex serial protocol with TLL levels, meaning that the communication can only go in one direction at a time. A communication package contains a start sequence following with the ID, length, data and ending with a checksum as seen in Fig. 3.11.



Figure 3.11: *Dynamixel servo message*

Depending on the ID and the instruction, a status package is returned. This package contains the status and the potential parameter data as seen in Fig. 3.12. To broadcast a message to all nodes the ID 254 are used. [9].



Figure 3.12: *Dynamixel servo status message*

3.5 Electronic

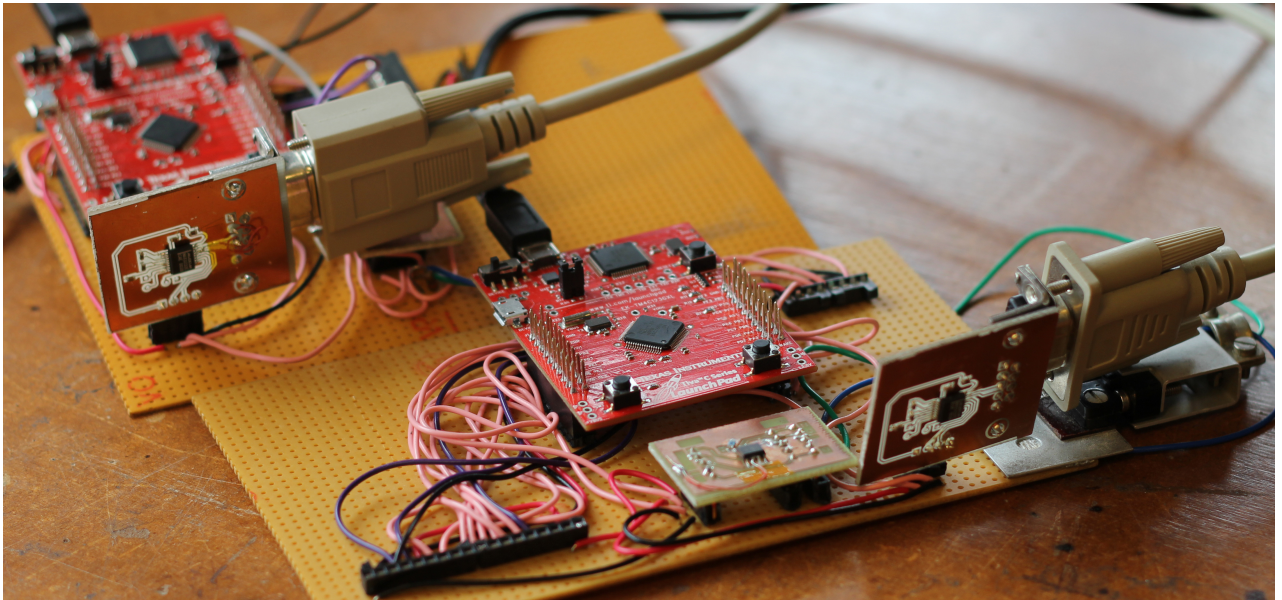


Figure 3.13: *Early prototype of the controller card*

The electronic system in CHABOT was designed to be an interface between a computer and the individual devices. Early versions of the steering electronics were developed using small interconnected modules on a veroboard¹ as seen in Fig. 3.13. Modules were designed to be a self-contained circuit board that provides a specific function used by CHABOT. The use of wiring allowed for easier modification during the development process. When all features were tested and verified, a PCB was manufactured.

¹A prototype board consisting of even distributed circular solder pads.

3.5.1 Architecture

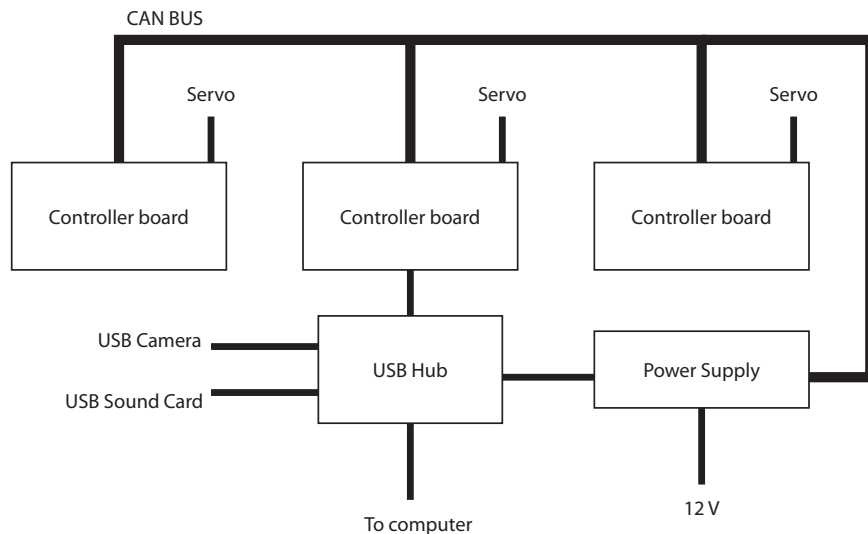


Figure 3.14: *CHABOT architecture*

The idea to allow an easy way to control different parts of the robot was important for the design of the electronic system. An initial requirement was that all data should be handled through one USB connector. This requirement along with the need to be upgradable resulted in an architecture that can be seen in Fig. 3.14. It was designed around two different buses. A CAN bus that transports the control commands between the different controller boards and a USB bus where devices such as cameras and speakers could be connected. One of the controller boards, also called master controller board, was connected to the USB to allow all communication to a computer go through one single USB cable. This design allowed for the connection of additional devices such as a Microsoft Kinect.

3.5.2 Controller

The controller boards were based around an ARM micro-controller evaluation board called "Tiva-C Launchpad". A Tiva has two extension connectors that allows for custom boards to be attached. These are in general called "booster packs". The CHABOT controller board was designed as a booster pack that provides communication through CAN and USB. This booster pack design can be seen in Fig. 3.15. The servomotors are controlled through a 5V half duplex protocol. Since the Tiva is a 3.3V device, a level shifter was required to allow communication with the servomotors. This resulted in the use of a tristate buffer from the 74HCT series. This device solves both the level problem and supports the half-duplex protocol.

CHABOT's controller was designed to be powered through its CAN bus connector. This provides both 12V and 5V for powering the servomotors and the Tiva board itself. The CAN bus was chosen to be used for communication between the different controllers. A simple protocol was designed featuring 3 different kinds of CAN messages: write, read and return. These messages allow one controller to manage another controller. In the first plan, a DSUB-9 connector was intended to be used for as a CAN bus, but due to its large size, a Firewire connector was chosen instead. A USB to USART adapter was added to allow a controller to be reached from a host computer. A FTDI FT230 provided a virtual serial port on the host computer to simplify development of a communication library.

A feature that was added was the different input and output devices. There were 2 analog input channels providing 12 bit resolution and 2 digital inputs and 2 outputs added to the design. These features were added to the controller's protocol despite the fact that there were no connectors added. A PWM signal was added for the reason that it was needed by the preassembled servomotor inside CHABOT's purchased claw.

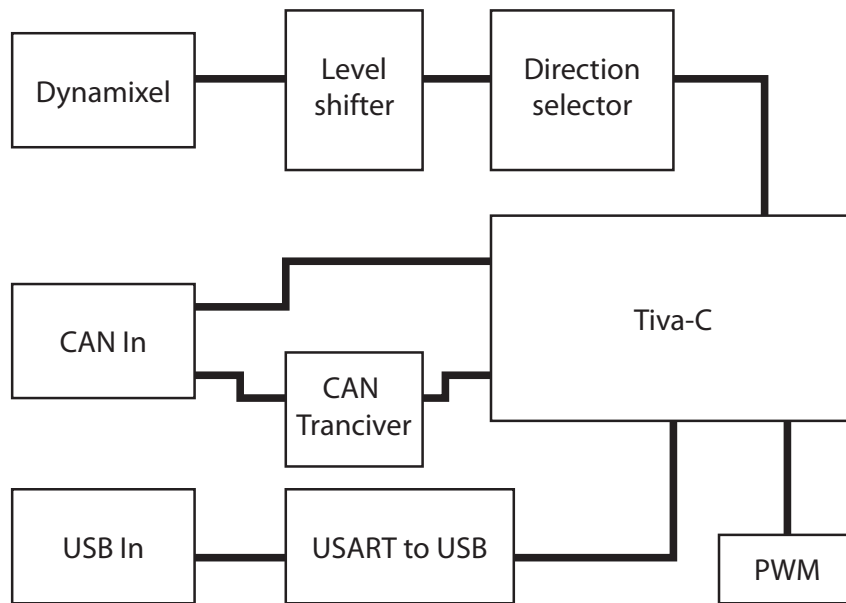


Figure 3.15: *Controller board*

3.6 Software

Since CHABOT was intended to be used for educational purposes, a design where the intelligence was moved to a host computer was chosen. CHABOT's software was split into two different software projects. One was the Firmware used by the controllers and the other was the communication library residing in the computer.

3.6.1 Firmware

CHABOT's firmware's main purpose was to provide an interface to the different devices. The Firmware was designed in a way that allowed the same code to be used independently whether a control board is a master or a slave node. To allow a simple expandable interface, a memory map-like solution was chosen. Each device's parameters represent a unique 16 bit address. The firmware was provided with parameters for controlling Dynamixel servomotors, digital in/out, analog in and PWM. This abstraction is used by the CAN protocol to transfer commands in a compact binary rather than sending a command as a string. A simple protocol was designed to transfer these read and writes. This protocol was also designed to be easy to extend with new types of messages and allowing the addition of new types of devices without modification of the protocol itself. For communication, a command interpreter was implemented to translate commands to a register write or to be sent to another controller.

3.6.2 Communication library

The communication library was designed as a glue to provide an interface to the controllers. This library was implemented in C++ using boosts and termios. The API was designed to be object oriented and to provide a simple interface to the different devices. A multithreaded design was chosen to support asynchronous communication. This allows the library to handle multiple commands simultaneously instead of waiting for each command to be finished. Another aspect was that this kind of design allowed for adding exception handling and to avoid buffer overflows.

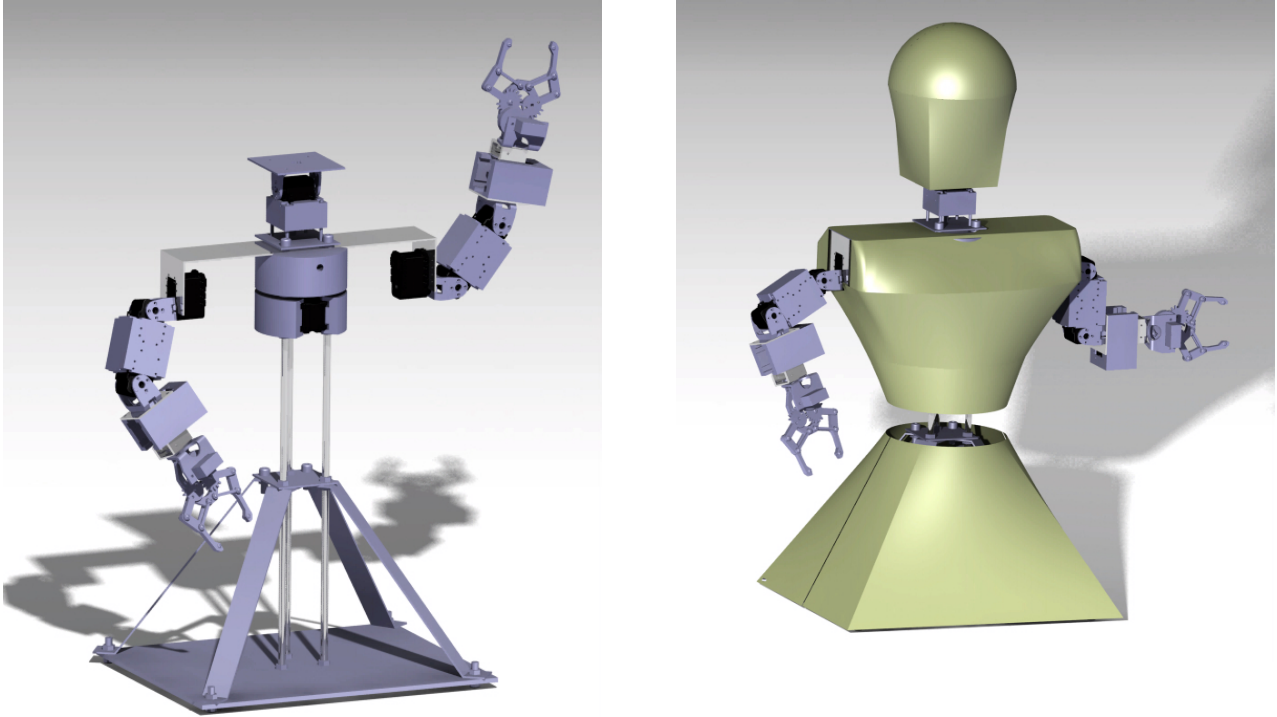


Figure 4.1: *3D Models of CHABOT*

4 Results

4.1 Objectives

The team was very pleased with the final result of the prototype. Despite the changes, the following criteria established at the beginning of the project were fulfilled:

- The robot is of modular design – both the arms and the head are easy to detach from the torso and can be exchanged for any other desired component. The neck is designed to not only support the head, but is also able support a Microsoft Kinect.
- The robot should serve as a platform for students to solve problems and learn how to manipulate the robot to perform a certain task – the robot is equipped with an interface in a preexisting programming language – C++, but it is prepared to be integrated with other languages. The entire torso has the ability to rotate 300 degrees, the head can tilt up and down and the arms have 4 degrees of freedom. These factors enable the students to control the robot to perform a wide range of tasks.
- There should be the possibility for further development of the robot – the base of the robot consisting of a tabletop support is removable and the construction can be continued to be equipped with legs. As stated above, the robot is prepared to be developed to support several programming languages and be equipped with different toolsets. The bus chosen for CHABOT is expandable and makes it possible to add new electronics without modifying existing components.
- The robot should be able to be used in exhibitions as a promotional device for Chalmers University of Technology – the shell of the robot is designed and constructed to be aesthetically pleasing whilst it is also easy to remove and exchange for a different cover.

Due to a range of different circumstances, some limitations and changes were made so that the actual final product could be the best result as possible. The main changes were made in the construction of the skeleton

and the design of the shell. The final results of the 3D models can be found in Fig. 4.1

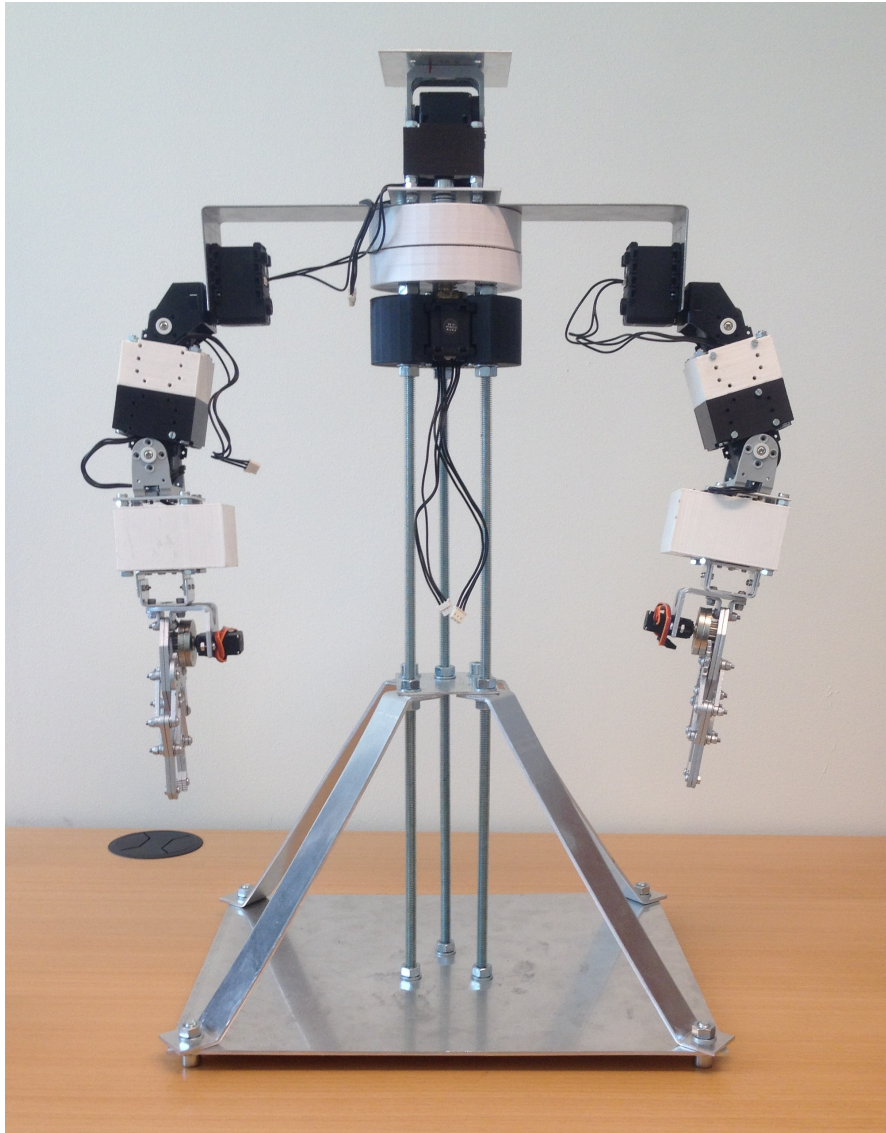


Figure 4.2: *CHABOT skeleton*

4.2 Cost Analysis

The cost analysis of both the constructed prototype and the estimated costs of the seven reproductions are found in the Appendix C. The CHABOT prototype was successfully constructed within the allotted budget of 10 000 SEK. Actually it came in quite a bit under budget. The final cost surmounted to around 7 000 SEK. Unfortunately the budget of recreating seven robots with the same components as the prototype consists of was impossible. The final cost per reproduced unit resulted in around 6200 SEK, thus a surmount of 1200 SEK. The estimated cost of the electrical components amounted to 4875 SEK the remaining 1325 SEK was an estimate of the costs for the metal and plastic.

The most expensive component, that takes up the greater deal of the budget are the AX12-A Dynamixel servomotors. Different vendors were weighed against each other to find the most cost effective deal to purchase seventy-seven servomotors. It turned out that the cheapest alternative was not the one with the lowest unit price per servo motor, but the one that gave a better discount for universities, had free shipping and the

additional connecting shoulder brackets came at a fraction of the cost compared to other vendors. This brought the final costs down substantially, but unfortunately not enough. To lower the final price even more the quality of the web camera had to be downgraded to a cheaper model.

When creating the prototype the team was able to build the skeleton from metal parts and plastic rapid prototyping parts offered by the prototype lab at Chalmers as a part of the organization in the bachelor thesis course. In the beginning of the project, the customers showed an interest in investing in a 3D printer for their department. Thereby the cost of reproducing the rapid prototyping parts were not included in the cost analysis.

To recreate the skeleton and shell most parts need to be created by hand in a well equipped workshop. The customer has the possibility to either construct all the parts themselves, or they could outsource the construction to an outside party for an additional cost. The cost analysis only reflects the cost of the customer recreating all the reproductions themselves.

The molds for the construction of the shells do not need to be recreated as they can be reused for all the reproductions. Thereby there is no additional cost for wood that the molds are made of.

4.2.1 Component Evaluation Method

The components that were limiting when it came to sizing, were chosen through the component-to-component evaluation method. These components were the servomotors and the mechanical claw/gripper. The matrices can be found in the Appendix D and E. The servomotor matrix resulted in two different servomotors having the same final score. The tie was broken by the customer insisting on the Dynamixel servomotor. This was not an issue when it came to the claws, the Dagu mrk 2 was the clear winner.

5 Discussion

As the project was of high complexity, there were several road bumps that required thought and changes in the initial planning. CHABOT could not be completed by the time the report was to be presented and the work will be continued during the coming two weeks until the presentation. Nonetheless, the majority of the goals were fulfilled during that time. The team has reflected over all parts of the project and the outcomes of these observations are presented in the section below.

5.1 Project Management & The Value Model

The teachings of the Value Model served as a perfect outline of how to successfully fulfill the primary objectives of the project. The defining and planning stage gave a clear visual picture of the task to be taken on by the team and established roles with designated responsibilities. As in many projects, the first plan is never the last. The time schedule with the deliverables and due dates was revised a couple of times during the duration of the project. It is difficult to say how this could have been avoided without establishing the fact that the group should have created a more realistic plan from the beginning and setting the appropriate limitations. Apart from that, a possible improvement could be to order the components in an earlier stage of the project. If the team had decided on all the components earlier, there would have been the possibility to have more time in the prototype lab to construct the final product.

5.2 Cost Analysis

In regards to the price point, the cost analysis played two parts; the first as a marker keeping track of the expenses throughout the duration of the project, the other as a calculation of the theoretical reproduction costs. The most challenging of the two was calculating the reproduction costs, as it was difficult to keep the prospective costs under the set budget per reproduction. A great deal of work was put into finding the cheapest alternative of all the components involved. Many different strategies were explored, buying components in bulk, petitioning for discounts given to universities and even compromising some quality aspects to find a lower price. The lowest unit price per reproduction was still 1200 SEK over budget.

There are a couple of ways that the costs could be able to come under set budget. One option could be to reduce the quality of the servomotors and opt for a cheaper version. The AX-12A Dynamixel servomotors were by far the most expensive component of the robot. These were requested by the customer and were therefore included in both the prototype and the reproduction cost analysis. Another approach would be to find a way to acquire the metal and the plastic for the skeleton and shell through different means. By only purchasing the electronic components for the seven reproductions, the final price point ends up at 4875 SEK. Both the final prototype expense report and the reproduction cost analysis can be found in the Appendix B and Appendix C.

5.2.1 Component Evaluation Method

The component evaluation method was useful in steering the team in the right direction with providing concrete courses of actions to be followed. Deciding on which machineries to purchase was not a simple assignment attributed to the extensive volume of suitable components.

The component-to-component matrix served very useful in deciding on the servomotors and the mechanical claws, more than the rest of the components involved. These two components entailed more relevant qualities that were to be considered rather than just the price point. With the help of the matrix the team was able to decide on the appropriate choice. The matrices can be found in the Appendix of this report.

Even though the evaluation methods were useful in the beginning of the project, they were of no relevant use in the later part of the project. The decisions that were made as an outcome of this method could have been taken directly simply through discussion. It could even be argued that the time spent on creating the matrices

and recreating them for the purpose of including them in the report, could be spent on more pertinent issues such as working on the construction.

5.3 Construction/Design Methods

The majority of the methods implemented in the project were very successful and the virtual tools were recognized as vital to that success. The ability to pre-assemble and check the construction for mistakes before building the actual prototype was used in all aspects of the project. As stated in the previous chapter, the final result was, with the exception of the necessary extra limitations, to the team's satisfaction.

Different parts of the CHABOT project could be separated into independent sub-projects, mainly into two different paths with only minimal dependencies on each other. One part containing the electronic and the software related tasks and the other containing the mechanical skeleton and the shell. This separation allowed each branch to work in parallel. As a result of this, more features could be developed over a shorter period of time. This strategy allowed the team to utilize the time of all the participants more efficiently.

5.4 Solid mechanics

The equations suggest that the current CHABOT is not very strong when moving and is not able to carry a great load. However the lifting capacity of the CHABOT has yet to be tested in experiments. On the other hand it is remarkably proficient in holding weights without moving the strained servomotor, which opens up the capacity to move heavy loads by gripping items and turning the torso instead of the arms. Possible solutions of the limited lifting power that is deemed to be too low in practical applications, is to replace the AX-12A Dynamixel servomotor with the stronger AX-18A Dynamixel servomotor in key joints. Another option would be to replace the aluminum claws with a lighter gripper and thus reducing the weight of the arm.

The neck is able to carry a greater amount of weight due to the shorter distance of the load from the servomotor. The forces affecting the waist and in extension the steel axle, was lower than initially feared. As the experiments with CHABOT will continue, time will tell if the strain causes long term wear on the waist.

5.5 Skeleton

The design of the skeleton and the arms of CHABOT began before any set weight limit or dimensions were in place. As a consequence, much had to be redone when the team decided on the servomotors. The end result was influenced by those earlier design decisions such as the arm length and the overall size of the CHABOT. As the plans for the shell were created simultaneously, drastic changes of either the form had to be worked around when problems arose to avoid an entire rebuild of the mechanical structure. Calculations of the soundness of the design was impossible to keep up to date, and in the end, the one constant to adhere to was the specifications of the AX-12A Dynamixel servomotor. This led to the utilizing and reliance of the rapid prototyping technique as it was both quicker and easier to completely design the parts virtually and send them for review to then adjust them accordingly rather than making parts out of aluminum that were obsolete in three days time.

As stated earlier, the waist created a lot of trouble when it came to the design. As a system to relieve the servomotors from forces perpendicular to the rotational axis often involves using cogs and different gears to turn the axis indirectly. The suggested solution was proposed and implemented at a time the servomotor was thought to only be able to rotate 300 degrees. This was a mistake due to a misreading of the data sheet causing the entire weight of the moving parts of the CHABOT to rest on that single axis and the cylinder of rapid prototyping plastic. Extensive testing of the waist system is therefore proposed to determine if the plastic is sufficient to handle both the prolonged forces of friction and the forces created when CHABOT does heavy lifting.

5.6 Shell

The shell of CHABOT evolved over the course of the project from a rigid suit of armor covering most of the robot into the final design of a head, breast plate and skirt combination. The arm-coverings were deemed unnecessary as it apart from limiting the movement of the arms would also add unnecessary weight thus reducing the amount of work CHABOT would be able to perform.

Due to the construction methods and mechanical limitations, the priority of the aesthetical factor in the primary proposed design was downgraded. This was unfortunate as one of the main goals of the project was to create an exiting new robot that would work as a promotional device for Chalmers at exhibitions and shows. On the other hand, the request of the customer was to create a robot that was an improvement of the existing lab robot in the course TIF160. By those standards the team did exceedingly well, as just the construction of the skeleton can be seen as an improvement of the previous version.

5.7 Electronics

In the early stages of the project, it was decided to implement the controller board as a single board. Due to the elevated price of creating this board, it was not able to be realized as the cost of reproducing these boards were to high. Thereby a decision to build around a development board was taken. This proved to be the most cost effective way of including a CAN bus, multiple USARTs and PWM devices. The selected card had two CAN controllers, eight different PWM channels and eight USART packed in reasonable package. The Modbus was primarily intended to handle the communication, but this was later changed since it was determined that CAN provided more features, including a possibility to send messages from a slave with out the master actively requesting the information.

6 Conclusion

The team as a whole is very satisfied with the CHABOT as all tasks set by the objectives are fully realised in the prototype with the notable exception being the LED display matrix. The CHABOT prototype is on all accounts believed to be an improvement of the previous robot utilized in the course TIF160 and the team looks forward to hearing actual usage feedback and data from the customer. Many different hurdles were overcome to reach this point, and a valuable lesson is to always be open to changes in the design, as it is very hard to predict how a project will develop when on the planning stage. By keeping the dialog open the team was able to focus on the objectives of the project rather than getting stuck in a single mindset or certain ways of reaching said objectives. The final incarnation of the CHABOT is not the same machine that was envisioned at the beginning of the project, but it is a marvelous machine that certainly embodies the traits it was built to emphasize; modularity, versatility and style.

6.1 Further work

The CHABOT team arrived at the following improvements and additions recommended to be carried out in the future.

A desired mechanical upgrade of the CHABOT that would add even more versatility, is the addition of a means of transportation. To keep to the humanoid theme the specific means proposed for CHABOT is a pair of working legs or leg like appendages. The addition of a battery power option and an internal computer like a Beagle Bone black or a Raspberry PI to allow wireless operation is also then suggested. This would eliminate the requirement of being attached to a computer station.

An improvement that could be made if the robot was to remain immobile, would be to upgrade the means of power supply. The current power supply is a fairly large switched regulator that can deliver 3A 5v. This power supply could be made smaller by using a regulator with a higher switching frequency.

The re-inclusion of a visual mouth would be beneficial. As at the present, there are no means of live communication, except for movements and gestures. The original idea was to use a LED display matrix with a resolution of 16x8 pixels. The initial version of the firmware was developed and included a working display, it would thereby not be such a large task to undertake this improvement.

The present version of the controller board has features like Analog input, Digital IO, PWM, CAN and Servo motor. Unfortunately, there are no connectors for analog or digital IO, which makes usage of analog in and digital IO more difficult to use.

There are several improvements to be made when it comes to the communication. CHABOT's communication protocol was originally intended to have a delivery assurance. Such features would be desirable in future versions of the firmware. The current version of CHABOT's communication library only includes support for C++. A relevant improvement would be to add support for languages such as Python and Ruby. Another addition could be to add support for Java and Matlab. Both Java and Matlab are used by many of Chalmers introduction courses, which would allow the opportunity to expand the usage of CHABOT in different course plans. By wrapping the C++ class into a container accessible from these programming languages could fulfill most of these goals. CHABOT's communication library only supports Unix like operating systems since the communication is handled with help of "termios". This means that there is support for BSD derivatives, Linux and Mac OS X but no support for Windows, which could also be a relevant expansion.

Bibliography

- [1] P. Lindstedt and J. Burenius, *The Value Model: How to Master Product Development and Create Unrivalled Customer Value*. Nimba, 2003, ISBN: 9789163063497. [Online]. Available: <http://books.google.se/books?id=Gq4JS3Pxpp0C>.
- [2] D. Hanson, “Exploring the aesthetic range for humanoid robots”, *Proceedings of Cognitive Science (CogSci 2006) Workshop on Android Science*, 2006.
- [3] M. Mori, The uncanny valley, *Energy* no. 7(4) 1970, 33–35, 1970, Translated by Karl F. MacDorman and Takashi Minato.
- [4] *St spi protocol*.
- [5] K. W. R. James F. Kurose, *Computer Networking: A Top-Down Approach, 6th ed.* Pearson, 2012.
- [6] C. K. T. K. Manny Sltero Jing Zhangm Chris Cockril, “Rs-422 and rs-485 standards overview and system configurations”, Texas Instruments, Tech. Rep., 2010.
- [7] *Modbus application protocol specification v1.1b3*, Modbus Organization.
- [8] D. C. Watterson, “Controller area network (can) implementation guide”, Analog devices, Tech. Rep., 2012.
- [9] 2014. [Online]. Available: http://support.robotis.com/en/product/dxl_main.htm.
- [10] P.-r. J. Ragnar Grahn, *mekanik, statik och dynamik*. Studentlitteratur, 2010.

A Component list

Name	#	Manufacturing	Material	Code
Torso				
Threaded rod 8mm	3	Purchase, cutting	Galvenized Steel	SK1-T
Bottom Platform	1	Cutting, drilling	6mm Aluminium	SK2-T
Square Support	1	Cutting, drilling	2mm Aluminium	SK3-T
Angle Support	4	Cutting, drilling, bending	2mm Aluminium	SK4-T
Torso Shoulder Beam	1	Cutting, drilling, bending	3mm Steel	SK5-T
Torso Shell Front	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH1-T
Torso Shell Back	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH2-T
Support Shell Front	1	Cutting, bendin	2mm Aluminium	SH3-T
Support Shell Back	1	Cutting, bendin	2mm Aluminium	SH4-T
Accordian Cylinder 150mm D	1	Purchase	Plastic	ACC-150
Servomotor	3	Purchase	Electronic component	DYN-SERV
Servomotor Bracket 45 Degree	2	Purchase	Aluminium	DYN-45
Rotation Top Part	1	3D Rapid Prototyping	Plastic	ROT1-T
Rotation Bottom Part	1	3D Rapid Prototyping	Plastic	ROT2-T
Rotation Shaft	1	Latheing	Brass	ROT3-T
Servomotor Support	1	3D Rapid Prototyping	Plastic	ROT4-T
AX12 Power Transmission Coupling	1	Cutting, drilling, turning, welding	Aluminium & Copper	ROT5-T
Nut 8mm	20	Purchase	Steel	M8
Screw 8mm	8	Purchase	Steel	S8
Can Bus				
IEEE1394	4	Purchase	Electronic component	CB1
Header, 6-Pin	2	Purchase	Electronic component	CB2
Controller				
Capacitor (Semiconductor SIM Model)	6	Purchase	Electronic component	CON1
Typical BLUE SiC LED	2	Purchase	Electronic component	CON2
IEEE1394	1	Purchase	Electronic component	CON3
USB, MINI, SMD, RA KME04-USBMU03A01	1	Purchase	Electronic component	CON4

Header, 10-Pin, Dual row	2	Purchase	Electronic component	CON5
Header, 3-Pin (Header 3)	1	Purchase	Electronic component	CON6
Header, 3-Pin (22-03-5035)	1	Purchase	Electronic component	CON7
Header, 2-Pin	1	Purchase	Electronic component	CON8
Header, 4-Pin	1	Purchase	Electronic component	CON9
Resistor	10	Purchase	Electronic component	CON10
3.3V CAN Transceiver SN65HVD230DR	1	Purchase	Electronic component	CON11
USB Basic UART Interface Chip,FT230XS-R	1	Purchase	Electronic component	CON12
74126	1	Purchase	Electronic component	CON13
Power Board				
Polarized Capacitor (Radial)	1	Purchase	Electronic component	PB1
Polarized Capacitor (Radial)	1	Purchase	Electronic component	PB2
Zener Diode	1	Purchase	Electronic component	PB3
Inductor	1	Purchase	Electronic component	PB4
Plug	4	Purchase	Electronic component	PB5
Header, 6-Pin	1	Purchase	Electronic component	PB6
Resistor	1	Purchase	Electronic component	PB7
SIMPLE SWITCHER® 3A (LM2576T-5.0)	1	Purchase	Electronic component	PB8
Arm				
Threaded rod 4mm	8	Purchase, cutting	Galvenized Steel	SK1-A
Arm Plate	2	Cutting, drilling	2mm Aluminium	SK2-A
Servomotor	4	Purchase	Electronic component	DYN-SERV
Servomotor Bracket	3	Purchase	Plastic	DYN-BR
Servomotor Horizontal Support	2	3D Rapid Prototyping	Plastic	SER-HOR
Servomotor Vertical Support	2	3D Rapid Prototyping	Plastic	SER-VER
Nut 4mm		Purchase	Steel	M4
Claw Adapter Part	2	Cutting, drilling, bending	2 mm Aluminium	CLAW-ADPT
Claw with Servomotor	1	Purchase	Electronic component	CLAW

Accordian Cylinder 80mm D	1	Purchase	Plastic	ACC-80
Head				
Servomotor Vertical Support	1	3D Rapid Prototyping	Plastic	SER-VER
Servomotor	1	Purchase	Electronic Component	DYN-SERV
Neck Base Plate	1	Cutting, drilling	2mm Aluminium	SK1-H
Head Shell Front	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH1-H
Head Shell Back	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH2-H
Accordian Cylinder 100mm D	1	Purchase	Plastic	ACC-100
Head Base Plate	1	Cutting, drilling	2mm Aluminium	SK2-H
Threaded rod 4mm	4	Purchase, cutting	Galvenized Steel	SK3-H

B Budget

Please note that the following figures are presented in the Swedish format, as such the commas are equivalent to periods in the English format. All prices are in SEK (Swedish kronas)

Cost Analysis of Prototype

Order	Supplier	Article #	Name	#	Unit Price	Price
1	FARNELL	2314937	TM4C123G, LAUNCHPAD, EVAL KIT	3	155,63	466,89
		2290395	DISPLAY, 0.8	1	69,3	69,3
		84521148	IC, CAN TRANSCEIVER, 1MPS, 8SOIC	3	6,59	19,77
		Price	555,96			
		Shipping	48,47			
		Total Price	604,43			
2	Crustcrawler	AX-12	Dynamixel AX12A (wires&screws)	12	287,01	3444,12
		SSB-45	AX45 bracket	2	89,17	178,34
		SSB-SHORT	AXShort bracket	2	77,67	155,34
		Price	3777,8			
		Shipping	351,58			
		Total Price	4129,38			
3	FARNELL	1739584	CD74HCT126	3	19,77	59,31
		9979620	MOLEX	3	1,21	3,63
		1200133	IEEE1394	6	15,35	92,1
		2081321	FTDI FT230XS	3	22,22	66,66
		1355761	MOLEX USB MINI A/B	3	14,81	44,43
		2100134	FIREWIRE 106,80	3	35,6	106,8
		1564682	LM2576T-5.0	1	25,48	25,48
		2322473	1N5822	1	5,4	5,4
		1308471	CHOKe, 100U PANASONIC	1	12,84	12,84
		1219478	100UF 50V PANASONIC	1	1,62	1,62
		1848278	100UF 10V PANASONIC	1	2,3	2,3
		Price	420,57			
		Shipping	48,47			
		Total Price	469,04			
4	RobotSavvy		DAGU ALUMINIUM CLAW & SERVO	2	297,99	595,98
		Price	595,98			
		Shipping	264,97			
		Total Price	860,95			
5	FARNELL	2334609	IDEAL POWER -25HK-AB-120A250	1	156,55	156,55
		224972	PLUG DC13A	1	63,57	63,57
		9733477	STIFTLIST	2	10,35	20,7
		2084276	STIFTHYLSA	2	9,7	19,4
		Price	260,22			
		Shipping	48,47			
		Total Price	308,69			

Cost Analysis of Prototype

6	Dustin	5010546650	X-MINI II CAPSULE WHITE	1	189	189						
		5010463018	LOGITECH WEBCAM C310	1	299	299						
		5010049741	DLINK DUB-H4 USB HUB 4-PORT USB	1	135	135						
		Price	623									
		Shipping										
		Total Price	623									
<table><tr><td>Total Costs</td><td>6995,49</td></tr><tr><td>Budget</td><td>10000</td></tr><tr><td>Difference</td><td>3004,51</td></tr></table>							Total Costs	6995,49	Budget	10000	Difference	3004,51
Total Costs	6995,49											
Budget	10000											
Difference	3004,51											

C **Reproduction costs**

Please note that the following figures are presented in the Swedish format, as such the commas are equivalent to periods in the English format. All prices are in SEK (Swedish kronas)

Supplier	Article #	Name	#	Unit Price	Price
FARNELL	2314937	TM4C123G, LAUNCHPAD, EVAL K	21	135,99 kr	2 855,79 kr
	8452148	IC, CAN TRANSCEIVER, 1MPS, 8SC	21	16,15 kr	339,15 kr
	1739584	CD74HCT126	21	4,40 kr	92,40 kr
	9979620	MOLEX	21	1,21 kr	25,41 kr
	1200133	IEEE1394	42	12,82 kr	538,44 kr
	2081321	FTDI FT230XS	7	22,22 kr	155,54 kr
	1355761	MOLEX USB MINI A/B	7	14,81 kr	103,67 kr
	2100134	FIREWIRE 106,80	21	35,60 kr	747,60 kr
	1564682	LM2576T-5.0	7	25,48 kr	178,36 kr
	2322473	1N5822	7	5,40 kr	37,80 kr
	1308471	CHOKE, 100U PANASONIC	7	12,84 kr	89,88 kr
	1219478	100UF 50V PANASONIC	7	1,62 kr	11,34 kr
	1848278	100UF 10V PANASONIC	7	2,30 kr	16,10 kr
	2334609	IDEAL POWER -25HK-AB-120A250	7	156,55 kr	1 095,85 kr
	224972	PLUG DC13A	7	63,57 kr	444,99 kr
	9733477	STIFTLIST	14	9,05 kr	126,70 kr
	2084276	STIFTHYLSA	14	6,40 kr	89,60 kr
	Price	6948,62			
	Shipping	100			
	Total Price	7048,62			
TrossenRobotics	AX-12	Dynamixel AX12A (wires&screws	77	294,31 kr	22 661,87 kr
	SSB-45	AX45 bracket	14	9,77 kr	136,78 kr
	Price	22798,65			
	Discount	1139,88			
	Shipping	0			
	Total Price	21658,77			
	5010627588	Deltaco webbkamera c-198	7	99,00 kr	693,00 kr
	5010049741	DLINK DUB-H4 USB HUB 4-PORT	7	135,00 kr	945,00 kr
	Price	1638			
	Shipping	100			
	Total Price	1738			
RoboSavvy		DAGU ALUMINIUM CLAW & SERV	14	297,99 kr	4 171,86 kr
	Price	4171,86			
	Shipping	264,97			
	Total Price	4436,83			
GBS		AL PLÅT 2000 X 1000 X 2,00	1	385,00 kr	385,00 kr
		AL PLÅT 3000 X 1500 X 5,00	2	2 345,00 kr	4 690,00 kr
	Price	4690			
	Shipping	0			
	Total Price	4690			

Verktysboden.se	63011	Gängstång 8mm x 2000	5	24,00 kr	120,00 kr
	55633	Gängstång 4mm x 1000	3	9,00 kr	27,00 kr
	321796	Bult M8 100st	1	83,00 kr	83,00 kr
	58968	Mutter M6 100st	2	60,00 kr	120,00 kr
	Price	350			
	Shipping	250			
	Total Price	600			
		2mm Polycarbonate plastic	7	500,00 kr	3 500,00 kr
	Price	3500			
	Shipping				
	Total Price	3500			
	Total Costs	43 672,22 kr			
	Budget	35 000,00 kr			
	Difference	- 8 672,22 kr			
	Unit Cost	6 238,89 kr			

D Servomotor Component-to-Component Matrix

	Dynamixel AX12 HS755HB	HS805BB	T0151
Price	293,00 kr	238,00 kr	296,00 kr
Max load (Kg/cm)	16,5	13,2	24,7
Speed (s/60)	0,19	0,23	0,14
Weight	55g	110g	152g
Rotation	360	360	360
Dimensions	50x32x38	59x29x59	66x30x58
			42x20,5x39,5

Dynamixel AX12 compared to the others

	Dynamixel AX12 HS755HB	HS805BB	T0151
Price	0	+	-
Max load (Kg/cm)	0	-	+
Speed (s/60)	0	-	-
Weight	0	-	-
Rotation	0	0	0
Dimensions	0	-	-
Total	0	-3	-3
			1

HS755HB compared to the others

	Dynamixel AX12 HS755HB	HS805BB	T0151
Price	-	0	-
Max load (Kg/cm)	+	0	+
Speed (s/60)	+	0	-
Weight	+	0	-
Rotation	0	0	0
Dimensions	+	0	-
Total	3	0	-3
			1

HS805BB compared to the others

	Dynamixel AX12 HS755HB	HS805BB	T0151
Price	+	+	0
Max load (Kg/cm)	-	-	0
			-

Speed (s/60)	+	+	0	+
Weight	+	+	0	+
Rotation	0	0	0	0
Dimensions	+	+	0	+
Total	3	3	0	3

T0151 compared to the others

	Dynamixel AX12 HS755HB	HS805BB	T0151
Price	-	-	0
Max load (Kg/cm)	+	+	0
Speed (s/60)	+	+	0
Weight	-	-	0
Rotation	0	0	0
Dimensions	-	-	0
Total	-1	-1	0

Total points			
Dynamixel AX12 HS755HB	HS805BB	T0151	
5	-1	-9	5

E Claw Component-to-Component Matrix

	Robotic Claw mrk 2	Dagu aluminium gripper	Vex-Claw kit
Price	151,00 kr	440,00 kr	132,00 kr
Opening	50	50	80
Servo	Additional purchase	Yes	No
Weight	290g	220g	181g

Robotic Claw mrk 2 compared to the rest

	Robotic Claw mrk 2	Dagu aluminium gripper	Vex-Claw kit
Price	0	-	+
Opening	0	0	+
Servo	0	0	-
Weight	0	+	+
Total	0	0	2

Dagu aluminium gripper compared to the rest

	Robotic Claw mrk 2	Dagu aluminium gripper	Vex-Claw kit
Price	+	0	+
Opening	0	0	+
Servo	0	0	-
Weight	-	0	+
Total	0	0	2

Vex-Claw kit compared to the rest

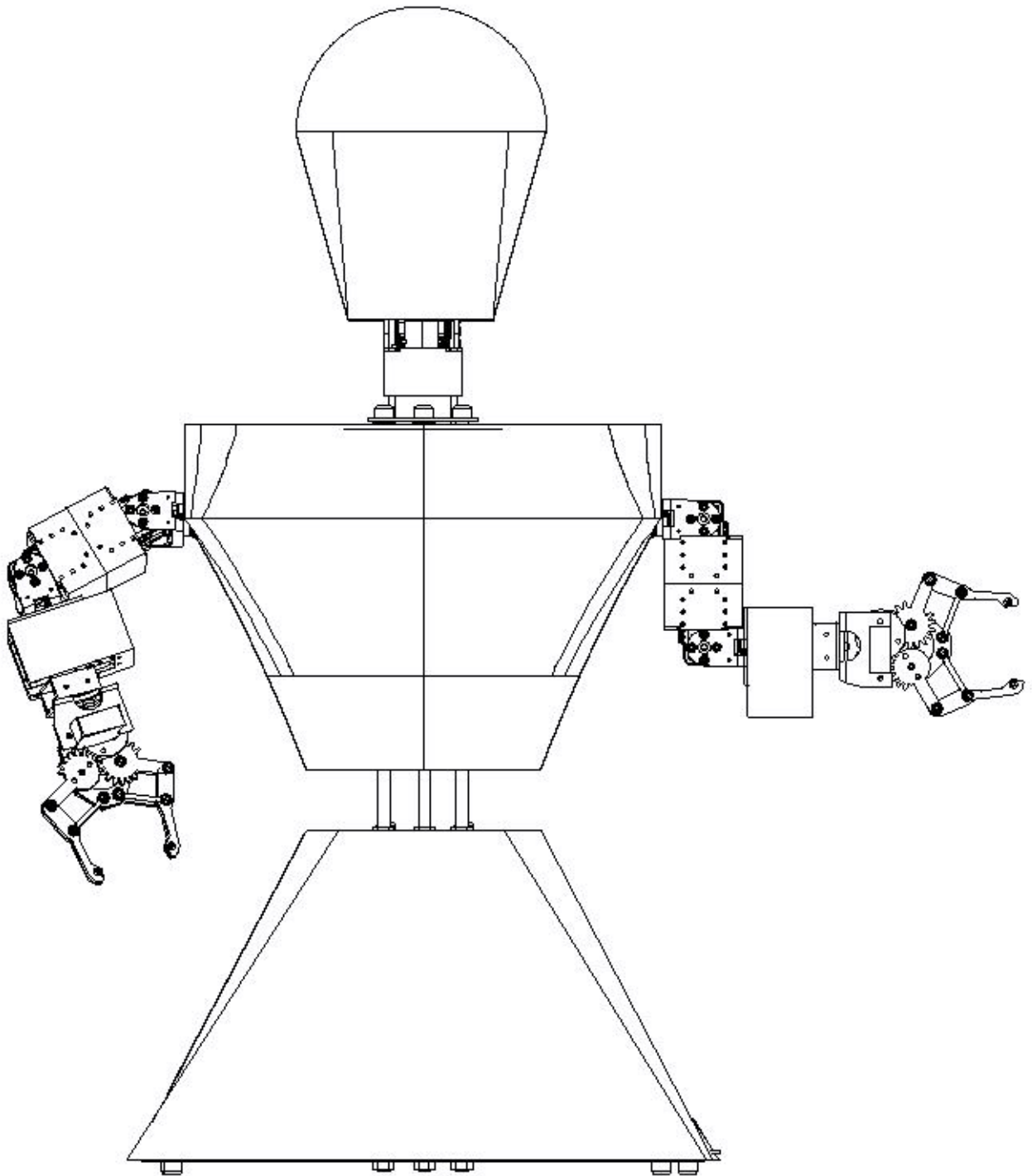
	Robotic Claw mrk 2	Dagu aluminium gripper	Vex-Claw kit
Price	-	-	0
Opening	-	-	0
Servo	+	+	0
Weight	-	-	0
Total	-2	-2	0

Total amount of points		
Robotic Claw mrk 2	Dagu aluminium gripper	Vex-Claw kit
-2	-2	4

F Construction manual

Construction manual is omitted in the printed version of the thesis. Please consult the online version.

CHABOT Robot Instruction Manual



Chalmers University of Technology
Department of Applied Mechanics
2014



CHABOT Robot Instruction Manual

Assembly Guidance

The subsequent manual is to be used in the following ways:

- The manual consists of assembly instructions of all mechanical parts, all the electronic assembly is dealt with in a separate manual.
- Each section begins with a schematic image of the assembly routine and then followed by technical drawings of the components that are to be produced by metalworking.
- The component list contains all the parts included in the robot, from the electronic parts to the bolts to the 3D printed parts. The list defines what material the part is made of, what production methods are used to produce the part, the pertinent product code and the amount to produced. With this information it should be simple to produce all the parts.
- The AX-12A Dynamixel servomotors are mounted with the accompanied brackets, screws and wires. Please review the AX-12A Dynamixel manual at http://support.robotis.com/en/product/dxl_main.htm .

The last part of manual depicts the assembly of the outer shell. This part is an initial rendering of how the shell is to be assembled. At the time the report was to be completed, the shell had not been assembled due to complications with the polycarbonate plastic delivery.

Karin Dankis
Alexander Davidsson
Eric Hardselius

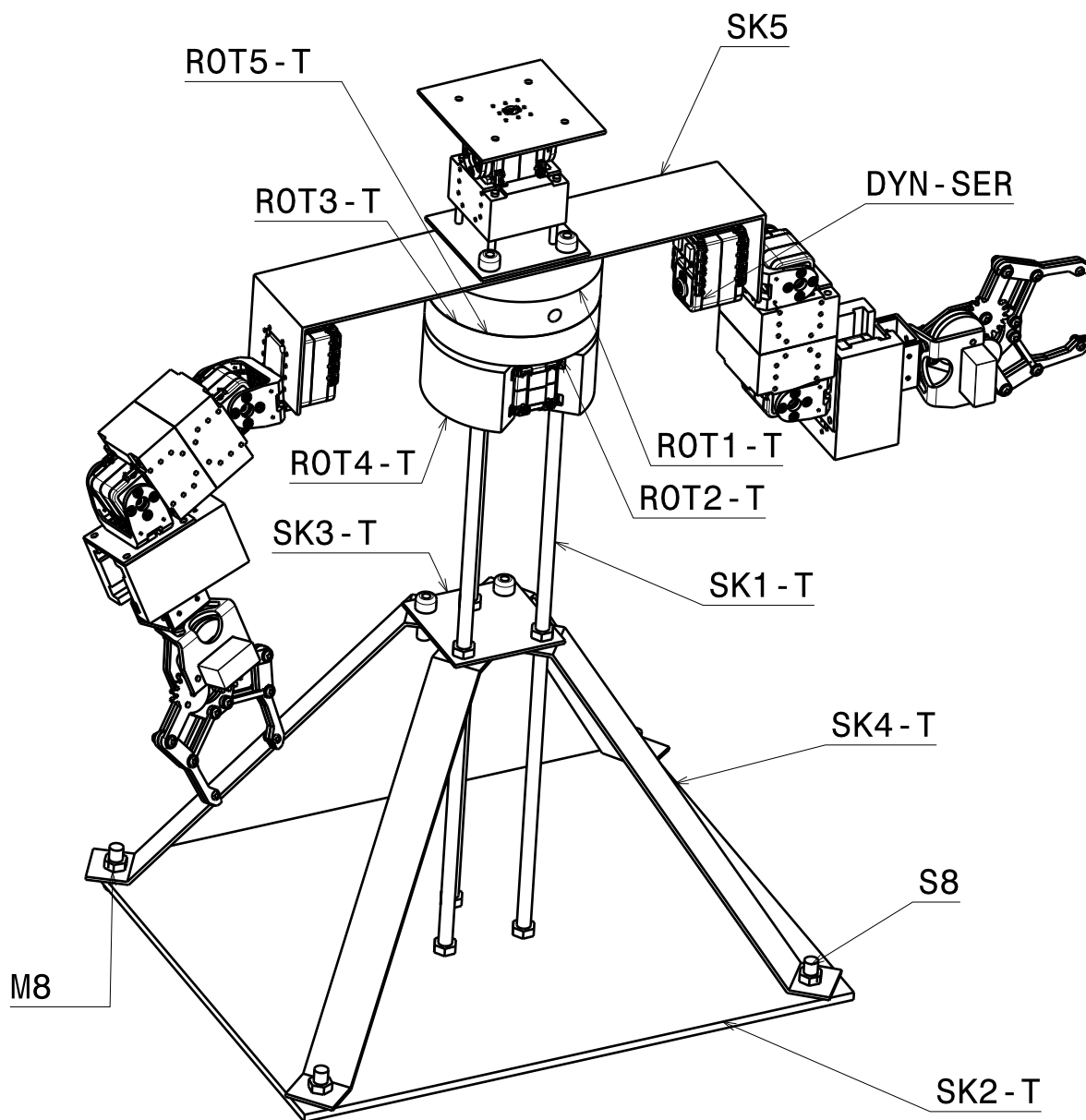
CHABOT Robot Instruction Manual

List of Components

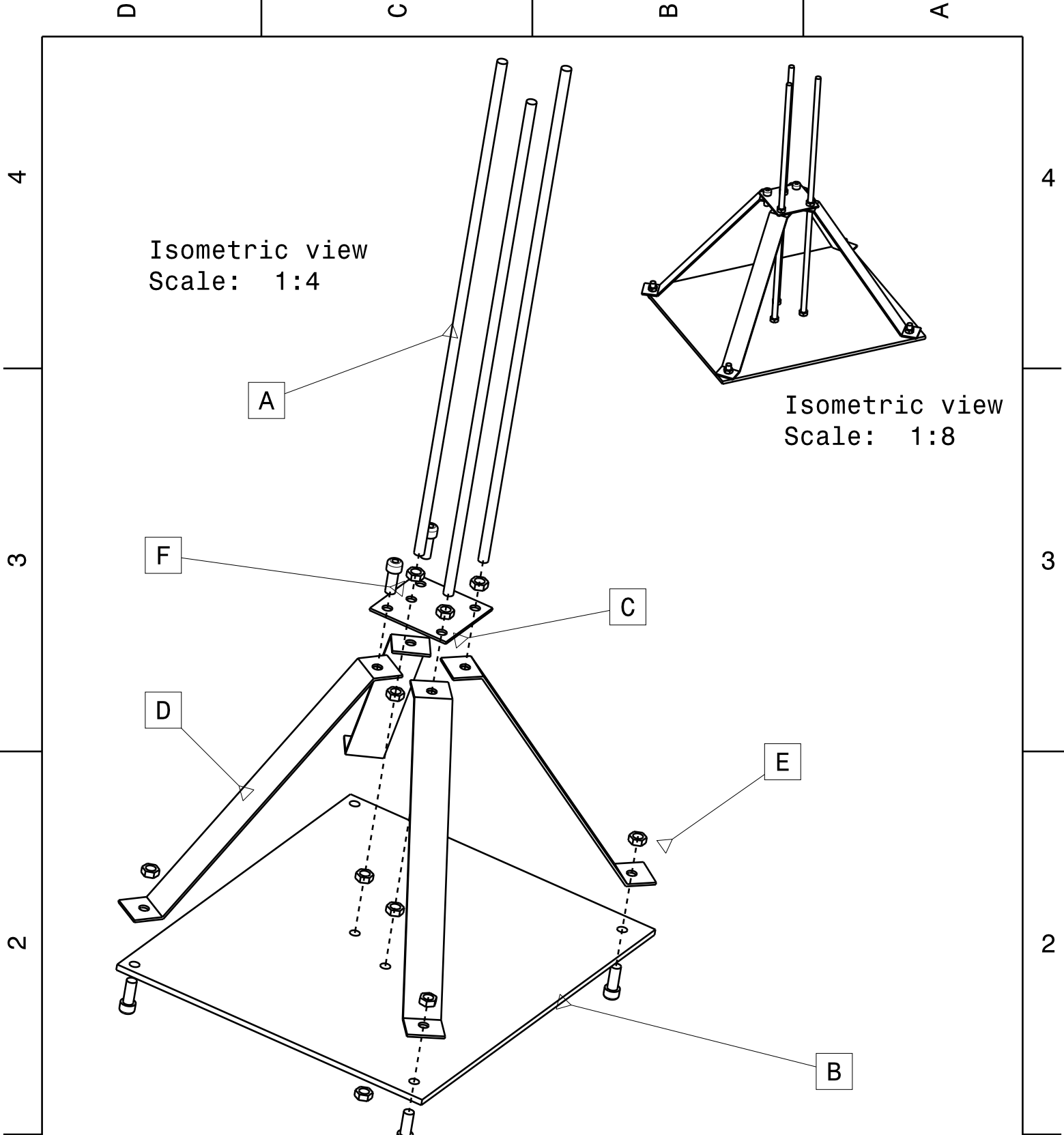
Name	#	Manufacturing	Material	Code
Torso				
Threaded rod 8mm	3	Purchase, cutting	Galvenized Steel	SK1-T
Bottom Platform	1	Cutting, drilling	6mm Aluminium	SK2-T
Square Support	1	Cutting, drilling	2mm Aluminium	SK3-T
Angle Support	4	Cutting, drilling, bending	2mm Aluminium	SK4-T
Torso Shoulder Beam	1	Cutting, drilling, bending	3mm Steel	SK5-T
Torso Shell Front	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH1-T
Torso Shell Back	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH2-T
Support Shell Front	1	Cutting, bendin	2mm Aluminium	SH3-T
Support Shell Back	1	Cutting, bendin	2mm Aluminium	SH4-T
Accordian Cylinder 150mm D	1	Purchase	Plastic	ACC-150
Servomotor	3	Purchase	Electronic component	DYN-SERV
Servomotor Bracket 45 Degree	2	Purchase	Aluminium	DYN-45
Rotation Top Part	1	3D Rapid Prototyping	Plastic	ROT1-T
Rotation Bottom Part	1	3D Rapid Prototyping	Plastic	ROT2-T
Rotation Shaft	1	Latheing	Brass	ROT3-T
Servomotor Support	1	3D Rapid Prototyping	Plastic	ROT4-T
AX12 Power Transmission Coupling	1	Cutting, drilling, turning, welding	Aluminium & Copper	ROT5-T
Nut 8mm	20	Purchase	Steel	M8
Screw 8mm	8	Purchase	Steel	S8
Can Bus				
IEEE1394	4	Purchase	Electronic component	CB1
Header, 6-Pin	2	Purchase	Electronic component	CB2
Controller				
Capacitor (Semiconductor SIM Model)	6	Purchase	Electronic component	CON1
Typical BLUE SiC LED	2	Purchase	Electronic component	CON2
IEEE1394	1	Purchase	Electronic component	CON3
USB, MINI, SMD, RA KME04-USBMU03A01	1	Purchase	Electronic component	CON4
Header, 10-Pin, Dual row	2	Purchase	Electronic component	CON5
Header, 3-Pin (Header 3)	1	Purchase	Electronic component	CON6
Header, 3-Pin (22-03-5035)	1	Purchase	Electronic component	CON7
Header, 2-Pin	1	Purchase	Electronic component	CON8
Header, 4-Pin	1	Purchase	Electronic component	CON9
Resistor	10	Purchase	Electronic component	CON10
3.3V CAN Transceiver SN65HVD230DR	1	Purchase	Electronic component	CON11
USB Basic UART Interface Chip, FT230XS-R	1	Purchase	Electronic component	CON12
74126	1	Purchase	Electronic component	CON13
Power Board				
Polarized Capacitor (Radial)	1	Purchase	Electronic component	PB1
Polarized Capacitor (Radial)	1	Purchase	Electronic component	PB2
Zener Diode	1	Purchase	Electronic component	PB3
Inductor	1	Purchase	Electronic component	PB4
Plug	4	Purchase	Electronic component	PB5
Header, 6-Pin	1	Purchase	Electronic component	PB6
Resistor	1	Purchase	Electronic component	PB7
SIMPLE SWITCHER® 3A (LM2576T-5.0)	1	Purchase	Electronic component	PB8
Arm				
Threaded rod 4mm	8	Purchase, cutting	Galvenized Steel	SK1-A
Arm Plate	2	Cutting, drilling	2mm Aluminium	SK2-A
Servomotor	4	Purchase	Electronic component	DYN-SERV
Servomotor Bracket	3	Purchase	Plastic	DYN-BR
Servomotor Horizontal Support	2	3D Rapid Prototyping	Plastic	SER-HOR
Servomotor Vertical Support	2	3D Rapid Prototyping	Plastic	SER-VER
Nut 4mm		Purchase	Steel	M4
Claw Adapter Part	2	Cutting, drilling, bending	2 mm Aluminium	CLAW-ADPT
Claw with Servomotor	1	Purchase	Electronic component	CLAW
Accordian Cylinder 80mm D	1	Purchase	Plastic	ACC-80
Head				
Servomotor Vertical Support	1	3D Rapid Prototyping	Plastic	SER-VER
Servomotor	1	Purchase	Electronic Component	DYN-SERV
Neck Base Plate	1	Cutting, drilling	2mm Aluminium	SK1-H
Head Shell Front	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH1-H
Head Shell Back	1	Vacuum Shaping	2mm Polycarbonate Plastic	SH2-H
Accordian Cylinder 100mm D	1	Purchase	Plastic	ACC-100
Head Base Plate	1	Cutting, drilling	2mm Aluminium	SK2-H
Threaded rod 4mm	4	Purchase, cutting	Galvenized Steel	SK3-H

CHABOT Robot Instruction Manual

Torso Assembly



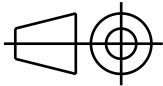
Isometric view
Scale: 1:4



DESIGNED BY:
CHABOT TEAM
DATE:
2014

SKELETON/TORSO

SIZE
A4



CHABOT

SCALE
1:1

WEIGHT (kg)
2.60

DRAWING NUMBER

SHEET
1 / 1

I	—
H	—
G	—
F	S8
E	M8
D	SK4-T
C	SK3-T
B	SK2-T
A	SK1-T

This drawing is our property; it can't be reproduced or communicated without our written agreement.

D

A

A

B

C

D

4

3

2

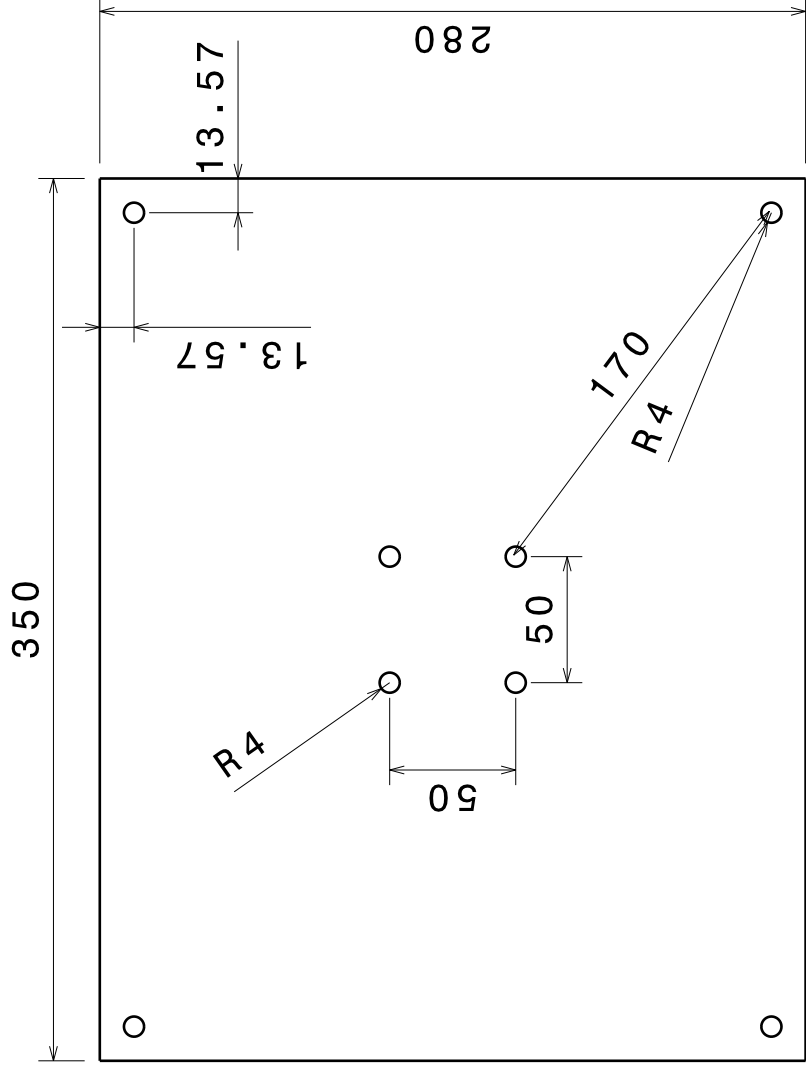
1

4

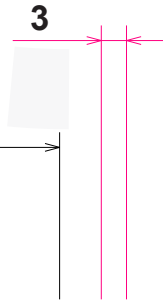
3

2

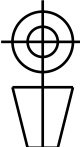
1



Side View
Scale: 1:3



DESIGNED BY: CHABOT TEAM		BOTTOM PLATFORM		I	—
DATE: 2014				H	—
				G	—
		F	—		
		E	—		
		D	—		
		C	—		
		B	—		
		A	—		

DESIGNED BY: CHABOT TEAM		BOTTOM PLATFORM		CHABOT		SHEET	
DATE: 2014						1 / 1	
SIZE A4		WEIGHT (kg)	DRAWING NUMBER	SK2-T		1 / 1	
SCALE 1 : 1						This drawing is our property; it can't be reproduced or communicated without our written agreement.	

This drawing is our property; it can't be reproduced or communicated without our written agreement.

A

D

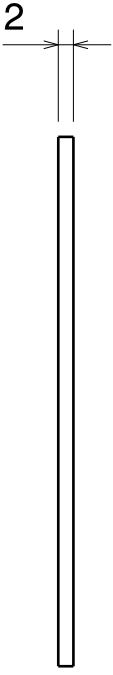
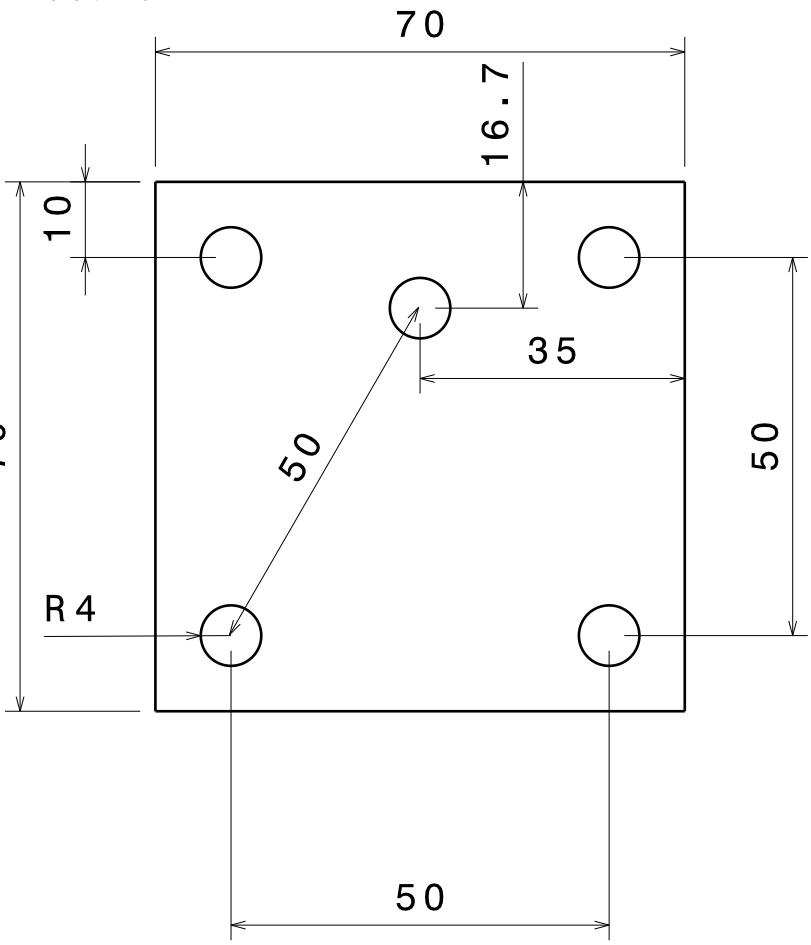
4

3

2

1

Front View
Scale: 1:1



Left view
Scale: 1:1

4

3

2

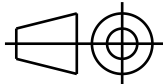
1

DESIGNED BY:
CHABOT TEAM
DATE:
2014

SQUARE SUPPORT PLATE

I	-
H	-
G	-
F	-
E	-
D	-
C	-
B	-
A	-

SIZE
A4



CHABOT

SCALE
1:1

WEIGHT (kg)
0.01

DRAWING NUMBER
SK3-T

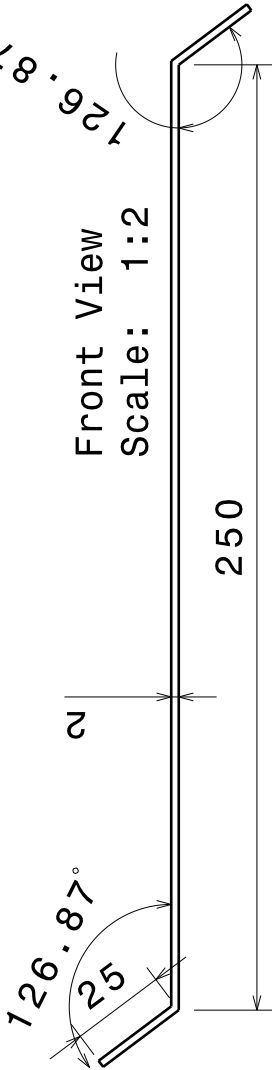
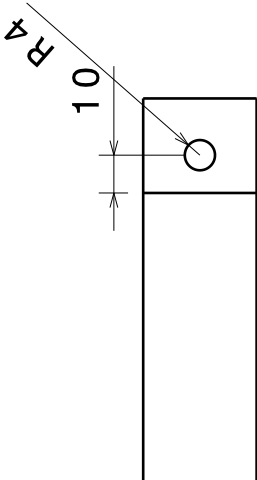
SHEET
1 / 1

This drawing is our property; it can't be reproduced or communicated without our written agreement.

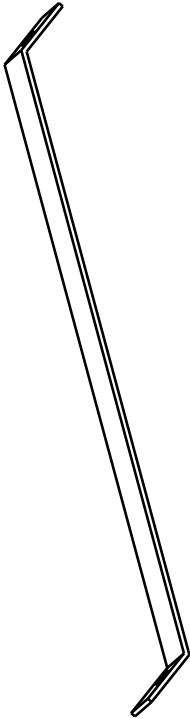
D

A

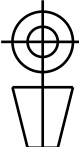
4 3 2 1 A



Bottom view
Scale: 1:2



Isometric view
Scale: 1:3

DESIGNED BY: CHABOT TEAM		Angle Support			I	-
DATE: 2014					H	-
					G	-
SIZE A4		CHABOT	F	-		
			E	-		
			D	-		
SCALE 1:1	WEIGHT (kg) XXX	DRAWING NUMBER SK4 - T	C	-		
			B	-		
			A	-		
SHEET 1 / 1			This drawing is our property; it can't be reproduced or communicated without our written agreement.			

D A

A

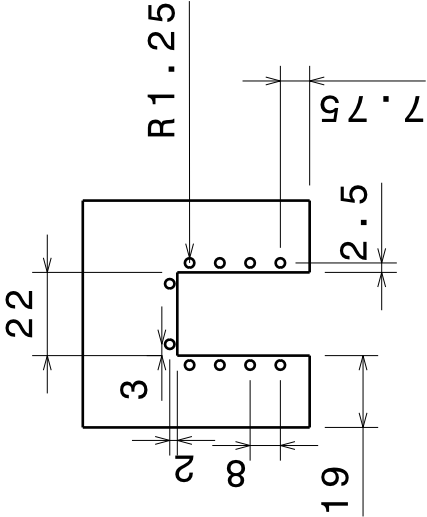
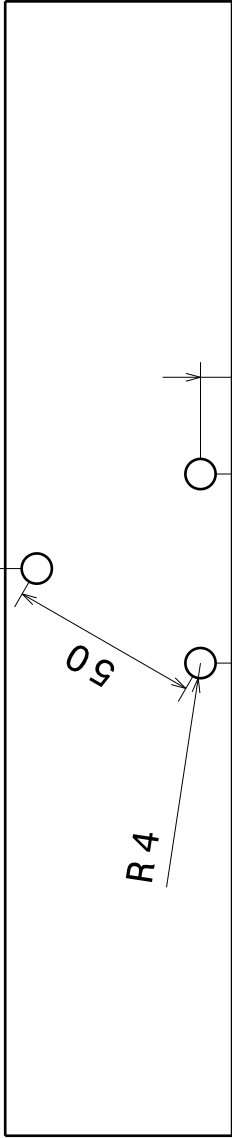
300

3

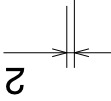
3

4

Top view
Scale: 1:2



Left view
Scale: 1:2

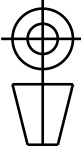


Front View
Scale: 1:2

09

2

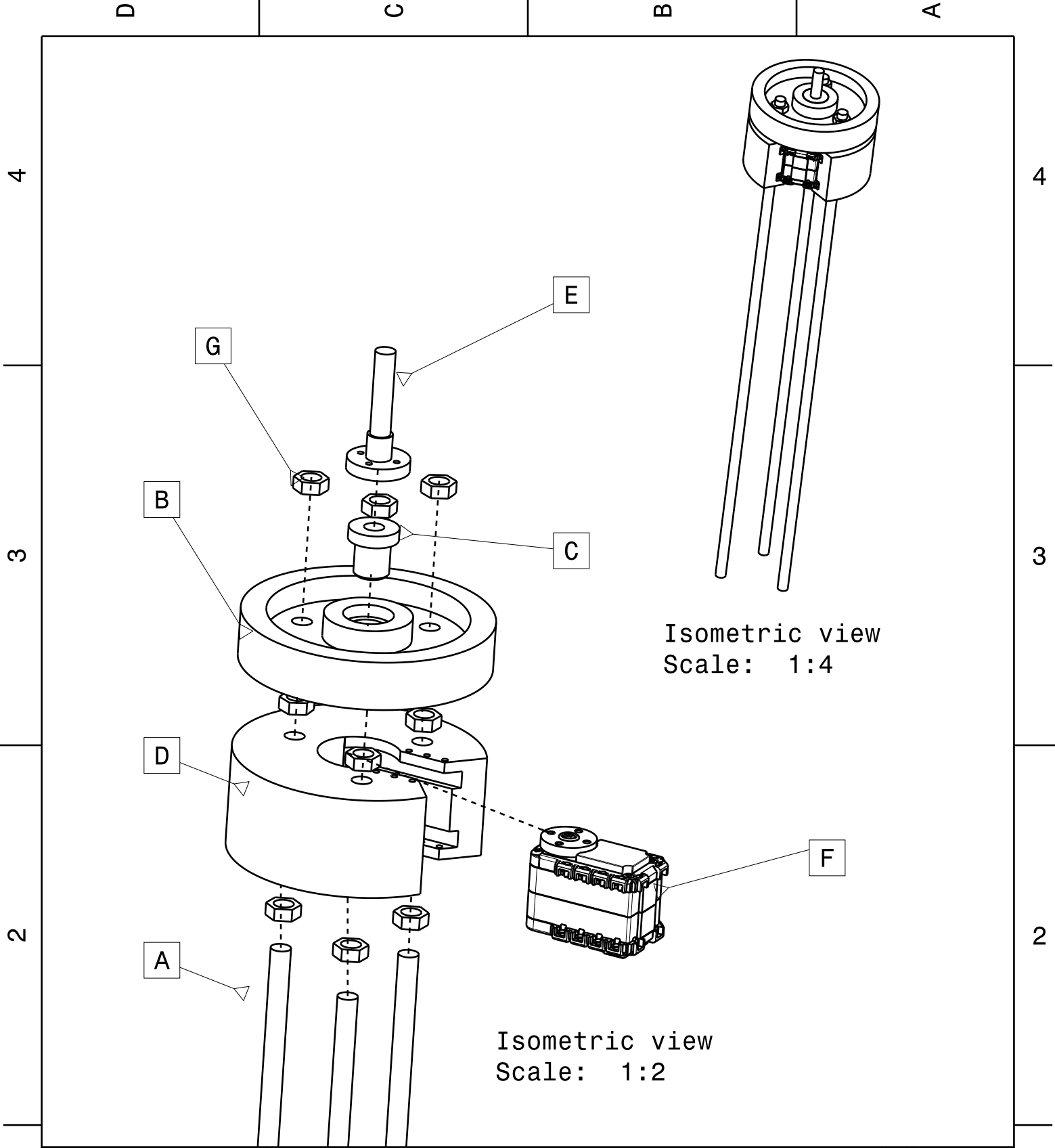
2

DESIGNED BY: CHABOT TEAM		I	-
DATE: 2014		H	-
CHECKED BY: XXX		G	-
DATE: XXX		F	-
SIZE A4		E	-
		D	-
SCALE 1:1		C	-
WEIGHT (kg) 0.13		B	-
DRAWING NUMBER SK5-T		A	-
SHEET 1/1			
CHABOT			
TORSO SHOULDER BEAM			

This drawing is our property; it can't be reproduced or communicated without our written agreement.

D

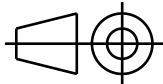
A



DESIGNED BY:
CHABOT TEAM
DATE:
2014

SKELETON/ROTATION

SIZE
A4



CHABOT

SCALE
1:1

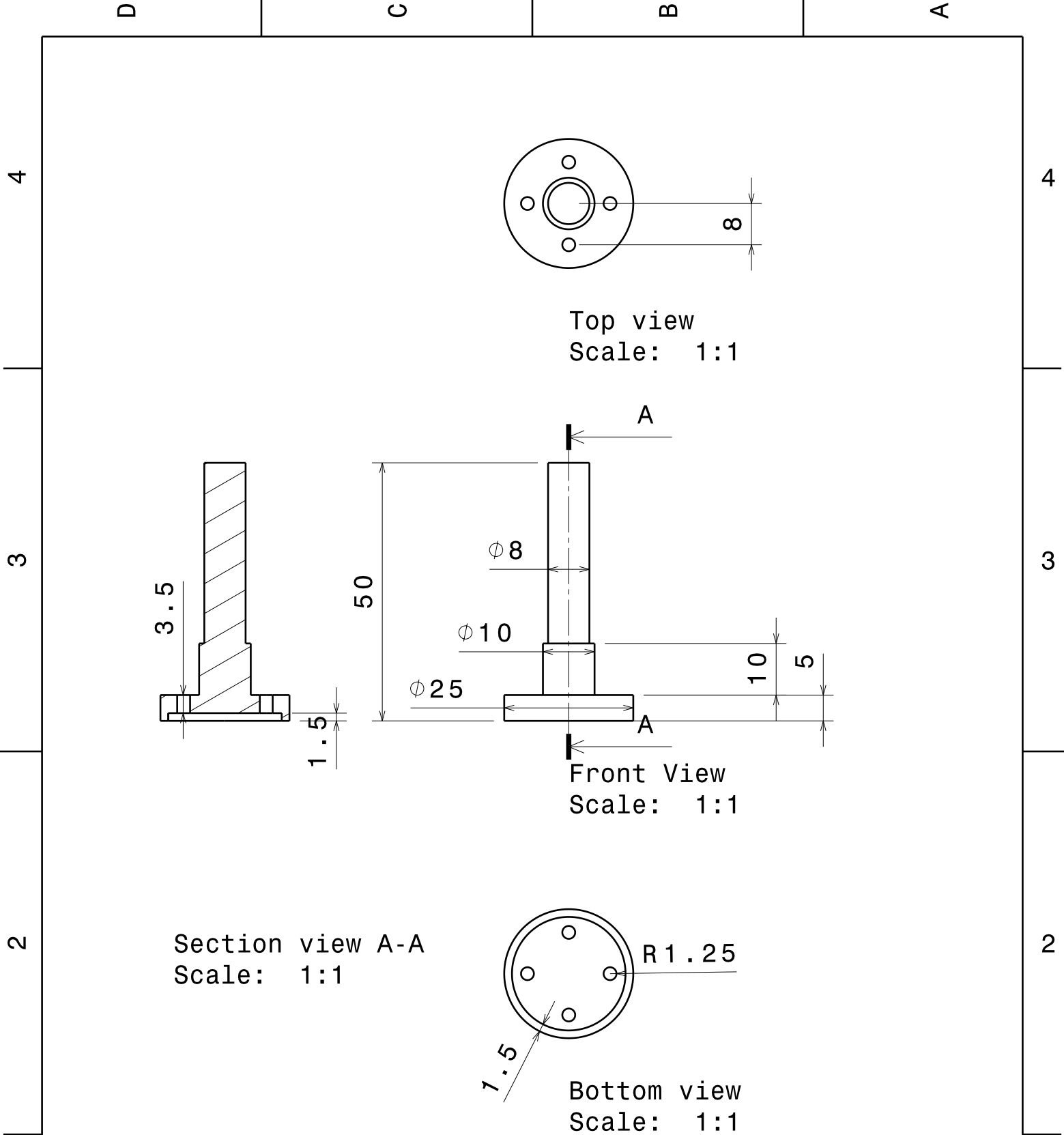
WEIGHT (kg)

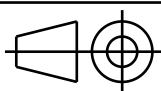
DRAWING NUMBER
CHABOT/TORSO/ROTATION

SHEET
1 / 1

I	—
H	—
G	M8
F	DYN-SERV
E	ROT5-T
D	ROT4-T
C	ROT3-T
B	ROT2-T
A	SK1-T

This drawing is our property; it can't be reproduced or communicated without our written agreement.



DESIGNED BY: CHABOT TEAM	
DATE: 2014	
CHECKED BY:	
DATE:	
SIZE A4	
SCALE 1:1	WEIGHT (kg)

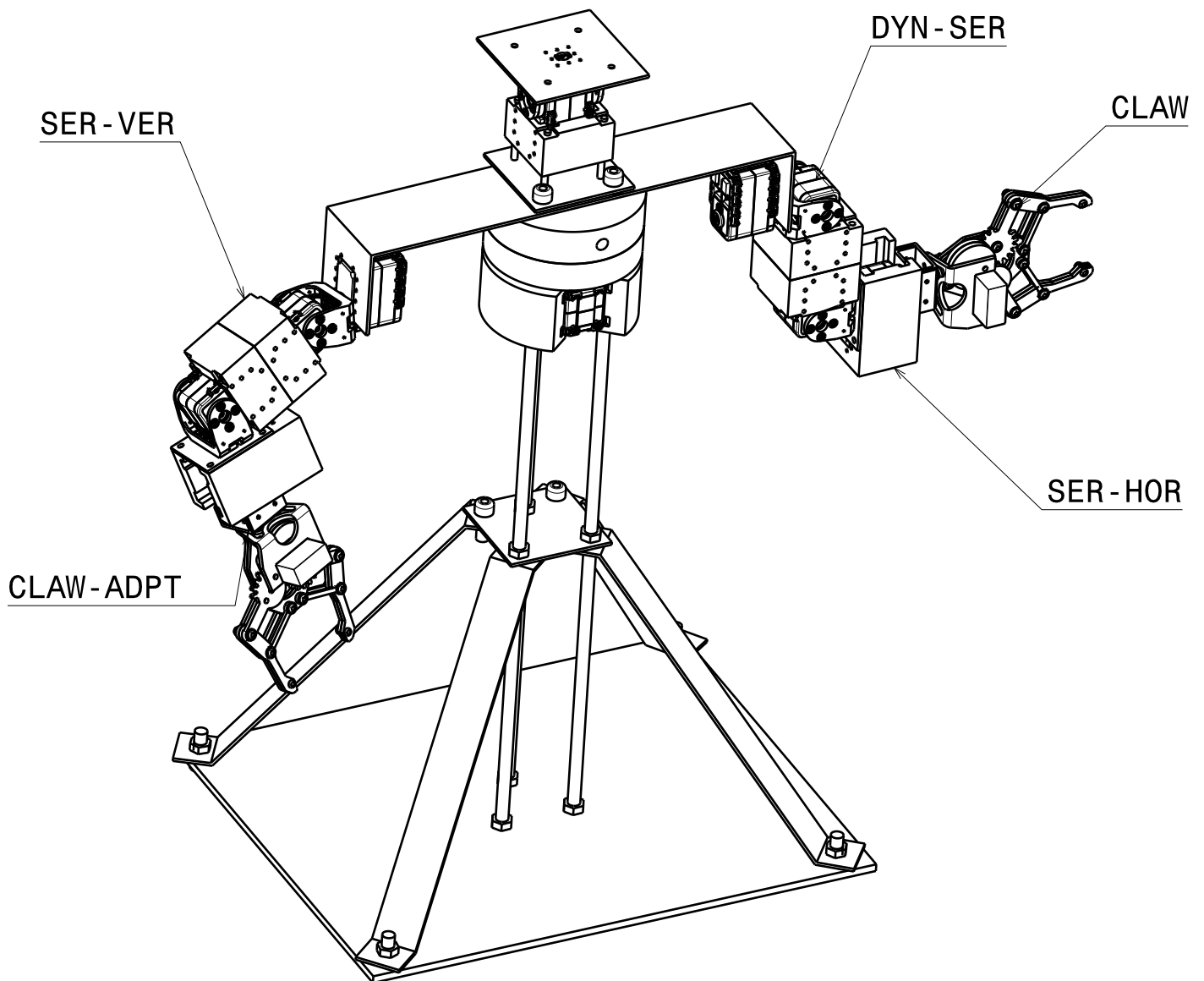
AX12 POWER TRANSMISSION COUPLING	
CHABOT	
DRAWING NUMBER ROT4-T	
SHEET 1 / 1	

I	—
H	—
G	—
F	—
E	—
D	—
C	—
B	—
A	—

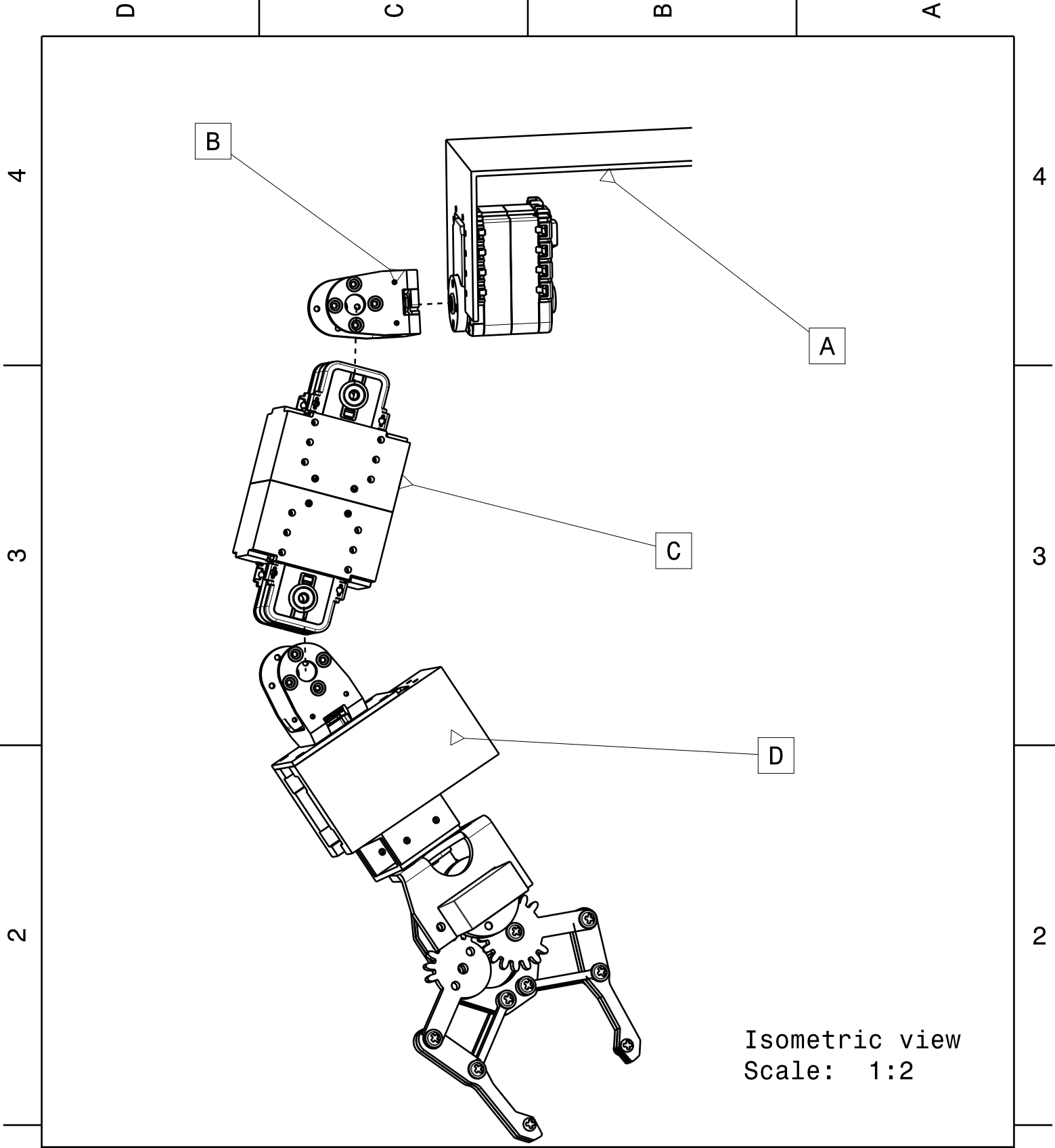
This drawing is our property; it can't be reproduced or communicated without our written agreement.

CHABOT Robot Instruction Manual

Arm Assembly



Isometric view
Scale: 1:4



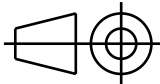
Isometric view
Scale: 1:2

DESIGNED BY:
CHABOT TEAM
DATE:
2014

ARM

I	—
H	—
G	—
F	—
E	—
D	FOREARM
C	UPPERARM
B	DYN-BRA
A	TORSO SHOULDER

SIZE
A4



CHABOT

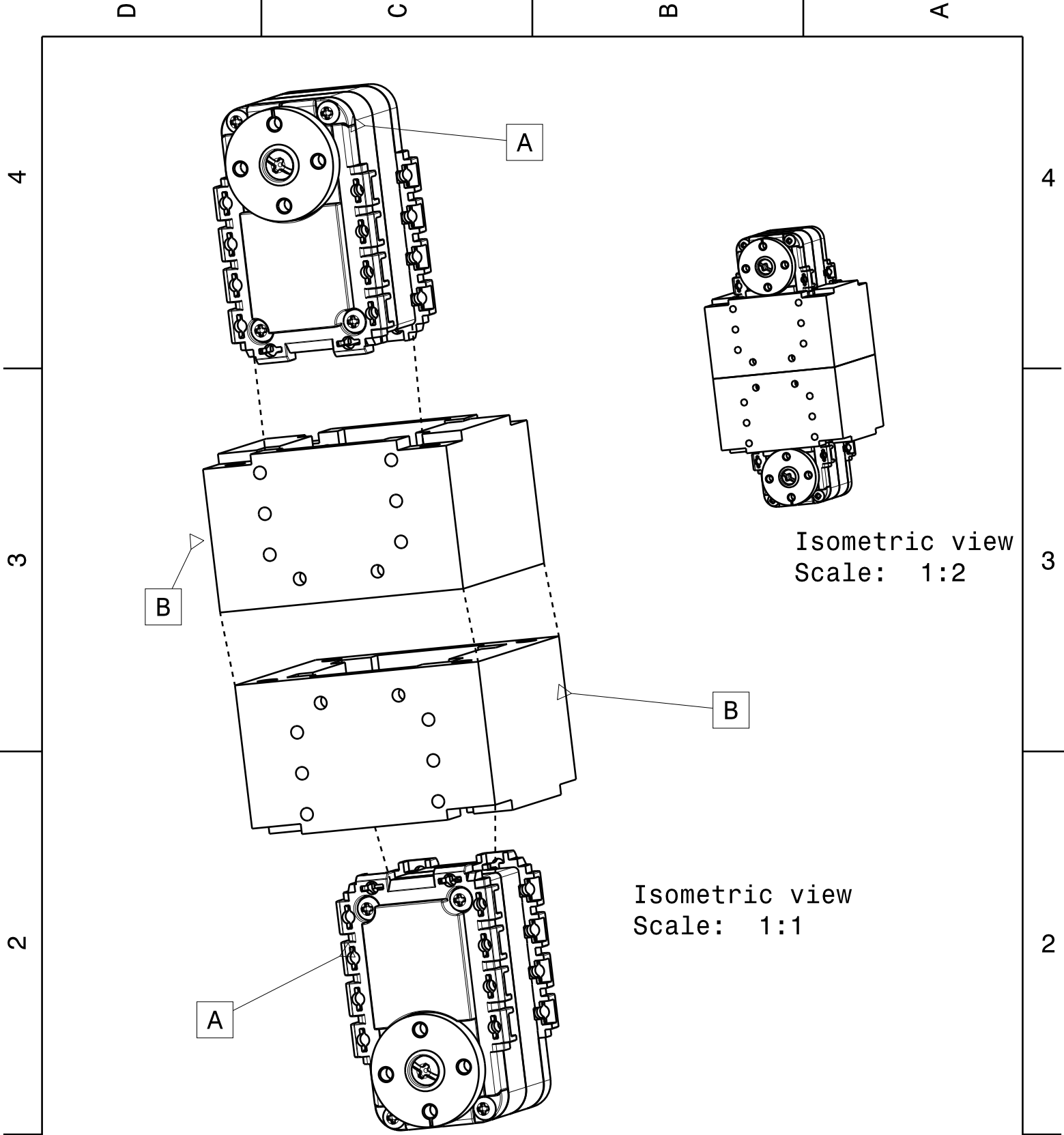
SCALE
1:1

WEIGHT (kg)

DRAWING NUMBER
CHABOT / ARM

SHEET
1 / 1

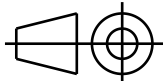
This drawing is our property; it can't be reproduced or communicated without our written agreement.



DESIGNED BY:
CHABOT TEAM
DATE:
2014

UPPER ARM

SIZE
A4



CHABOT

SCALE
1:1

WEIGHT (kg)

DRAWING NUMBER

CHABOT / UPPERARM

SHEET
1 / 1

I	—
H	—
G	—
F	—
E	—
D	—
C	—
B	SER-VER
A	DYN-SERV

This drawing is our property; it can't be reproduced or communicated without our written agreement.

4

4

3

3

2

2

1

1

D

C

B

A

C

A

Isometric view
Scale: 2:3

E

E

B

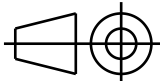
D

F

DESIGNED BY:
CHABOT TEAM
DATE:
2014

FOREARM/CLAW

SIZE
A4



CHABOT

SCALE
1:1

WEIGHT (kg)

DRAWING NUMBER
CHABOT/FOREARM/CLAW

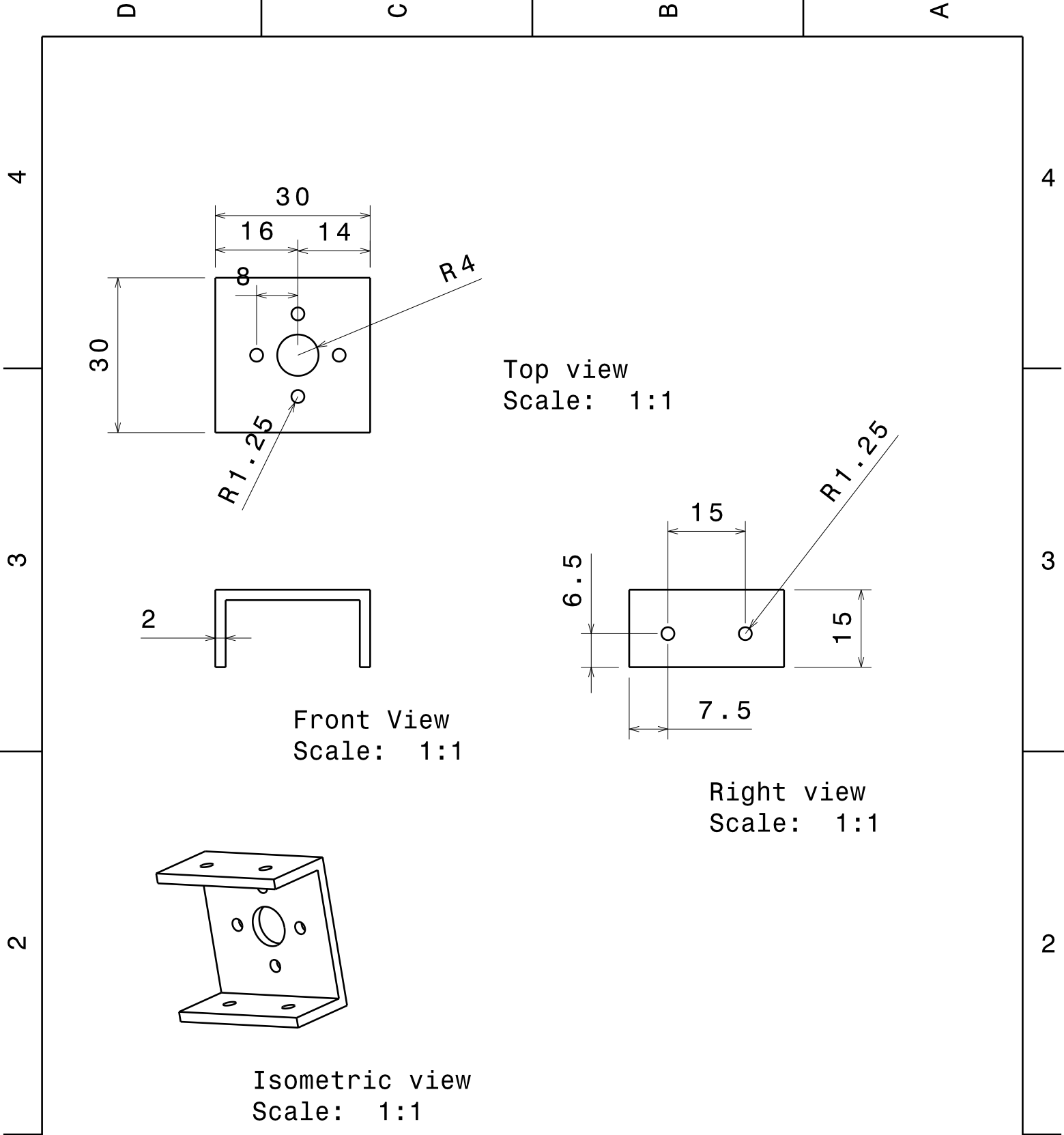
SHEET
1 / 1

I	—
H	—
G	—
F	CLAW
E	CLAW/ADPT
D	SER-HOR
C	DYN-SERV
B	SK2-A
A	DYN-BRA

This drawing is our property; it can't be reproduced or communicated without our written agreement.

D

A

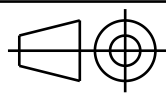


DESIGNED BY:
CHABOT TEAM
DATE:
2014

CLAW ADAPTER COUPLING

I	—
H	—
G	—
F	—
E	—
D	—
C	—
B	—
A	—

SIZE
A4



CHABOT

SCALE
1:1

WEIGHT (kg)

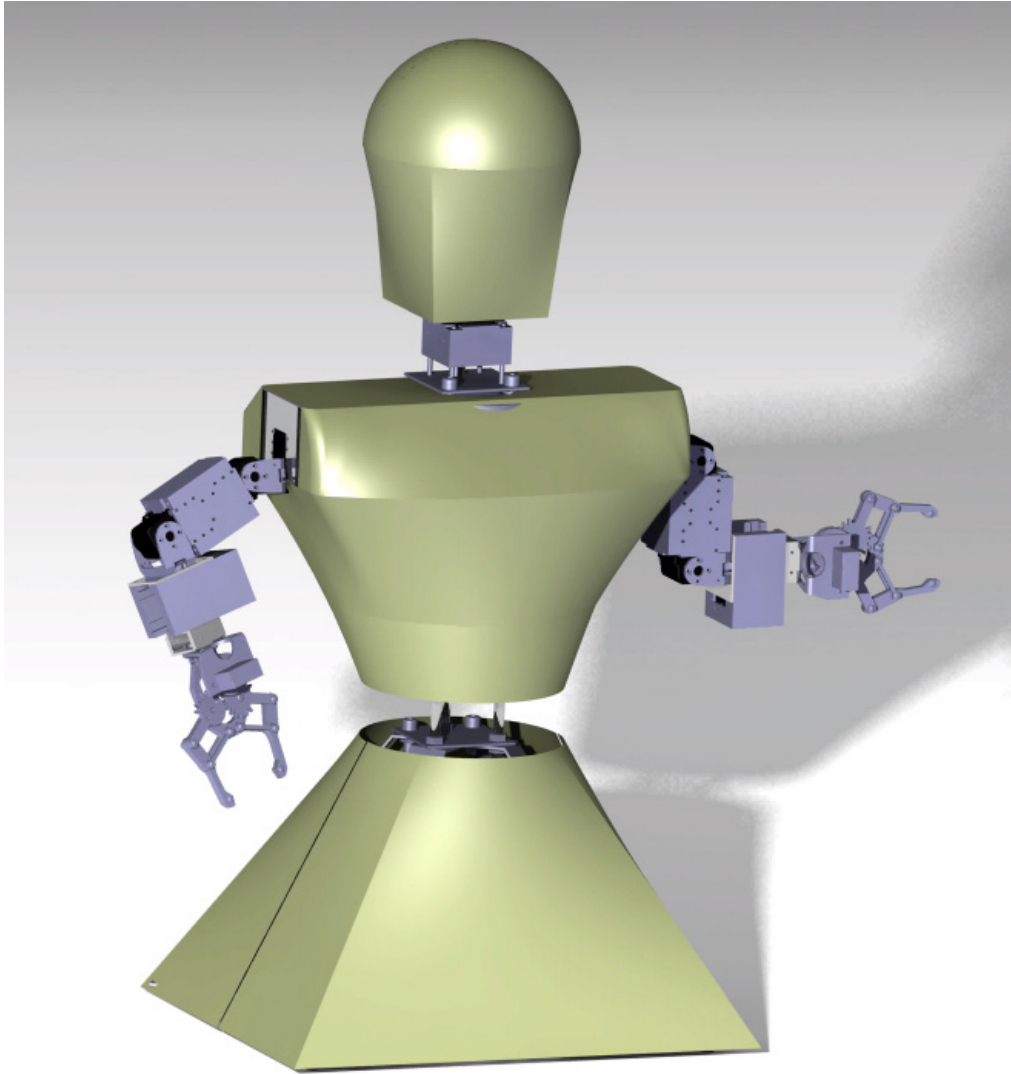
DRAWING NUMBER
CLAW-ADPT

SHEET
1 / 1

This drawing is our property; it can't be reproduced or communicated without our written agreement.

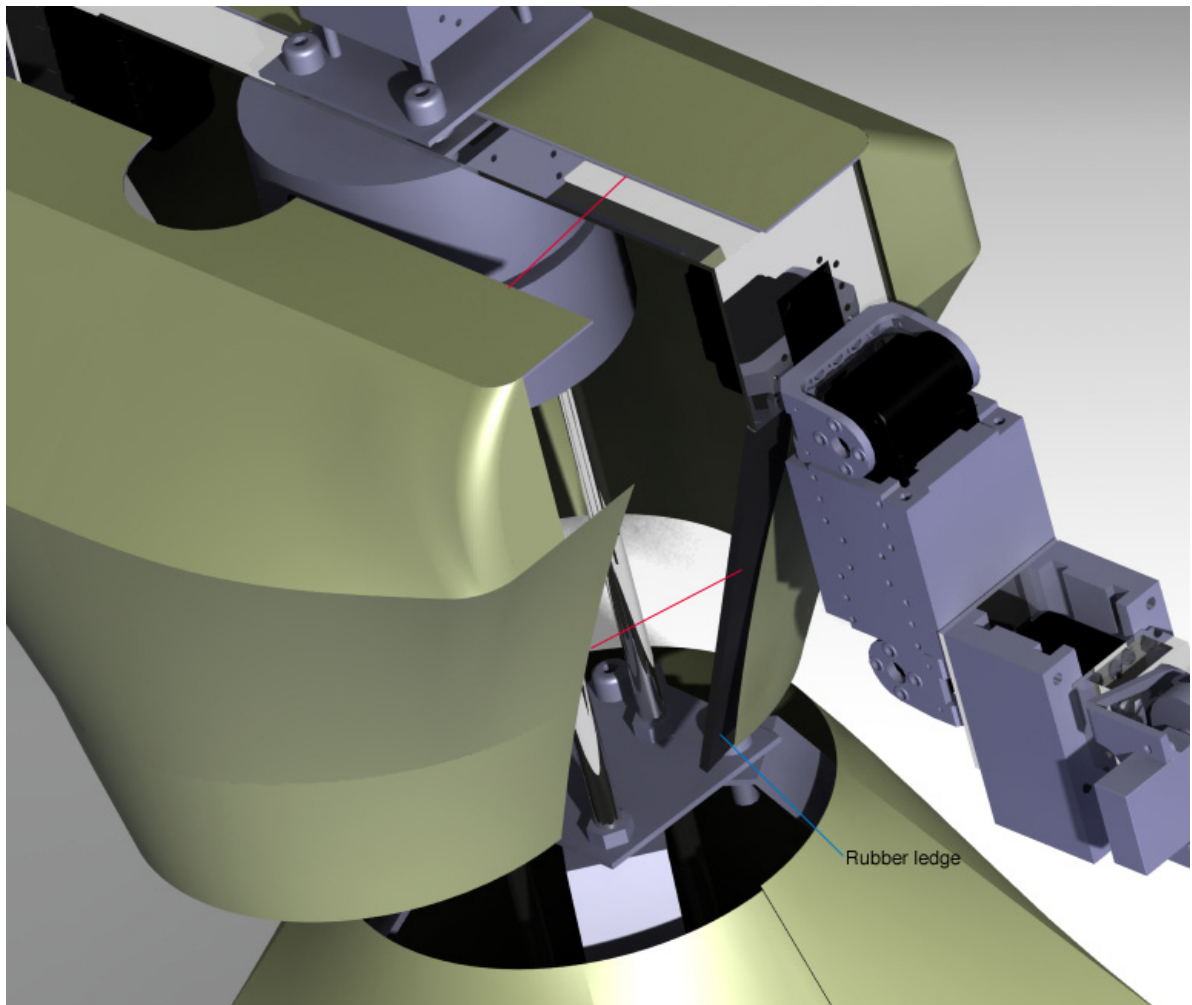
CHABOT Robot Instruction Manual

Shell Assembly



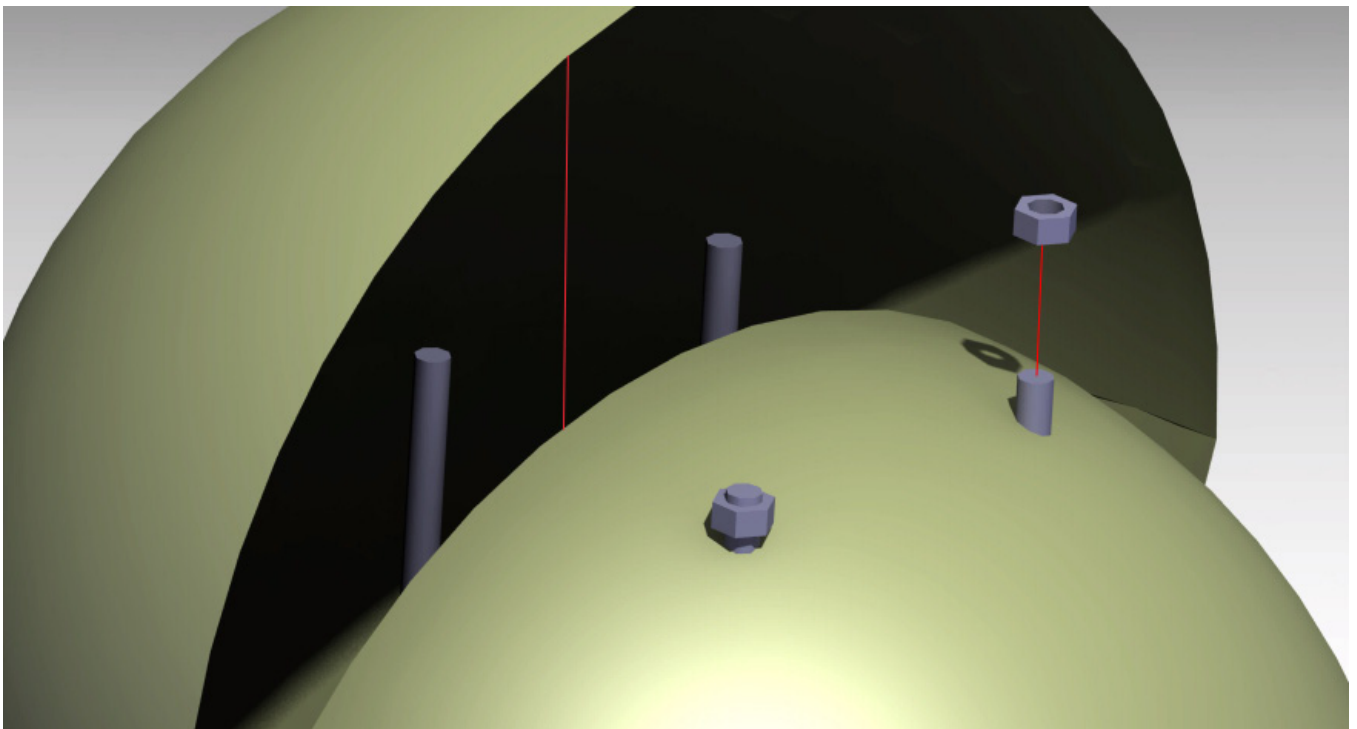
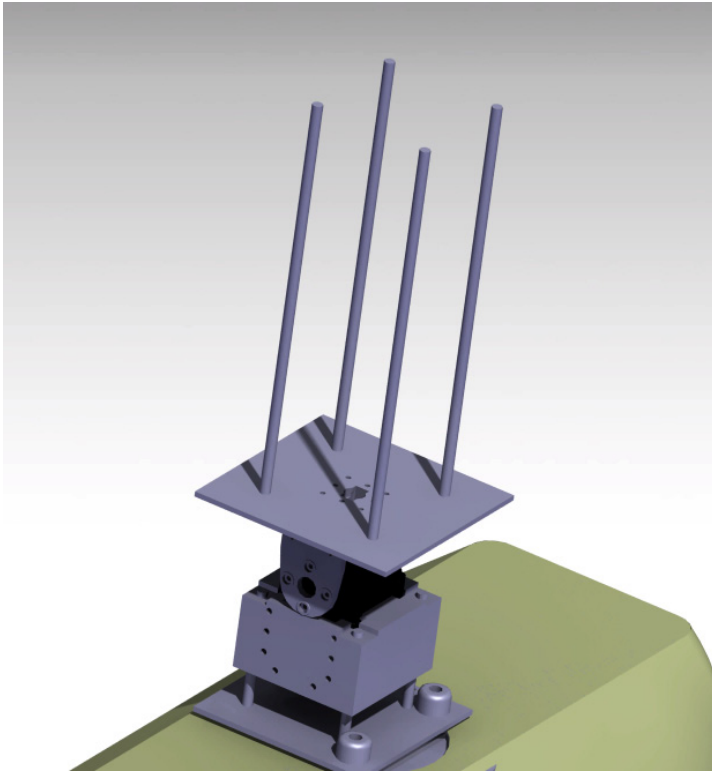
CHABOT Robot Instruction Manual

Torso Shell Assembly



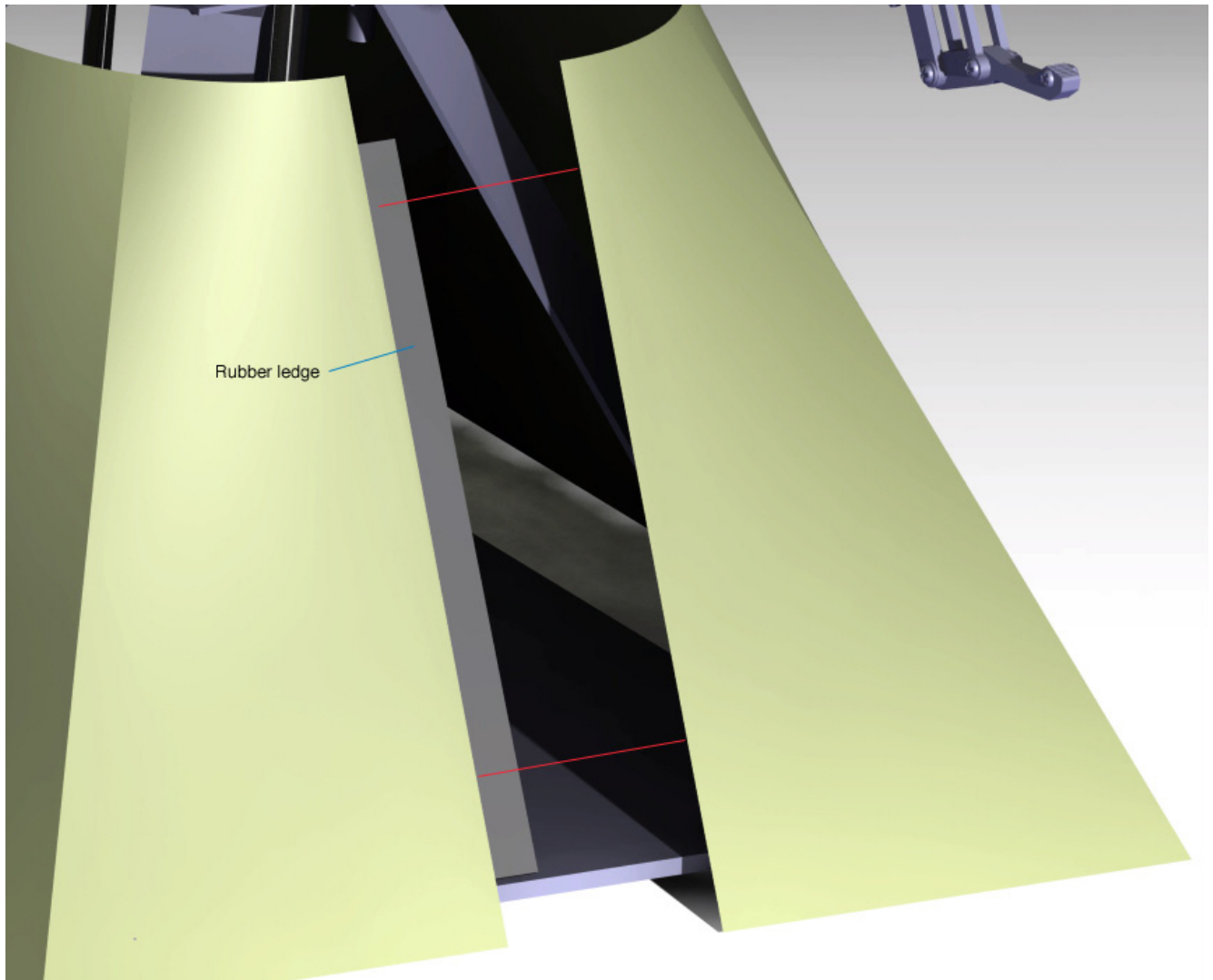
CHABOT Robot Instruction Manual

Head Shell Assembly



CHABOT Robot Instruction Manual

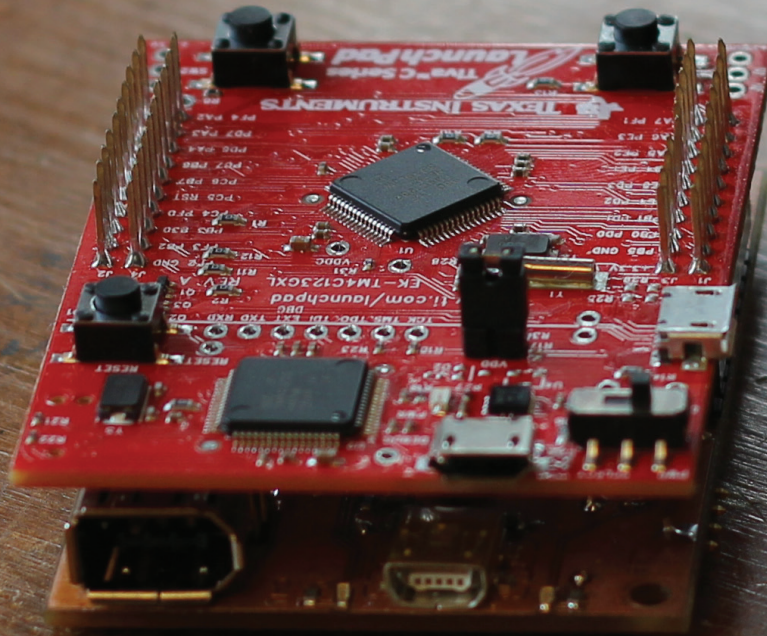
Support Shell Assembly



G Controller manual

Controller manual is omitted in the printed version of the thesis. Please consult the online version.

CHABOT Robot Instruction Manual



Electronic Manual

Chabot Controller

Karin Dankis

Alexander Davidsson

Erik Hardselius

June 6, 2014

Part of ChaBOT bachelor's thesis in Electrical engineering.

Contents

1	Controller registers	7
1.1	\$1xxx Dynamixel	7
1.1.1	\$1nn0 Raw dynamixel package	7
1.1.2	\$1nn1 Servo speed	8
1.1.3	\$1nn2 Servo torque	8
1.1.4	\$1nn3 Servo position	8
1.1.5	\$1nn4 Servo moving	8
1.1.6	\$1nn5 Servor load	9
1.2	\$2xxx IO Devices	9
1.2.1	\$20n0 Digital IO	9
1.2.2	\$21n0 Analog input	10
1.2.3	\$2200 PWM output	10
2	Computer protocol	11
2.1	Message types	11
2.1.1	Example communication	12
2.2	Command set	12
2.2.1	Generic/Low-level	12
2.2.2	Write	13
2.2.3	Read	13
2.3	Devices	14
2.3.1	Servo	14
2.3.2	PWM	16
2.3.3	Analog in	17
2.3.4	Digital in	17
2.3.5	Digital out	18
2.3.6	Display	18
2.4	EEPROM	19
2.4.1	Write	19
2.4.2	Read	19

2.5	Configuration	20
2.5.1	List	20
2.5.2	Write	20
2.5.3	Read	20
3	CAN protocol	21
3.1	Basic protocol	21
3.1.1	Identification	21
3.2	Message format	22
3.2.1	Write message	22
3.2.2	Read message	23
3.2.3	Return message	23
4	CHABOT API	25
4.1	How to use API	25
4.2	Lo-level API	26
4.2.1	addMessage	26
4.2.2	addMessageAsync	27
4.3	Device API	27
4.3.1	setServoPostion	27
4.3.2	setServoPostionAsync	28
4.3.3	getServoPostion	28
4.3.4	getServoPostionAsync	28
4.3.5	setServoSpeed	28
4.3.6	setServoSpeedAsync	29
4.3.7	getServoLoad	29
4.3.8	getServoLoadAsync	30
4.3.9	getServoMoving	30
4.3.10	getServoMovingAsync	30
4.3.11	setPWM	31
4.3.12	setPWMAsync	31
4.3.13	setDigital	31
4.3.14	setDigitalAsync	32
4.3.15	getDigital	32
4.3.16	getDigitalAsync	32
4.3.17	getAnalog	33
4.3.18	getAnalogAsync	33
5	Schematics & PCB	35
5.1	Controller Board	35
5.2	CAN Hub board	40

5.3 Power supply	42
----------------------------	----

Chapter 1

Controller registers

CHABOT controller contains parameter which is used to control the different devices. Main difference with these parameters from a normal memory-map based system is that each address stores different multiples of bytes.

1.1 \$1xxx Dynamixel

Register area 1000 to 1fff is dedicated for Dynamixel servo motors. On this controller contains 6 different parameters to read from or written to. 'nn' should be replaced with the ID of the dynamixel.

Address		What	Usage
\$1nn0	r/w	Raw dynamixel package	Allow sending a raw command to dynamixel 'n'.
\$1nn1	w	Servo speed	Update dynamixel 'n' speed.
\$1nn2	w	Servo torque	Update dynamixel 'n' speed.
\$1nn3	r/w	Servo position	Update or returns dynamixel 'n' position.
\$1nn4	r	Servo moving	Returns if dynamixel 'n' is moving or not.
\$1nn5	r	Servo load	Returns dynamixel 'n' current load.

Table 1.1: ChaBOT registers

1.1.1 \$1nn0 Raw dynamixel package

This register provides direct access to the dynamixel's own bus. A typical message starts with a 2 byte address followed by one or more bytes.

Example:

```
write this 0x1020 0x1e 0x01 0xff
```

1.1.2 \$1nn1 Servo speed

Changes speed on a servo. Requires 1 16bit parameter or 2 8 bit parameters containing the new speed. Valid values are between 0 to 0x1ff.

Example:

```
write this 0x1061 0x1ff
```

1.1.3 \$1nn2 Servo torque

Changes a servos torque. Requires 1 16bit parameter or 2 8 bit parameters containing the new torque limit. Valid values are between 0 to 0x3ff.

Example:

```
write this 0x11a2 0x3ff
```

1.1.4 \$1nn3 Servo position

Changes a servos torque. Requires 1 16bit parameter or 2 8 bit parameters containing the new torque limit. Valid values are between 0 to 0x3ff. Can be used to return current position of the servo. Note that servo position is not known before a initial position is set.

Example:

```
write this 0x1013 0x100
```

1.1.5 \$1nn4 Servo moving

Return a non zero response if the servo is moving.

Example:

```
read this 0x1034
```

1.1.6 \$1nn5 Servor load

Return a load. Where a value between 0 to 1023 is a load in counter clock wise direction where a value between 1024 to 2047 us a load in clock wise direction. Value 1024 is equal to 0.

Example:

```
read this 0x1035
```

1.2 \$2xxx IO Devices

IO devices are located in the 2xxx to 2fff region.

Address		What	Usage
\$20n0	r/w	Digital io	Set a digital out put or reads a digital input.
\$21n0	r	Analog input	Returns value of a analog input.
\$2200	w	PWM output	Sets duty cycle on PWM channel 0.

Table 1.2: ChaBOT registers

1.2.1 \$20n0 Digital IO

The controller board contains 2 digital input and 2 digital outputs. The input and the output is located on different pins. Reading a register will return current state on input. If a input is active it will return 1 otherwise it will return 0. When writing it will change state on the addressed output pin.

Type	ID	PIN on Tiva board
Input	0	PF0
Input	1	PF4
Output	0	PE1
Output	1	PE2

Table 1.3: Pinout for digital input and output

1.2.2 \$21n0 Analog input

Return the current value from a 12bit analog to digital converter located on pin PB2 on the Tiva board.

1.2.3 \$2200 PWM output

Sets the duty rate on the PWM. A value between 0 to 255 is expected.

Chapter 2

Computer protocol

To communicate with the main controller a simple serial protocol is used. The main controller also called Master will then communicate thru CAN to other nodes. A controller can be in two different modes. One mode is called Debug mode which will turn the communication to a console and the non debug mode is intended to be used with a library. The C library will not work with a controller in Debug mode.

2.1 Message types

A Controller can respond with 5 different message types. All messages are sent as a line except # which is a indicator that a controller is ready for a new command.

Hash is not a message by is sent from the controller when it's ready to receive a command. This response will not send a new line character.

S Start message is the first message returned from a command, and it is used to return command id, which is used to send response on a specific command. S [7] ;

E Error message is used to send error messages that is related to a specific command. E [7]
An error message here

// Comment message is a message that is safe to ignore, it's main function is to make debugging easier.

R Return message is the last message related to a command. This message will return status and potential data. R [3] ERR; , R [7] OK; or R [7] OK (2) 0x1ff 0x22

2.1.1 Example communication

```

⇒#
⇐set 7 servo 2 position 0x1ff↵
⇒S [22]↵
⇒// Write dyanmixel(2) position: 0x01ff↵
⇒// Package recived from: 0x02↵
⇒// Length: 0↵
⇒// Status: No error↵
⇒// Checksum OK!↵
⇒R [4] OK;↵
⇒#
⇐set 7 servo 3 position 0x1ff↵
⇒S [3];↵
⇒// Write dyanmixel(1) position: 0x01ff↵
⇒E [3] No response detected↵
⇒R [3] ERR;↵

```

Above is a example of a communication with a controller where \Rightarrow means that it's data from the controller and \Leftarrow indicates that it's from the user or a computer. \Leftarrow symbols indicates a new line character. The above communication shows a computer instructing 2 different servos to change their position to 0x01ff. Communication to servo 2 dose succeed where servo 3 doesn't respond and a error is retuned.

2.2 Command set

This is a list of commands supported by a controller. Each command will be presented with a line similar to following line:

```
w(rite) [ctrl] [reg] [value]
```

Where characters inside '()' can be omitted for a shorter command and character inside '[]' are parameters.

2.2.1 Generic/Low-level

There are two commands which is gives a user a low level access to different registers on a controller. These function are intended for communication with devices which contains

perhpiral not supported by the controller.

2.2.2 Write

Write is a small command to send a raw register write to a specific address

w(rite) [ctrl] [reg] [value]

Parameter	Function	Commona value
ctrl	Specify target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
reg	A register address	A value from 0 to 65535 used by a device
value	A value to be written	On ore more integers to be stored

Examples:

write this 0x1023 22

write 255 25 0x23 0x4f 0x01ff

w 7 0x1523 0x22

2.2.3 Read

Write is a small command to send a raw register write to a specific address

r(ead) [ctrl] [reg]

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
reg	A register address	A value from 0 to 65535 used by a device

Examples:

read this 0x1023

read 255 25

r 7 0x1523

2.3 Devices

There is also a sub set commando designed for simplifying control of different devices. In this revision only Servo, PWM, Analog I/O and Digital I/O is supported. These are controlled thru *set* and *get* commands.

`s(et) [ctrl] [device]`

`g(et) [ctrl] [device]`

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital

2.3.1 Servo

Functions used to control dynamical type of servo.

Speed

Changes moving speed.

`s(et) [ctrl] s(ervo) [ch] s(peed) [value]`

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	Servo id	A id from 0 to 254
value	Speed	0x000-0x1ff

Examples:

`read this 0x1023`

`read 255 25`

`r 7 0x1523`

Position

Changes position or reading current position of a servo

s(et) [ctrl] s(ervo) [ch] p(osition) [value]

g(et) [ctrl] s(ervo) [ch] p(osition)

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	Servo id	A id from 0 to 254
value	Speed	0x000-0x1ff

Examples:

set this servo 5 position 0x100

set 7 servo 1 position 0x1ff

s this s 5 p 0x1ff

Torque

Defines torque limit.

s(et) [ctrl] s(ervo) [ch] t(orque) [value]

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	Servo id	A id from 0 to 254
value	Torque	0x000-0x1ff

Examples:

set this servo 5 torque 0x100

set 7 servo 1 torque 0x1ff

s this s 5 t 0x1ff

Moving

Returns if a servo is moving or not.

`g(et) [ctrl] s(ervo) [ch] m(oving)`

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	Servo id	A id from 0 to 254

Examples:

`set this servo 5 moving`

`set 7 servo 1 m`

`s this s 5 m`

Load

Returns current load on a specific servo.

`g(et) [ctrl] s(ervo) [ch] l(oad)`

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	Servo id	A id from 0 to 254

Examples:

`set this servo 5 load`

`set 7 servo 1 load`

`s this s 5 l`

2.3.2 PWM

Controls a PWM signal with a specific duty cycle.

s(et) p(wm) [ch] [value]

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	channel	Only on channel is available
value	duty cycle	A value from 0 to 255

Examples:

```
set 7 pwm 0 0x00
```

```
s this p 0 0xff
```

2.3.3 Analog in

Allows reading of analog signal from a external device or sensor.

g(et) a(nalog) [ch]

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
ch	channel	Only on channel is available

Examples:

```
set 7 analog 0
```

```
s this a 0
```

2.3.4 Digital in

Digital inputs

g(et) d(igital) [ch]

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device ch	A device name channel	servo, pwm, analog, digital 0 to 4

Examples:

```
set 7 digital 0
s this d 2
```

2.3.5 Digital out

Digital outputs

s(et) d(igital) [ch] [value]

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device ch	A device name channel	servo, pwm, analog, digital 0 to 4
value	output	1 or 0

Examples:

```
set 7 digital 0 1
s this d 2 0
```

2.3.6 Display

Writing data to display row

s(et) display [row] [value]

Examples:

```
set 7 display 0 1
s this dpy 2 0
```

Parameter	Function	Commona value
ctrl	Target controller	A node number between 1 to 254 for a external node. 0 or 'this' for pointing to same node or 255 for broadcast.
device	A device name	servo, pwm, analog, digital
row	row number	0 to 8
value	line data	A 16bit b/w image data

2.4 EEPROM

Functions to alter a controllers EEPROM. DO NOT USE THESE FUNCTIONS.

2.4.1 Write

Function to write a 32bit integer to the controllers EEPROM

`ew(rite) [reg] [value]`

Parameter	Function	Commona value
reg	Address	A value from 0 to 511.
value	line data	A 16bit b/w image data

Examples:

`ewrite 0 0xff00ff00`

`ew 2 0x00ff00ff`

2.4.2 Read

Function to read a 32bit integer from the controllers EEPROM

`er(ead) [reg]`

Parameter	Function	Commona value
reg	Address	A value from 0 to 511.

Examples:

`eread 0`

er 2

2.5 Configuration

Configuration is used to set different parameter used by the controller such as CAN id or to activate/deactivate debug mode.

2.5.1 List

Lists all parameters values.

```
c(onfig) l(ist)
```

2.5.2 Write

Writes a value to a paramter.

```
c(onfig) w(rite) [parameter] [value]
```

2.5.3 Read

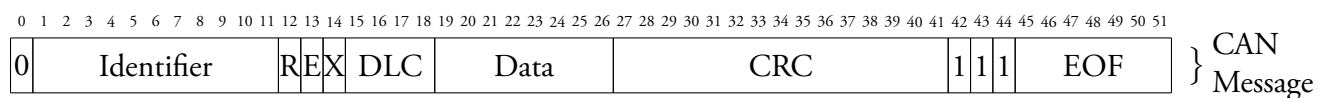
```
c(onfig) r(ead) [parameter]
```

Read out a parameter.

Chapter 3

CAN protocol

CAN is a serial bus designed by Bosch in 1983. ChaBOT uses a 1 Mb/s bus with 11 bit id.



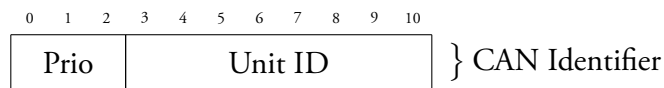
CAN support message from 1 to 8 byte.

3.1 Basic protocol

The basic protocol used is based around the idea of registers. All commands is sent thru a small command set which let you write or read registers located in the controller. The CAN protocol itself dose not determine how different type of units is controlled, this is determined with help of register maps that expose the different parameters of each type of unit.

3.1.1 Identification

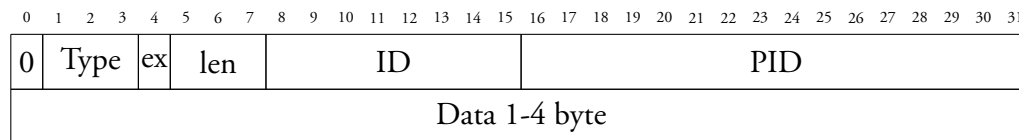
A bus can contain 254 units with a ID between 1 to 254. ID 0 is reserved for special types of broadcast and 255 is the dedicated broadcast address. Unit id constitute the lower 8 bit of the CAN id where the upper 3 bit is priority. Controller firmware will translate ID 0 to it's own ID.



3.2 Message format

There are currently 3 different messages supported by the CAN protocol: Read message, Write message and Return message.

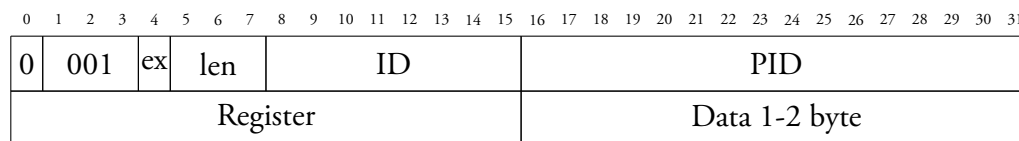
Structure of a typical message is a 3 to 8 byte size string containing



Name	Function
Type	Indicate message type
ex	Indicates multiple message in transfer (NOT IMPLEMENTED)
len	Length of response in message
ID	Senders CAN id
PID	Message ID.

3.2.1 Write message

A write message is a to a register with max 2 bytes of data.



Name	Function
Register	Which register in controller to written to.
Data	1 to 2 bytes of data.

3.2.2 Read message

Reads a register from a controller.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	010	ex	len	ID												PID															
Register																															

Name	Function
Register	Which register in controller to read from.

3.2.3 Return message

Response generated from a write or read message.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	011			ex	len			ID								PID															
Status								Data 1-3 byte																							

Name	Function
Status	Status returns a non zero value on error
Data	1 to 3 bytes of data. (Only on read message)

Chapter 4

CHABOT API

CHABOT API is a used by a computer to communicate with CHABOT control system. It's written in C++ and based on boost library.

4.1 How to use API

```
#include <iostream>
#include <boost/lexical_cast.hpp>
#include "Runner.h"

int main(int argc, char* argv[])
{
    std::cout << "CLIENT Starting Runner" << std::endl;
    chabot::Runner* runner = new chabot::Runner("/dev/cu.usbserial", 9600);

    if (runner->isReady())
    {

        getchar();
        std::cout << "CLIENT Send message to node (NON BLOCKING)" << std::endl;
        runner->setServoPosition(1, 1, 0x3ff);

        getchar();
```

```

std::cout << "CLIENT_Send_message_to_↵
node_(NON_BLOCKING)" << std::endl;
runner->setServoPosition(1,1,0x100);

getchar();
std::cout << "CLIENT_Send_message_to_↵
node_(BLOCKING)" << std::endl;
runner->setServoPosition(1,1,0x1ff);
std::cout << "CLIENT_Send_message_to_↵
node_(BLOCKING)-DONE" << std::endl;

getchar();
std::cout << "Destroying_runner" << std↵
::endl;
delete runner;
}
std::cout << "END" << std::endl;
return 0;
}

```

To connect to CHABOT a class called *chabot::Runner* is used. This call will return a exception if no serial port is found. There are both a hi level api that abstracts the controller and a low level allowing to send raw message directly to the controller board available thru this class.

```

chabot::Runner* runner = new chabot::Runner("/dev/cu.↵
usbserial",9600);

```

To disconnect from the controller the delete function is used. This will cause the API to stop and disconnect from the controller.

```

delete runner;

```

4.2 Lo-level API

addMessage and addMessageAsync sends a raw message to a controller. They uses the syntax from Chapter 2. These function is intended to allow raw control of the controller and is used by the higher-level api to execute their commands.

4.2.1 addMessage

```
int chabot::Runner::addMessage(std::string message, int ←
    *length, int maxlength, unsigned char *data);
```

This command will send a command and wait for the response from the controller.

Parameter	Function
message	A String containing a command
length	Pointer to a integer for storing received length.
maxlength	Buffer size
data	Pointer to a local buffer

4.2.2 addMessageAsync

```
void chabot::Runner::addMessageAsync(std::string message ←
    , MessageCbl cbl);
```

Parameter	Function
message	A String containing a command
cal	Is a callback function which is called when request is done.

addMessageAsync dose a non blocking function. This mean that the function doesn't wait for the request to be completed, instead it will allow the program to execute next instruction.

4.3 Device API

Device API is a set of function that is used for different input and output related to the controller. The asynchronous function contains a callback function that is used to return information such as received information and status. Callback function has no return value and only a pointer as a parameter.

```
void callbackfunction(Message* msg)
```

4.3.1 setServoPostion

```
int chabot::Runner::setServoPostion(int controller, int ←
    servo, unsigned short position);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
position	A position value between 0x000 - 0x3ff where 0x1ff is center position.

setServoPostion change a position on a servo.

4.3.2 setServoPostionAsync

```
int chabot::Runner::setServoPostionAsync(int controller ,  $\leftarrow$ 
int servo , unsigned short position , MessageCbl cbl);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
position	A position value between 0x000 - 0x3ff where 0x1ff is center position.
cbl	Is a callback function which is called when request is done.

setServoPostion change a position on a servo.

4.3.3 getServoPostion

```
int chabot::Runner::getServoPostion(int controller , int  $\leftarrow$ 
servo , unsigned short *position);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
position	A pointer to a short to store result in

Retrieves current position form servo.

4.3.4 getServoPostionAsync

```
int chabot::Runner::getServoPostion(int controller , int  $\leftarrow$ 
servo , MessageCbl cbl);
```

Retrieves current position form servo.

4.3.5 setServoSpeed

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
cbl	Is a callback function which is called when request is done.

```
int chabot::Runner::setServoSpeed(int controller , int ↵
servo , unsigned short speed);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
speed	A speed value between 0x000 - 0x3ff where 0x3ff is max speed.

setServoSpeed change the speed on a servo.

4.3.6 setServoSpeedAsync

```
int chabot::Runner::setServoSpeedAsync(int controller , ↵
int servo , unsigned short speed , MessageCbl cbl);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
speed	A speed value between 0x000 - 0x3ff where 0x3ff is max speed.
cbl	Is a callback function which is called when request is done.

setServoSpeed change the speed on a servo.

4.3.7 getServoLoad

```
int chabot::Runner::getServoLoad(int controller , int ↵
servo , unsigned short *value);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
value	A pointer to store the result.

Retrieves current load on servo.

4.3.8 getServoLoadAsync

```
int chabot::Runner::getServoLoad(int controller , int ↵
    servo , MessageCbl cbl);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
cbl	Is a callback function which is called when request is done.

Retrieves current load on servo.

4.3.9 getServoMoving

```
int chabot::Runner::getServoLoad(int controller , int ↵
    servo , unsigned char *value);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
value	A pointer to store the result.

Retrieves if servo is moving servo. Data is non-zero while a servo is moving.

4.3.10 getServoMovingAsync

```
int chabot::Runner::getServoLoad(int controller , int ↵
    servo , MessageCbl cbl);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
cbl	Is a callback function which is called when request is done.

Retrieves if servo is moving servo. Data is non-zero while a servo is moving.

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
val	Set PWM ration.

4.3.11 setPWM

```
int chabot::Runner::setPWM(int controller , int servo ,  $\leftarrow$ 
unsigned short position);
```

Controls the ratio of the PWM signal where a large value is equal to a longer pulse.

4.3.12 setPWMAsync

```
int chabot::Runner::setPWMAsync(int controller , int ch ,  $\leftarrow$ 
unsigned char val , MessageCbl cbl);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
val	Set PWM ration.
cbl	Is a callback function which is called when request is done.

Controls the ratio of the PWM signal where a large value is equal to a longer pulse.

4.3.13 setDigital

```
int chabot::Runner::setDigital(int controller , int ch ,  $\leftarrow$ 
unsigned char val);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
val	Set 1 or 0 to change output.

Controls the digital output pins.

4.3.14 setDigitalAsync

```
int chabot::Runner::setDigitalAsync(int controller , int ↵
    ch , unsigned char val , MessageCbl cbl );
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
val	Set 1 or 0 to change output.
cbl	Is a callback function which is called when request is done.

Controls the digital output pins.

4.3.15 getDigital

```
int chabot::Runner::getDigital(int controller , int servo ↵
    , unsigned char *value );
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
value	Pointer to store the result.

Retrieves value of a digital input.

4.3.16 getDigitalAsync

```
int chabot::Runner::getDigitalAsync(int controller , int ↵
    servo , MessageCbl cbl );
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
cbl	Is a callback function which is called when request is done.

Retrieves value of a digital input.

4.3.17 getAnalog

```
int chabot::Runner::getAnalog(int controller , int servo ,  $\leftarrow$ 
    unsigned short *value);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
value	A pointer to where to store the 12 bit analog value returned from the controller.

Retrieves the 12bit value of a analog input.

4.3.18 getAnalogAsync

```
int chabot::Runner::getAnalogAsync(int controller , int  $\leftarrow$ 
    servo , MessageCbl cbl);
```

Parameter	Function
controller	Controller id, 0 to address the local controller.
servo	A servo id.
cbl	Is a callback function which is called when request is done.

Retrieves the 12bit value of a analog input.

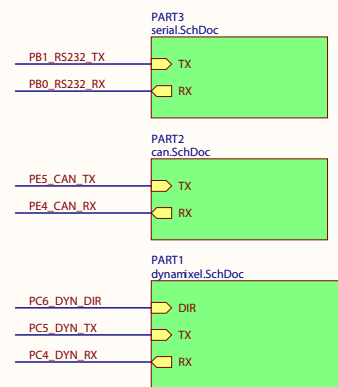
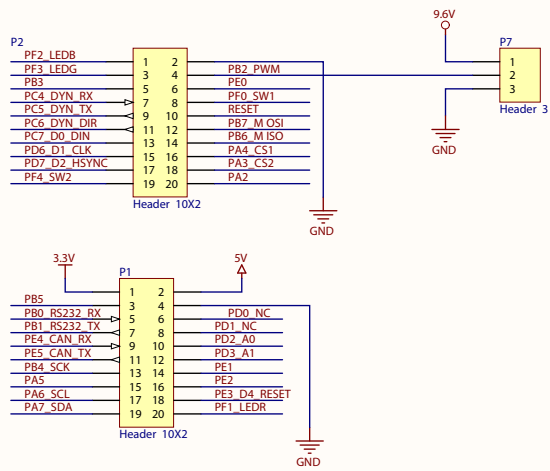
Chapter 5

Schematics & PCB

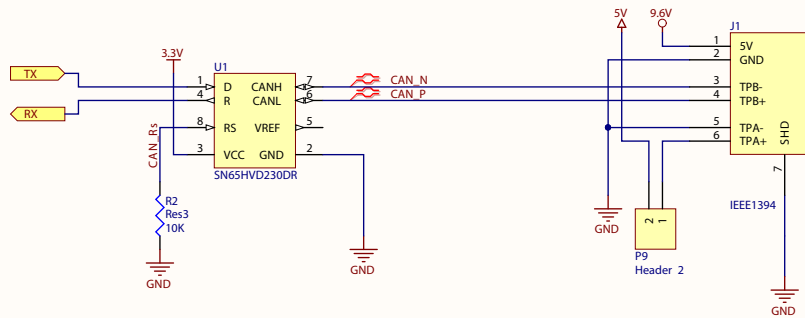
In following pages is the schematic for the different custom boards used in CHABOT project.

5.1 Controller Board

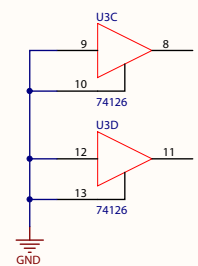
Controller board contains a CAN transceiver based on Texas Instruments SN65HVD320 used for communication between the different controllers. There is also components to allow connection to a Dynamixel. It's based on a 74HCT126 that is both used for direction control and logic level conversion. Last there is a USART to USB circuit with is based on FTDI's FT230X which is a USB to serial controller common in USB to RS232 adapters.



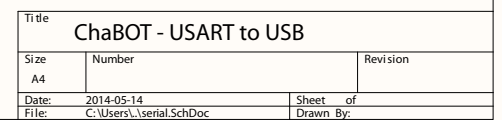
Title		
ChaBOT - Top schematic		
Size	Number	Revision
A4		
Date:	2014-05-14	Sheet of
File:	C:\Users\...\chaobot.mkl1.SchDoc	Drawn By:



Title		
ChaBOT - CAN Transceiver		
Size	Number	Revision
A4		
Date:	2014-05-14	Sheet of
File:	C:\Users\...\can.SchDoc	Drawn By:



Title			ChaBOT - Dynamixel		
Size		Number		Revision	
A4					
Date:	2014-05-14			Sheet	of
File:	C:\Users\...\dynamixel\SchDoc			Drawn By:	



5.2 CAN Hub board

CAN hub is used for interconnection between the controller boards. It's a passive backplane and doesn't contain any logic it's self. It's intended to be connected to a power supply and a 120Ω terminator. CAN busses requires two terminators, but the power supply contains one the terminators.

5.3 Power supply

The Power supply provides 5V with maximum 3A. It's a switched power supply based on a buck converter, using a low switch frequency.

