UNIVERSITAT POLITÈCNICA DE CATALUNYA

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

A DEGREE THESIS

Application with data acquisition of a FESTO planar surface gantry EXCM

Author:
Xavier Vilanova Bernal

Advisor:
Jan Pascual Alsina

Barcelona, October 2016

Acknowledgements

I would like to express my appreciation and thanks to all those who, through their cooperation, have contributed to the realization of this Final Project:

Firstly, my sincere thanks to my tutor in this project Jan Pascual, for giving me the opportunity to do it.

Special thanks to the company Festo for taking the time to meet us and invite us to their facilities for training courses fulfill ourselves and for allowing us the material

And finally, to my dearest and closest family, despite the setbacks that have arisen, such as work and personal problems, they have known how to help me in the course, and they have encouraged me every moment. Most especially to my partner, Laura.

Thank you so much

Abstract

The following document contains the Final Project Degree of Xavier Vilanova Bernal.

It is based on process automation in the field of biomedicine and trying to facilitate the work of a biologist and / or doctor.

The implementation of this project is to cultures microorganisms in petri dishes, which so far have been managed manually, can make a big improvement in many aspects for laboratories and hospitals.

For this purpose, it has been designed, developed and manufactured a prototype containing a 3-axis robot, an adapted clamp and an OpenSource controller with connectivity to a communication platform with Smartphones.

Throughout the entire document has everything you need to understand and reproduce this prototype.

Resum

El següent document conté el Projecte de Final de Grau (PFG) de Xavier Vilanova Bernal.

Es basa en l'automatització de processos en l'àmbit de la biomedicina i que intenta facilitar les tasques d'un biòleg i / o mèdic.

L'aplicació d'aquest projecte és als cultius de microorganismes en les plaques de Petri, que fins al moment s'han gestionat de manera manual, pot suposar una gran millora en molts aspectes per a laboratoris i hospitals.

Per a aquesta finalitat, s'ha dissenyat, desenvolupat i fabricat un prototip que conté un robot de 3 eixos, una pinça adaptada i una controladora OpenSource amb connectivitat a una plataforma de comunicació amb Smartphones.

Al llarg de tot el document es trobarà tot el necessari per entendre i reproduir aquest prototip.

Resumen

El siguiente documento contiene el Proyecto de Final de Grado (PFG) de Xavier Vilanova Bernal.

Se basa en la automatización de procesos en el ámbito de la biomedicina y que intenta facilitar las labores de un biólogo y/o médico.

La aplicación de este proyecto es a los cultivos de microorganismos en las placas de Petri, que hasta el momento se han gestionado de manera manual, puede suponer una gran mejora en muchos aspectos para laboratorios y hospitales.

Para esta finalidad, se ha diseñado, desarrollado y fabricado un prototipo que contiene un robot de 3 ejes, una pinza adaptada y una controladora OpenSource con conectividad a una plataforma de comunicación con Smartphones.

A lo largo de todo el documento se encontrará todo lo necesario para entender y reproducir este prototipo.

Glossary

Word	Definition
Methacrylate	Also called polymethylmethacrylate or PMMA, according to its acronym in English, is a highly transparent thermoplastic, rigid, tough enough and easy to mold heat
Solidworks	Is a CAD (computer aided design) software for mechanical 3D modeling
Polystyrene	It is a thermoplastic polymer obtained by polymerization of styrene monomer, is transparent, rigid and brittle.
Agar	Growth medium used to culture microorganisms or small plants like the moss
Autoclave	Used to sterilize equipment and supplies by subjecting them to high-pressure saturated steam at
VGA Connector	A three-row 15-pin DE-15 connector
Servomotor	A rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration
PLA	Polylactide, a biodegradable thermoplastic aliphatic polyester derived from renewable resources
FDM	Fused deposition modeling, an additive manufacturing technology commonly used for modeling, prototyping, and production applications
FCT	Festo Configuration Tool, a software to configure the controllers of Festo
SD Card	A non-volatile memory card format
GPIO	General-purpose input/output , a generic pin on an integrated circuit or computer board
Electrovalve	A valve is controlled by an electric current through a solenoid
CSI-2 Bus	It defines an interface between a camera and a host processor
Lua	A lightweight multi-paradigm programming language designed primarily for embedded systems
Python	A widely used high-level, general-purpose, interpreted, dynamic programming language

HDMI	High-Definition Multimedia Interface is a proprietary audio/video interface for transferring uncompressed video data and compressed or uncompressed digital audio data
Protoboard	A construction base for prototyping of electronics
Optocoupler	A component that transfers electrical signals between two isolated circuits by using light, prevent high voltages from affecting the system receiving the signal

List of figures

List of tables

Table of contents

Chapter 1
Introduction

1.1 Motivation and objectives

Over the last years, there has been a breakthrough in the automation of certain processes for resource optimization. I'm not only talking about industrial or production processes, I'm also referring to health, pharmaceutic or even domestic staff. We've all seen the new pharmacies that dispense drugs automatically or robots that keep the floor clean day after day without any supervision. Well, as final project I wanted to join this new wave by proposing a new automation witch, in my opinion, can improve the current health system. And what exactly I want to improve? First of all the quality of the process, thanks to a docketed control and warning failure systems. I also intend to improve the life of the process user, due to time and effort savings. Finally, of course there will be an economic improvement for the employer, as in most automations.

Personally, I think it is a really thought out and worked final project, because I am very interested in biomedicine and in all its aspects. That is why I decided to focus my career on this direction and I hope that this project is only the beginning of many other useful and successful ones.

1.2 State of the art

Introducing Petri dishes

In any microbiology laboratory of the world we can find some small transparent circular boxes composed by two parts known as Petri dishes. Its name comes from a scientist who had this great idea and thank to that, thousands of scientists have been able to cultivate fungi, bacteria and all kinds of microorganisms under controlled conditions since then.

Julius Richard Petri was born on May 31, 1852, and although it did not achieve any Nobel Prize, he was a key scientist in the history of microbiology. After studying at the military academy Kaiser Wilhelm-Akademie, he doctorate at the Charité Clinic in Berlin. By 1876 he was already working for the famous Robert Koch, Nobel Prize in Physiology and Medicine in 1905, and it was while working with the famous German scientist when he invented his famous Petri dish.

He basically joined two round clear glass tops to allow samples to isolate and grow different microorganisms under controlled conditions. What seems simple and normal nowadays, in that moment was a real revolution for microbiology and medicine.

During the nineteenth century highly contagious epidemics were declining population of all over the world, and thanks to the Petri dishes, the scientist were able

to isolate microorganisms that caused diseases such as diphtheria or anger and to find the cure for them.

How Petri dishes are?

Petri dishes are shallow cylindrical glass or plastic plate with cover used by biologists and / or doctors.

There are several types, but they usually have rings or grooves in their caps and bases so that when they are stacked, they don't slip.

Those who are made of plastic, they are only for one use due to the possibility of contaminations. In contrast to that, glass petri dishes can be reused after sterilization (by autoclaving or dry heat during half an hour in a hot air oven at 121ºC for 15minutes).

Which is the main use of Petri dishes?

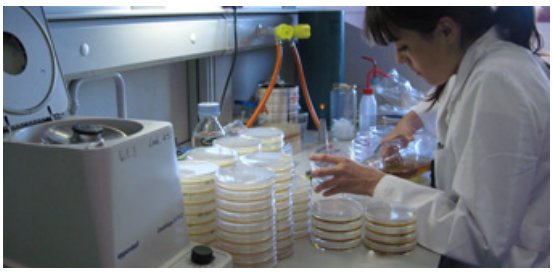
Primarly, they are often used to make agar plates for microbiology studies. The dish is partially filled with liquid containing hot agar and a mixture of specific ingredients which may include nutrients, blood, salts, carbohydrates, dyes, indicators, amino acids or antibiotics. Once the agar cools and solidifies, the plate is ready to be inoculated with a sample of a microorganism.

Then, they are incubated upside down to reduce the risk of contamination by airborne particles and to prevent accumulation of water condensation, otherwise it could disrupt or compromise a sample.

These particular dishes are used as well for eukaryotic cell culture in a liquid medium or on solid agar, to observe the plant germination, the behavior of very young animals or fluids drying in an oven, an everyday laboratory practice. Their transparency and flat profile are commonly used as a temporary recipient for display samples, especially liquids, with a low power microscope.

Current situation

Currently, the use of petri dishes do not have any kind of automation that supports these experimental processes.



1.2.1 Examples of experiments with Petri dishes

A. Experiment: Direct Contact

In this type of experiment, microorganisms are transferred directly to the prepared petri plate via direct contact. You can test the effectiveness of different soaps by treating different petri dishes with "dirty" hands before washing and "clean" hands after washing. Or, you can press a variety of common objects like coins, combs, etc. on different plates and compare the bacteria growth that results.

- What you need

Prepared petri plates with agar medium and nutrients.
Bacteria on hands, paws, etc.
Wax pencil for labeling dishes.
Masking tape.
Bleach.

- What to do.

Agar prepared petri dishes should be refrigerated until used and always stored upside down, this keeps condensation which forms in the lid from dropping onto and disrupting the bacteria growing surface.

When ready to use, let dishes come to room temperature before taking samples (about one hour).

Without tearing the agar surface, inoculate the dish by gently pressing fingers, finger nails, coin, etc onto agar surface. (Direct contact of lips or tongue is NOT a good idea.) Replace cover on dish, tape closed, and label each dish so you know the source of the bacteria. Store upside down.

Let grow in undisturbed warm location. Bacteria can grow at any temperature from about ambient room temperature (hopefully around 70°F) all the way up to about 100°F. ***Do not place in sunlight or on a heating register.***

You should see growth within a couple of days. The dishes will start to smell which means the bacteria are growing.

Make observations and keep records of what you see growing in each dish. Can you make any conclusions about what objects had the most bacteria?

Before disposing of dishes in the trash the bacteria should be destroyed. Pour a small amount of household bleach over the colonies while holding dish over sink. Caution - do not allow bleach to touch your skin, eyes or clothes. It will burn!

B. Experiment: Collected bacteria samples

Use a sterilized inoculating loop or sterile swabs to collect bacteria from different locations and then streak each petri dish with your sample. This involves a bit more technique than Experiment 1 but offers a wider choice of bacteria sampling locations. Swabs can be run over doorknobs, bathroom fixtures, animal mouths, etc.

- What you need.

Prepared petri dishes containing agar medium and nutrients.
Bacteria collected from doorknobs, bathroom fixtures, etc.
Wax pencil for labeling dishes.
Masking tape.
Sterile swabs or inoculating loop.
Alcohol burner (source of flame to sterilize inoculating loop).
Bleach.

- What to do.

Prepared petri dishes should be refrigerated until used and always stored upside down (i.e media in upper dish, cover on bottom). This keeps condensation which forms in the lid from dropping onto and disrupting the bacteria growing surface.

When ready to use, let dishes come to room temperature before taking samples (about one hour).

Collect bacteria from each location using one swab (or resterilized innoculating loop) for each new spot.

Inoculate each dish by streaking a pattern gently across the entire agar surface without tearing into it. Another common technique is to divide each plate into four quadrants by marking the lid with a cross. Streak your sample in straight lines starting in quadrant 1***. Generally, after a few days, quadrant one will show the most growth***. Depending on bacteria abundance on the swab, quadrant 4 may show no grow or only a few colonies. It is sometimes easier to distinguish different bacteria types in this low growth, less cluttered area.

Replace cover on dish, tape closed, and label each dish so you know the source of the bacteria. Store upside down.

Let grow in undisturbed warm location, ideally in an environment around 100° F (37° C) - not in sunlight or on a heating register.

You should see growth within a couple of days. The dishes will start to smell which means the bacteria are growing.

Make observations and keep records of what you see growing in each dish. Can you make any conclusions about what locations had the most bacteria?

C. Experiment: Testing the effectiveness of bacteria killing agents.

In this type of project, a petri dish is inoculated with bacteria then a paper disk (filter paper) treated with an antiseptic agent is placed in the dish***. After several days, a halo develops around the paper disk indicating a zone of no growth. Comparisons can be made between different antibacterial agents.***

- What you need.

Prepared petri dishes containing agar medium and nutrients.
Bacteria collected from doorknobs, bathroom fixtures, etc.
Wax pencil for labeling dishes.
Masking tape.
Sterile swabs or inoculating loop.
Alcohol burner (source of flame to sterilize inoculating loop).
Antibacterial agent (soaps, disinfectants, etc.).
Sterile water.
Test tubes, 12 x 75mm.
Filter paper or paper towel.
Small containers in which to soak paper disks.
Hole punch.
Tweezers.
Ruler.
Bleach.

- What to do.

Prepared petri dishes should be refrigerated until used and always stored upside down (i.e media in upper dish, cover on bottom). This keeps condensation which forms in the lid from dropping onto and disrupting the bacteria growing surface.

Prepare sterilized water by boiling water and letting cool to room temperature.

When ready to use, let petri dishes come to room temperature before taking samples (about one hour).

Prepare antiseptic disks by using a hole punch to create paper disks out of a piece of filter paper or paper towel. Soak one disk in each antibacterial agent to be tested.

Collect bacteria from each location using one swab for each new spot.

Fill a small test tube partly full of sterilized water. Dip bacteria laden swab into water.

This will transfer some of the bacteria you collected into the water. Now, inoculate a petri dish by pouring the water into the dish so the entire surface is covered. Pour out excess water. Repeat for each bacteria sample using fresh water and clean test tube each time.

Place a pretreated antiseptic disk in each inoculated petri dish.

Replace cover on dish, tape closed, store upside down. Be sure to label each petri dish with a name or number.

Let grow in undisturbed warm location, ideally in an environment around 100° F (37° C) - not in sunlight or on a heating register.

You should see growth within a couple of days. You should also see a "halo" around each disk indicating a no growth zone. Measure and compare the size of the kill zone to determine effectiveness of each antibacterial agent.

Before disposing of dishes in the trash the bacteria should be destroyed. Pour a small amount of household bleach over the colonies while holding dish over sink. Caution - do not allow bleach to touch your skin, eyes or clothes. It will burn!

Chapter 2
Proposal

Why we shoud automate Petri disches applications?

As you have read in section *1.2 State of the art*, experimental processes with Petri dishes are manual and they require a high workload by monitoring whether the plates are ready or not. That is because of the timing difference between experiments, the result or the objective of the process.

These experimental petri dishes processes require time-out from 1 to several weeks to perform actions on them.

The main idea is to support the management of experiments in petri dishes that are made in laboratories for biological and / or medical order to have greater control over experiments and less workload.

Until today, someone presence was required in the laboratory to perform and decide all actions associated with these kind of experiments, so our idea is to give a solution by turn the process more telematics and much more efficient.

Proposal?

Our proposal is to design an automated system (called *twentythree*), that it will combine a three-axis robot and an open hardware controller. It will solve the workload that requires control and timeouts decisions in most laboratories.

This design will be able to interact telematically with the operator without any person in the laboratory. To do so, there will be a camera where you can see the progress of the experiments.

2.1 Description

Before staging the description, you have to take into account that this is only a small-scale model that will manage Petri dishes, so their sizing and application types depend on the needs of the end operator and types of experiments. To describe this design it will be divided into different sections to make it more understandable:

Handling and structure

For the handling of Petri dishes, it will be used a two motion axes robot of the brand Festo. To that robot we will added a double-acting piston with a valve 5/2 in order to have a third axis of movement that will be the one to shift the Petri dishes.

We will design as well a clamp, which will be assembled in the third axis, to make the manipulator be able to adapt to the Petri dishes.

The structure of the manipulator is my own design and it will include several stations with different features and applications.

Functions and applications

The features and applications included in this project are the most common in petri dishes experiments. The motion sequences that have the robot will depend on the type of experiment you will use.

There will be three different stations, described as:

1. Interaction station
    a. Entrance of dishes
    b. Exit of dishes
    c. Camera

2. Post-culture station
    a. Storage at high temperature (Heater)
    b. Autoclaving
    c. Storage at low temperature (Refrigerator)

3. Pre-culture stations
    a. Waiting storage

Interactions stations will be located on the lower floor, and will house the entrance of the Petri dishes to a new process and the output of the plates once you have made the decision to remove the system boots. It will include a position where a camera to take an instant photo whenever you need, to verify the correct use of the

Petri dish, and also to monitor the processes before making a decision.

The ***post-culture stations*** will be located on the middle floor, and will include a high temperature storage, also a low temperature one, and finally an interaction point with the autoclaving to sterilize the dishes. These are the applications which are required once you have culture a Petri dish, then, they are absolutely necessaries.

The ***pre-culture station*** is the place where the agar dishes will be inserted before the process of the culture.

Communication and control

The control of the system will be taken by the open source microcontroller Raspberry Pi. It will be programed with Python and it will communicate with the robot using a cable with 15 pines VGA connector.

The communications with the system *twentythree* will be carried out by a messenger service application called Telegram. As you may know, this is a famous application that can be in any device such as smartphones, tables or a simple computer.

So, with that project you will be able to add this system in your mobile application, as you do it with a normal contact, and you will interact with it by commands. Of course the system will answer you anytime, and even it will send you the photos.

Commands

As I commented previously, the commands are used to interact with the system with the mobile application. Commands, explained below, should be written in the right moment, because not at any time they will be effective.

/ping	To check the connection with the robot
/start	To initialize and prepare the robot
/reboot	To reinitialize the system
/shutdown	To shutdown the system
/in	To include a dish in the system
/preculture	To keep the dish before the culture
/postculture	To make an action after the culture
/savehot	To keep the dish in the high temperature zone
/savecold	To keep the dish in the low temperature zone
/autoclave	To put the dish in the interaction point with the autoclaving
/petrinfo	To make an action with some dishes inside the system
/petrihot	To select the dish which is in the high temperature zone
/petricold	To select the dish which is in the low temperature zone
/petripre	To select the dish which is in the pre-culture station
/1 or /2 or /3	To select the preculture spot in function if is empty or not
/takepic	To take a photo and send it
/out	To take out of the system a dish

2.2 Operation Diagram

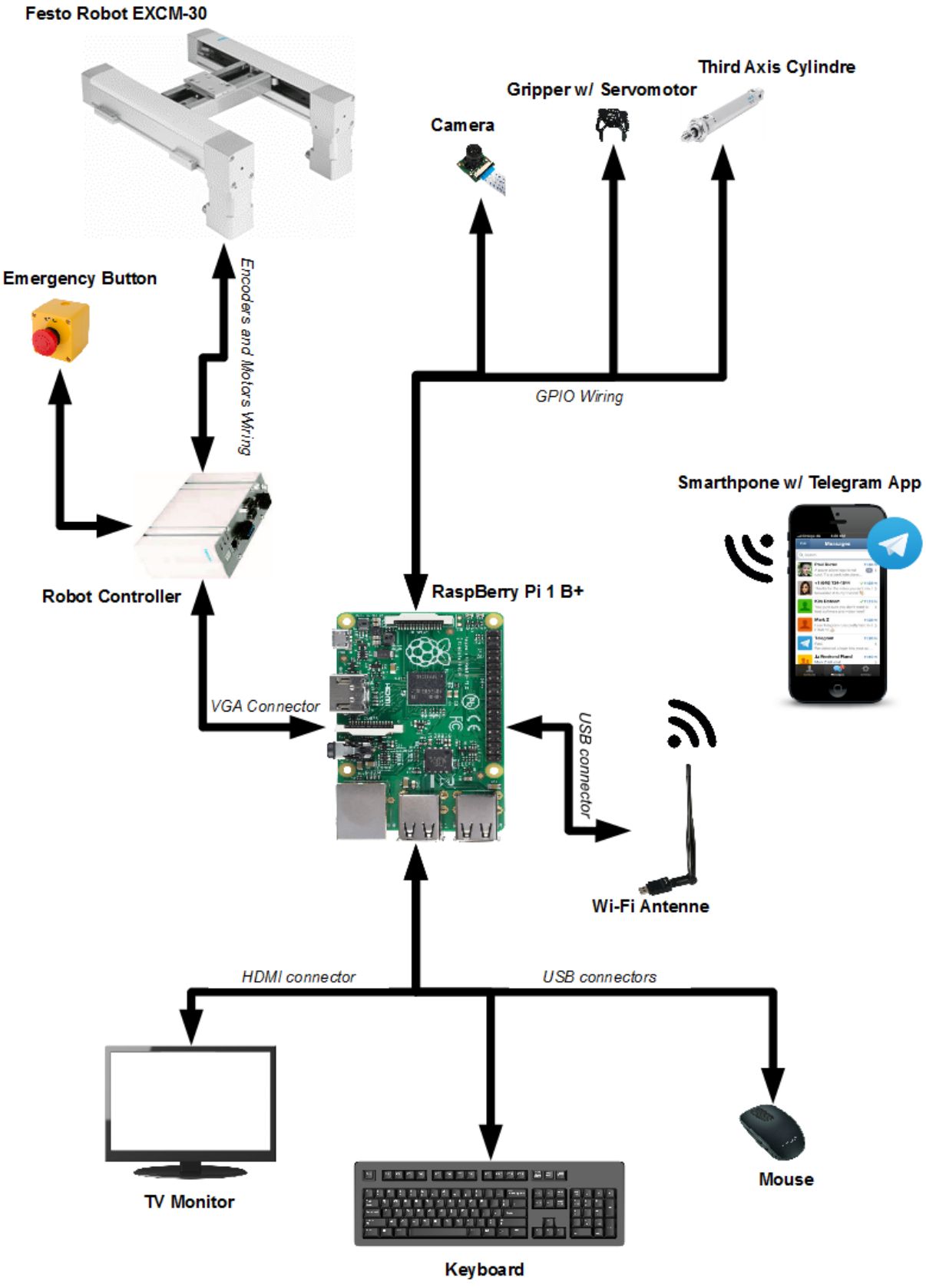


Figure 1: Operation Diagram

Chapter 3
Design proposal

3.1 Robot EXCM-30

What is the EXCM-30 robot?

The EXCM-30 robot is one of the manipulators which is offered by the automation products company called Festo.

Festo is company impulse by the engineering, production and sales of electrical and air products, to control and drive technology or industrial process automation. It was founded in 1925 by Albert Fezer and Gottlieb Stoll. Initially, the company manufactured wood cutting tools and later it diversified into the automation industry.

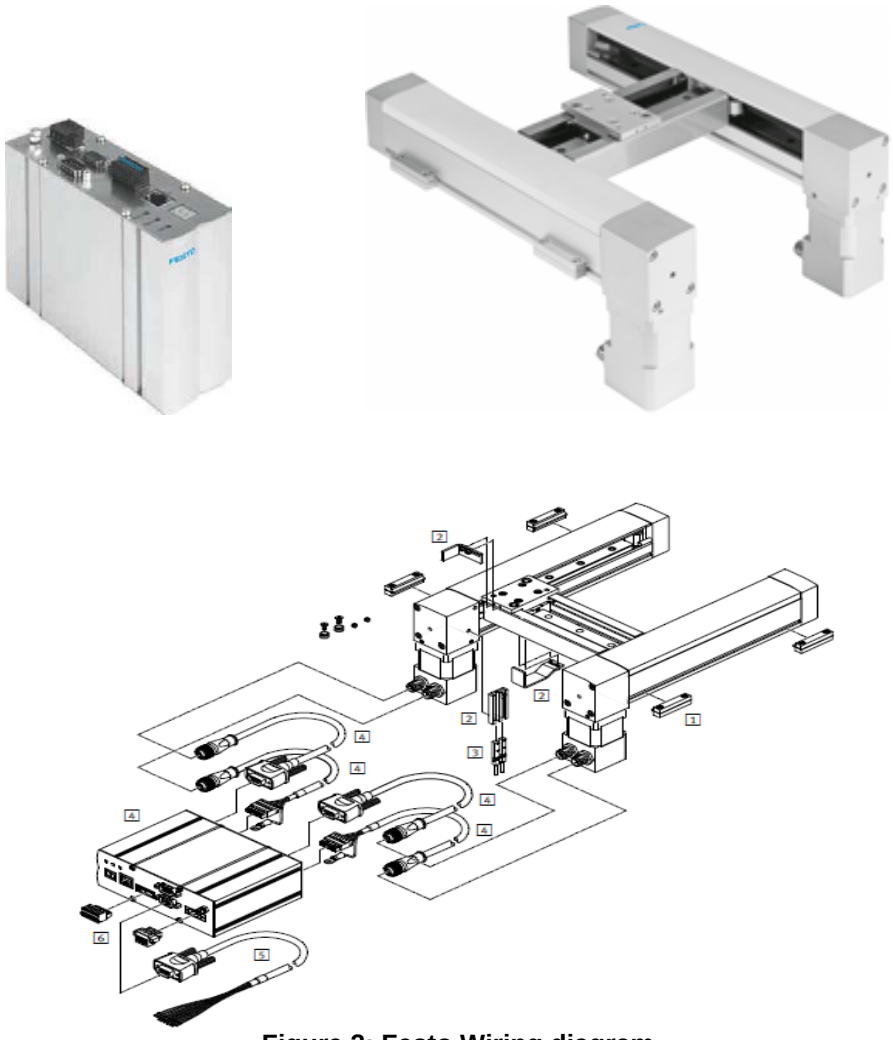


Figure 2: Festo Wiring diagram

Description

The ideal solution if every millimeter is crucial: the compact gantry EXCM. Great functionality combined with extremely compact design and maximum coverage workspace. The drive system of parallel kinematics has minimal moving masses and it is delivered configured for an easy commissioning.

This type of robot is designed for desktop applications in the field of assembly and handling small parts, or in automated laboratory processes. The ball bearing guide is suitable for heavy loads.

Operation

The dentate belt moves the carriage in two dimensions (X and Y). The drive system with two stationary motors; regulated operation positions (closed loop). The motors are coupled to the toothed belt. The belt is guided by reversing rollers:

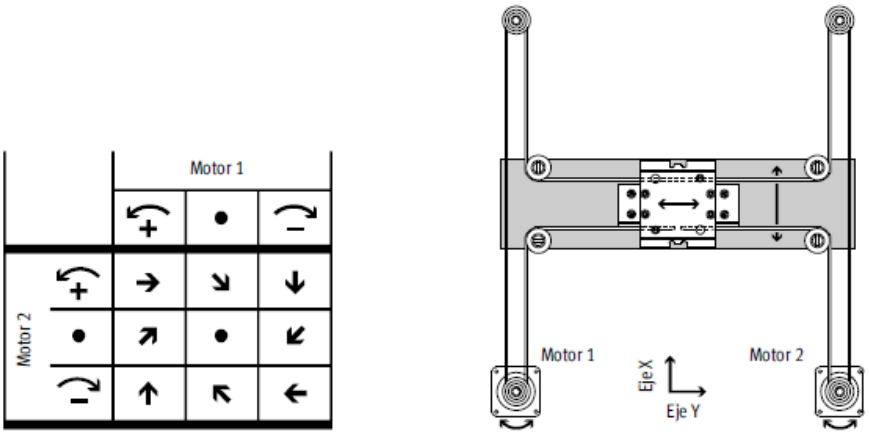


Figure 3: Festo Motors function

By the correct operation of the motors, the car can move to any position in space.

Drive unit and controller

Standardization: a set of drive unit and controller class IP20 for Festo plug and work. By encoder, performance servo driven closed circuit.

Communication

Extreme versatility: inputs / outputs for advancing up to 64 positions. CANopen and Ethernet for maximum freedom of movement and up to 250 positions

Applications

Food and screwing small pieces, placing dots of glue, electronic evidence: advance to contact points, tests, versatile positioning of parts and components in assembly processes, tasks palletizing and unloading pallets, manufacturing / assembly desktop.

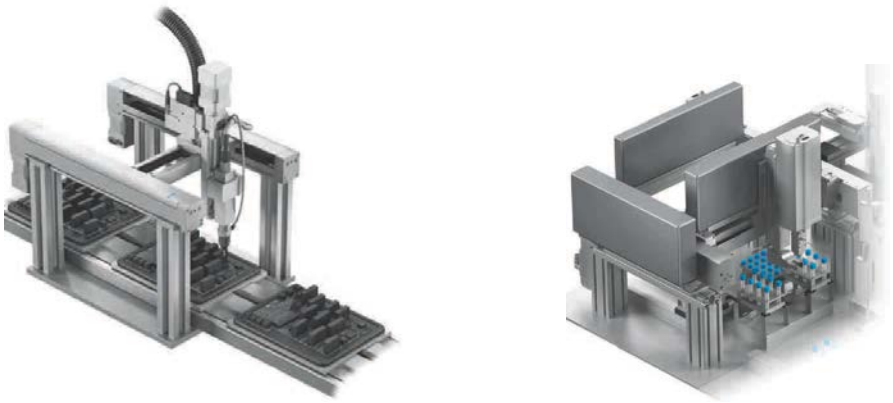


Figure 4: Festo aplications

Technical specifications

		EXCM-10	EXCM-30
Carrera eje X	[mm]	150, 260, 300, 360, 460, 700	Estándar: 100, 150, 200, 300, 400, 500 Sobre demanda: 90 ... 700
Carrera eje Y	[mm]	110	110, 160, 210, 260, 310, 360
Carga útil máx.	[kg]	0,5	3
Velocidad máx.	[m/s]	0,3	0,5 (opcional con servo AC: 1)
Aceleración máx.	[m/s^2]	3	10
Precisión de repetición	[mm]	± 0,1	± 0,05
Precisión de posicionamiento	[mm]	± 0,5 absoluta	± 0,5 absoluta

Table 1: Technical specifications of EXCM-30

3.1.1 Structure

3.1.1.1 Sketch

About the structure, it has been designed with a perpendicular gantry to the support base and this way it can make the operation at stations.

The design has been based on a structure similar to the following:

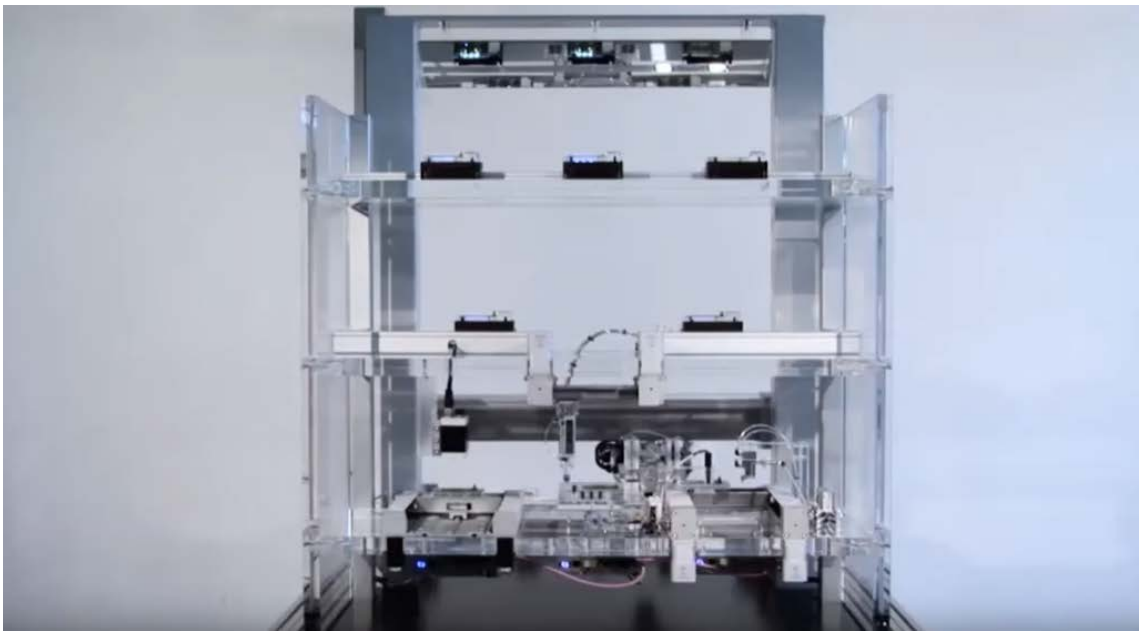


Figure 5: Structure Sketch

It will be composed by three stations:

Station 1: it will be the lowest season, and it will house the entrance of samples in Petri plates on the left side and the output on the right side. In the entrance of the dishes there will be installed the camera. In the exit of the dishes there will be a stop sensor.

Station 2: on the left side we will find the storage space for high temperature (heater), on the central position there will be the autoclave space, and finally in the right side we will find the storage space for low temperature (refrigerator)

Station 3: this is the top one and in our case it will have three storage spaces with a stop sensor for Petri dishes in each space.

3.1.1.2 Design

It has been designed by SolidWorks, and the chosen material has been the methacrylate due to its right properties for the prototype.

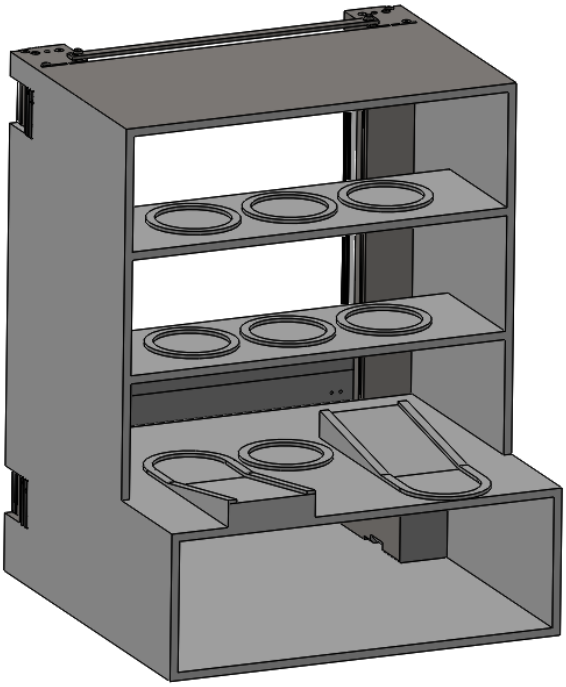


Figure 6: Design Front view

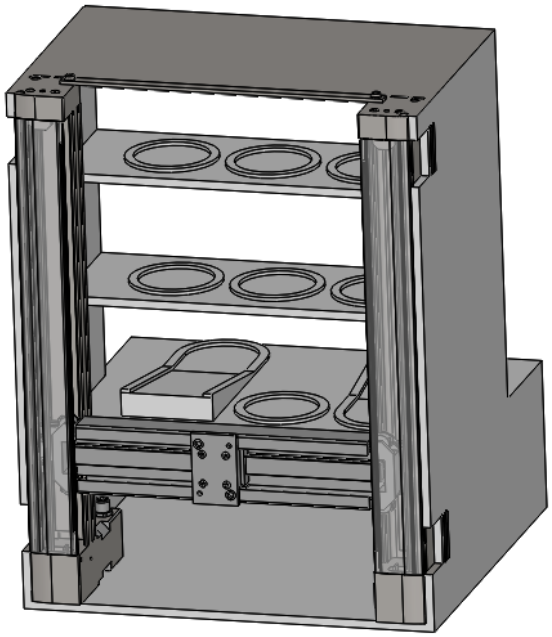


Figure 7: Design Rear view

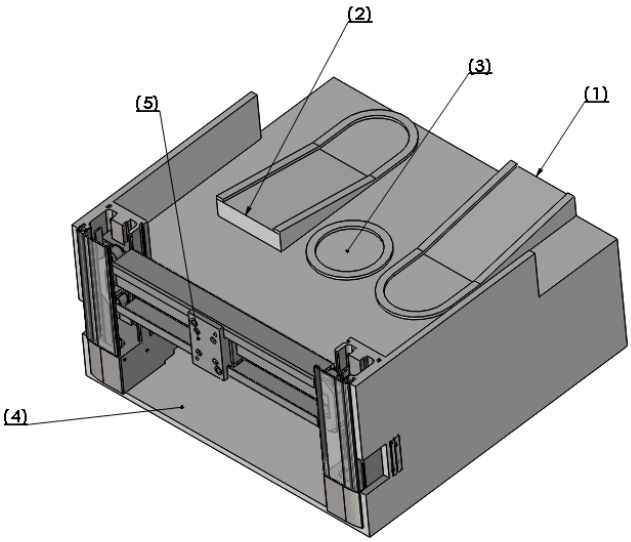


Figure 9: Design First flow view

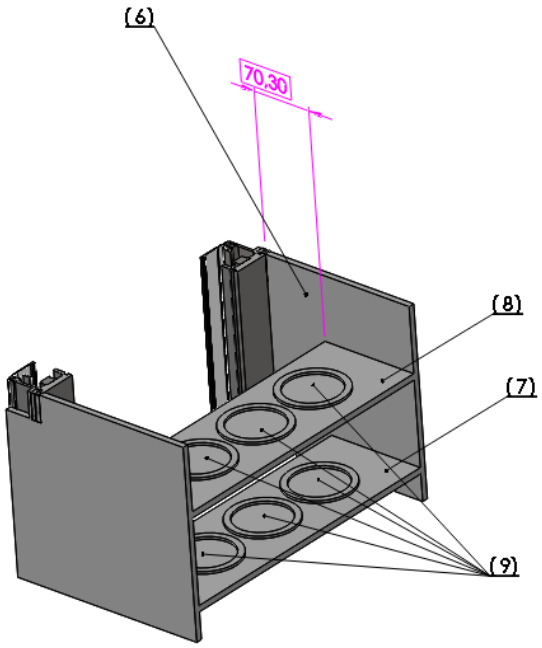


Figure 8: Design storage floor view

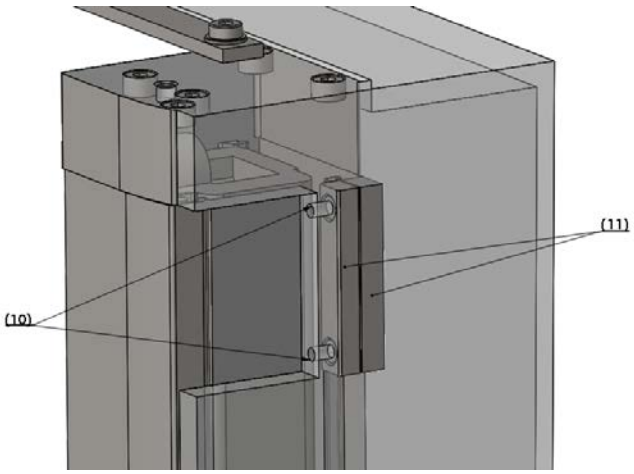


Figure 10: Design Mooring's details

Descriptions

(1) Destined to the entrance of dishes, with a descent ramp to inside and lateral guides.

(2) Destined to the exit of dishes, with a descent ramp to outside and lateral guides.

(3) This position is designed for Petri dishes to be photographed, so a camera will be installed in the superior part.

(4) Inferior floor, there will be all the installation of the robot including the robot control and the Raspberry with its connections.

(5) It shows the moorings where there will be installed the third axis.
(6) Crossing zone and clamp manipulation.
(7) Intermediate floor
(8) Superior floor
(9) Petri dishes positions with a little barrier to impede the movement.
(10) Metric pins 3 located at four structure moorings to hold the robot.
(11) Support that includes the robot.

3.1.2 Third axis

3.1.2.1 Election

For the third axis of the robot, it will be used a doble effect cylinder of the brand Festo, and there's a model in the following figure.

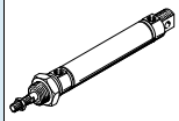
Funciona-miento	Ejecución	Diámetro del émbolo [mm]	Carrera [mm]	Carrera específica[1)] [mm]	Vástago					
					Doble S2	Prolon-gado K8	Rosca exterior			Rosca interior K3
							Prolon-gado K2	Corta K6	Especial K5	
Doble efecto	DSN-... – Sin detección de posiciones									
		8, 10	10, 25, 40, 50, 80, 100, 125, 160, 200, 250, 300, 320, 400, 500	1 ... 100	–	–	–	–	–	–
		12, 16		1 ... 200						
		20		1 ... 320						
		25		1 ... 500						

Table 2: Cylinders characteristics

The configuration of the cylinder is:

- Piston diameter: 12mm
- Function: Double effect
- Run: 80mm
- Attenuation: Elastic tops

In this case, the exact model would be: normalized cylinder DSN-12-100-P

3.1.2.2 Assembly Design

For the assembly of the third axis, first of all we have to desing a mooring for the gripper designed in *section 3.1.3*.

As you can see in the following figure, the basal part of the clamp has a perforation which is adapted to the piston for a correct subjection with a metric nut 12.

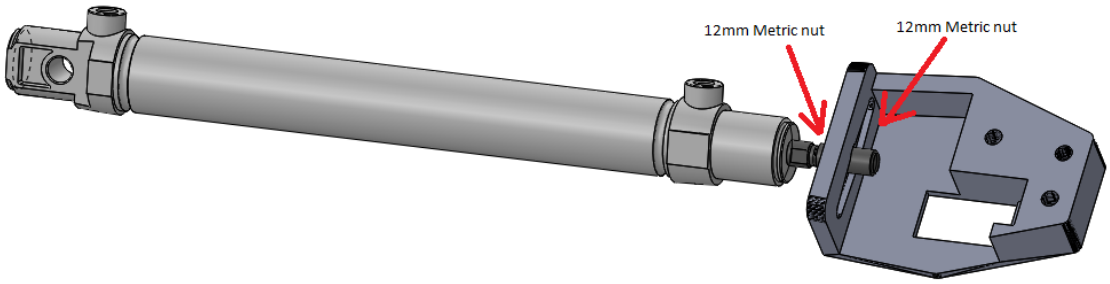


Figure 11: Assembly Cylinder-Gripper

This combination has to be assembled to the manipulator plate designed by Festo, as you can see below.

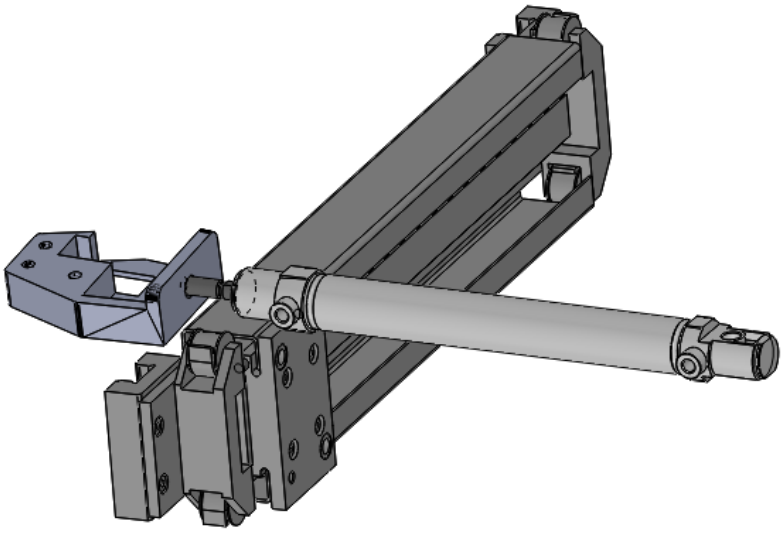


Figure 12: Assembly Piston-Robot

A new piece will have to be designed assembled to the movement plate base from Robot to hold the piston.

But is here where I have an impediment. The design of the piston is cylindrical, and has been impossible for me to design a piece for assembling this piston with the robot base.

So, I have studied the case, I think one solution would be to change the piston for another model with different design.

3.1.2.3 Alternative Assembly Piston-Robot

The alternative is to use the model ADN-12-80-A-P-A also from Festo

Función	Ejecución	Tipo	Diámetro del émbolo	Carrera		Detección de posiciones	Amortiguación	
							Fija	Auto-rregulable
			[mm]	[mm]		A	P	PPS
Doble efecto	Tipo básico							
		ADN	12	5, 10, 15, 20, 25, 30, 40	1 ... 300			
			16	5, 10, 15, 20, 25, 30, 40, 50	1 ... 300			
			20, 25	5, 10, 15, 20, 25, 30, 40, 50, 60	1 ... 300			■
			32, 40, 50	5, 10, 15, 20, 25, 30, 40, 50, 60, 80	1 ... 400	■	■	Ø 20 ... 100
			63	10, 15, 20, 25, 30, 40, 50, 60, 80	1 ... 400			
			80, 100	10, 15, 20, 25, 30, 40, 50, 60, 80	1 ... 500			
			125	-	1 ... 500			

Tabla 3: Alternative piston characteristics

With the same configuration like the first election.

The configuration of the cylinder is:

- Piston diameter: 12mm
- Function: Double effect
- Run: 80mm
- Attenuation: Elastic tops

This model is cube-shaped and it has anchoring slots that facilitates to design a piece that fits.

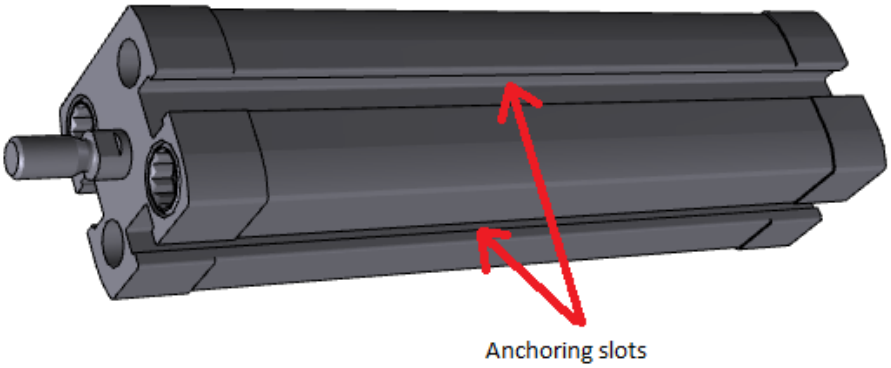


Figure 13: Alternative Piston

3.1.3 Gripper terminal

The terminal gripper has been designed based on the following characteristics:

- Petri dishes manipulation 2901 (standard size : 60mm x 15mm)
- Gripper opening by servomotor 9G
- Gripper base with the possibility to fix the third axis

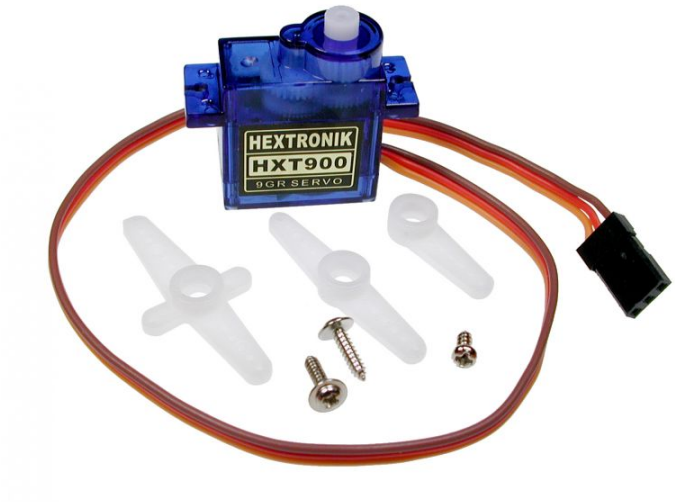
Petri plates talked in *section 2.1* have the following form:

This is the standard 2901, it would be made of polystyrene (PS) in our case.

Figure 14: Petri Dish Standard 2901

3.1.3.1 Servomotor 9g

The gripper opening has been designed to be directed by a servomotor 9G, which achieves all the characteristics required to move the grippers and the dishes in the robot movements.



Velocity: 0.10 sec/60° @ 4.8V

Torque: 1.8 Kg-cm @ 4.8V

Function voltage: 3.0-7.2V

Function temperature: -30℃ ~ 60 ℃

Rotation angle: 180°

Pulse wide: 500-2400 µs

Cable length: 24.5cm

Figure 15: Servomotor 9g

Its wiring connections are simple, although as this servomotor will be in the third axis group, there will have to enlarge the wiring to stock up on all robot movements.

The red cable would be the supply (between 3and 7,2volts), the brown one would be the negative and the orange one would be the binary pulse sent by the control raspberry, as you can see below.

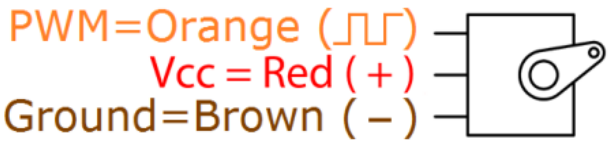


Figure 16: Servomotor wiring

The servomotor will move lineally clockwise or reversed depending on the type of pulse sent out by the control raspberry. In the figure you can see an example of how the movements will be:

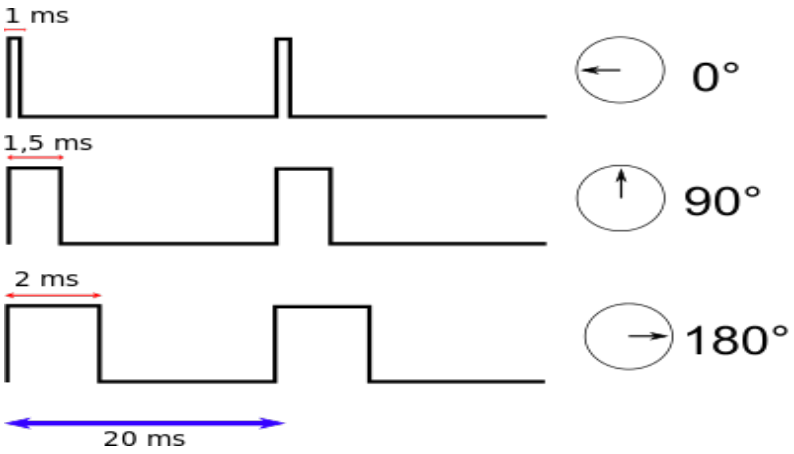


Figure 17: Servomotor Signal function

3.1.3.2 Design

It has been designed in Solidworks 2016 so both arms of the gripper can catch correctly a normal Petri dish (circumference diameter of 600mm).

In the following figure, we can see all the pieces assembled.

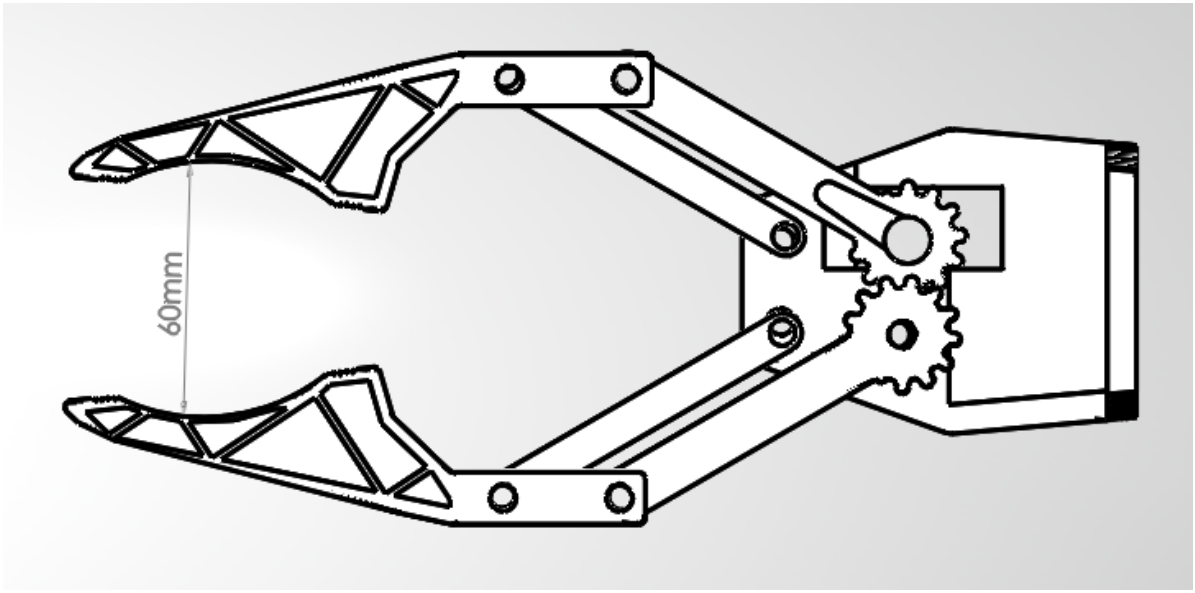


Figure 18: Gripper design

The design consist of:

- Two claws joined to two connectors with ceiling pins.

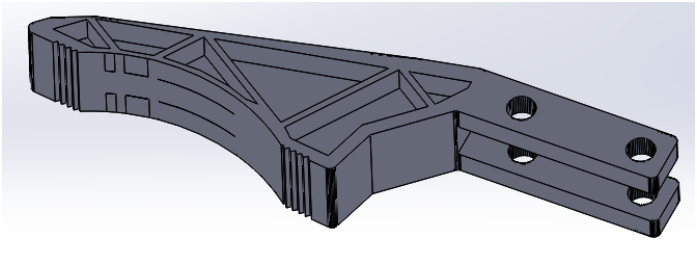


Figure 19: Claw of Gripper

- Two ceiling pins to make the move stronger.

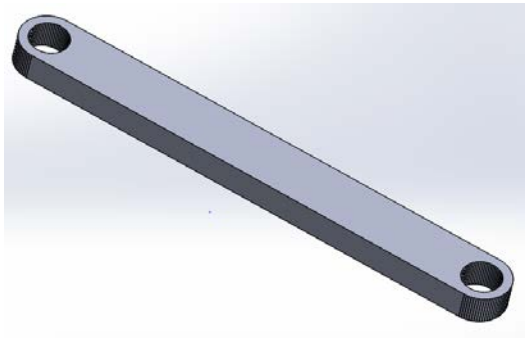


Figure 20: Ceiling pins

- A ceiling connector with a round gear.

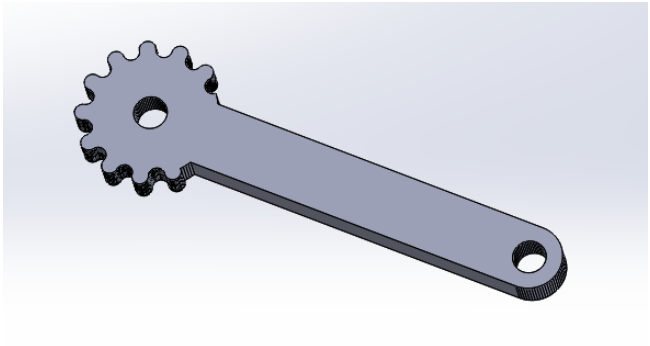


Figure 21: Ceiling pin with gear

- A ceiling conector with a round gear and the join helix piece of the servomotor.

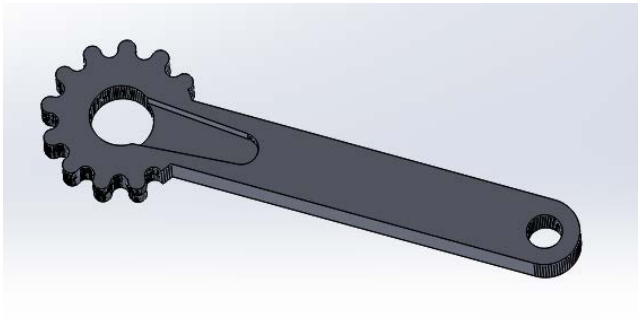


Figure 22: Ceiling Pin for Servomotor 9g

- A basal piece where will be fited the servomotor, with pins for the ceiling conectors and a terminal pin to join together with the third axis.

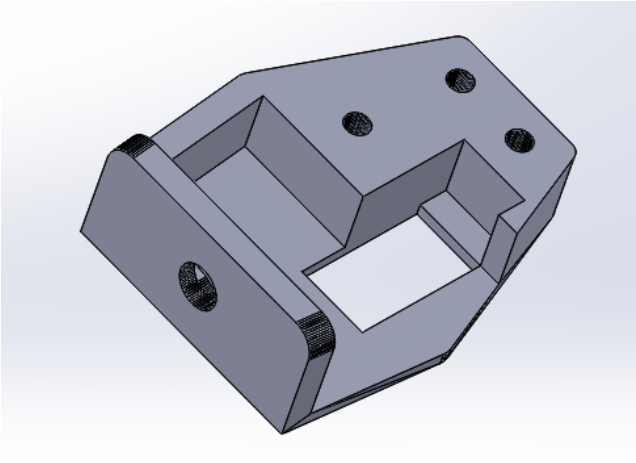


Figure 23: Main piece of the gripper

3.1.3.3 Construction

For the construction of the gripper, it will be used a 3D printer and it will be printed in PLA with FDM technology.

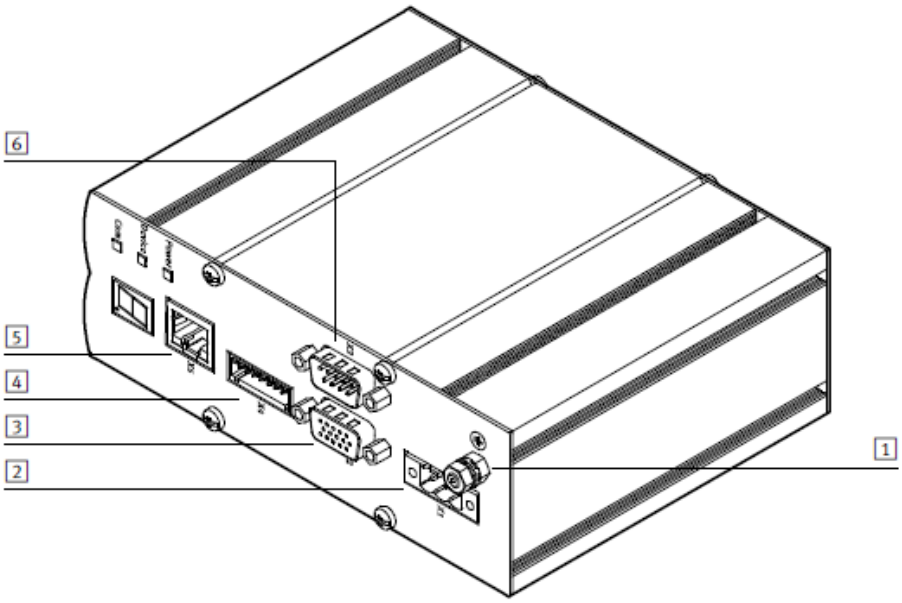
It will be used all the FabLab services in Terrassa.
This way, the production cost of the gripper will reduce.

FABLAB TERRASSA

3.1.4 Controller

The H-gantry EXCM-30 includes a control to manage depending on the E/S interface of the motors and encoders of the robot.

3.1.4.1 Front part of the controller



1 Tierra funcional
2 Fuente de alimentación [X1]
3 Interfaz I/O [X2]
4 Interfaz de parada de emergencia [X4]
5 Interfaz de Ethernet [X5]
6 Interfaz CANopen [X3]

(2) Power source

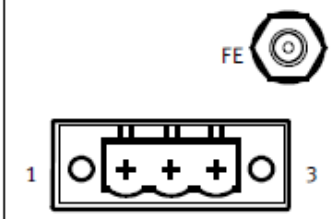
Conexión	Pin	Función		
	FE	Tierra funcional		
	1	Tensión de la lógica	+24 V (±15 %)	Alimentación de la electrónica de mando
	2	Tensión de la carga	+24 V (±15 %)	Alimentación del paso de salida de potencia y del motor
	3	Potencial de referencia	0 V	Potencial de referencia para tensión de la carga, tensión de la lógica e interfaz de control

Table 4: Wiring of Robot Power Source

It will be used the 24v and 5amperes Siemens 6ES7307-1EA01-0AA0.

(3) Interface I/O

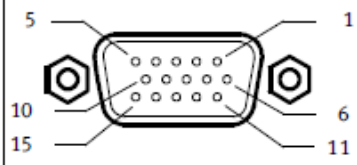
Conexión	Pin	Función	
	1	24VL	Salida: listo para la comunicación
	2	DI 1	Entradas: selección de frase
	3	DI 2	
	4	DI 3	
	5	DI 4	
	6	DI 5	
	7	6 DI	No se utiliza
	8	START	Entrada: inicio de frase
	9	ENABLE	Entrada: desbloquear regulador
	10	RESET	Entrada: validar fallo
	11	Enabled	Salida: desbloquear regulador
	12	FAULT	Salida: fallo
	13	Ack	Salida: validación
	14	MC	Salida: Motion Complete
	15	0 V	Potencial de referencia

Table 5: Wiring of Robot I/O

It has been designed a male connection with PCB connectors for the interconnection of the I/O interface with Raspberry Pi.

Figure 24: VGA Connector

(4) Emergency Stop interface

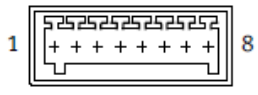
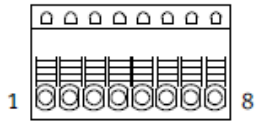
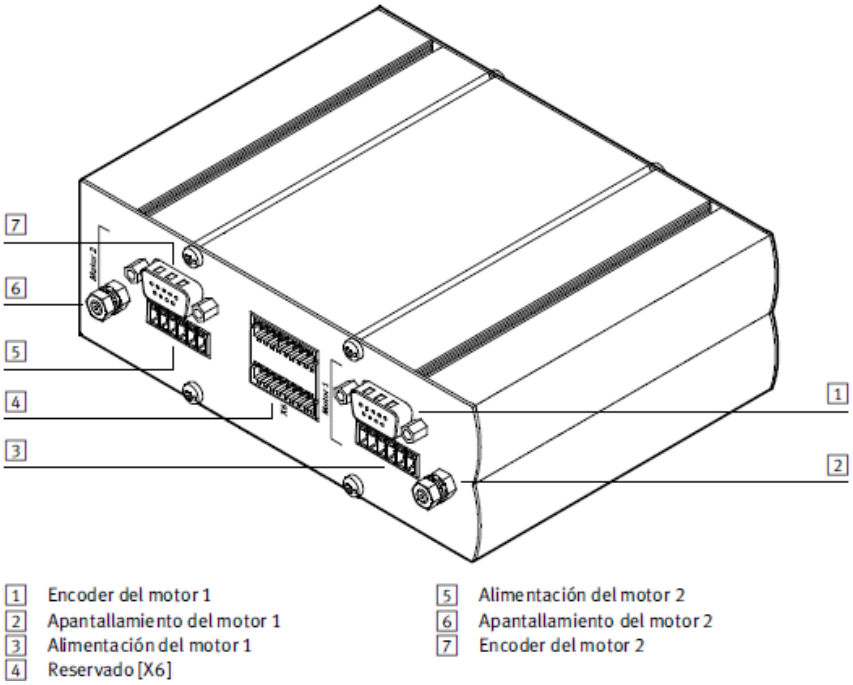
Conexión	Pin	Función	
Interfaz en el controlador  Conector lado de conexión 	1	+24 V lógica	Salida: tensión de la lógica +24 V
	2	TO	Entrada: interrumpir la tensión de alimentación de los motores (con 0 V)
	3	ES[1)]	Entrada: activar rampa de frenado (con 0 V)
	4	RB	Entrada: soltar freno (con +24 V)
	5	FAULT[2)]	Salida: hay un fallo (con +24 V)
	6	DIAG1 (Contacto 1)	Los contactos de diagnosis están libres de potencial. El contacto de diagnosis es de baja impedancia cuando la alimentación del excitador está desconectada (contactos de diagnosis 1 y 2 puenteados).
	7	DIAG2 (Contacto 2)	
	8	0 V (GND)	Potencial de referencia

Table 6: Wiring of Emergency Stop interface

Pin 1, pin 2 and pin 3 will be the only ones used for the manipulator security.

3.1.4.2 Back part of the control

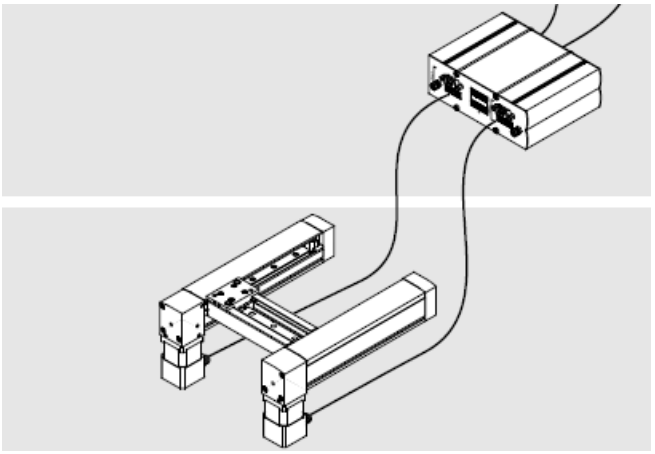


1 Encoder del motor 1
2 Apantallamiento del motor 1
3 Alimentación del motor 1
4 Reservado [X6]
5 Alimentación del motor 2
6 Apantallamiento del motor 2
7 Encoder del motor 2

This wiring connection will be directly with de robot:

(1)(2)(3) Is the wiring connection of motor 1

(5)(6)(7) Is the wiring connection of motor 2



Motor connector functioning

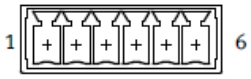
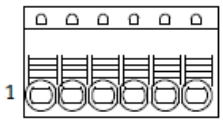
Conexión[1)]	Pin	Función	
Interfaz en el controlador  Conector lado de conexión 	1	Ramal A	Conexión de ambos ramales del motor
	2	Ramal A/	
	3	Ramal B	
	4	Ramal B/	
	5	Br+	Conexión del freno de sostenimiento. Resistente a cortocircuitos y sobrecargas. BR– = GND, BR+ se conecta (24 V de carga)
	6	BR–	

Table 7: Function of motor connector

Encoder conection functioning

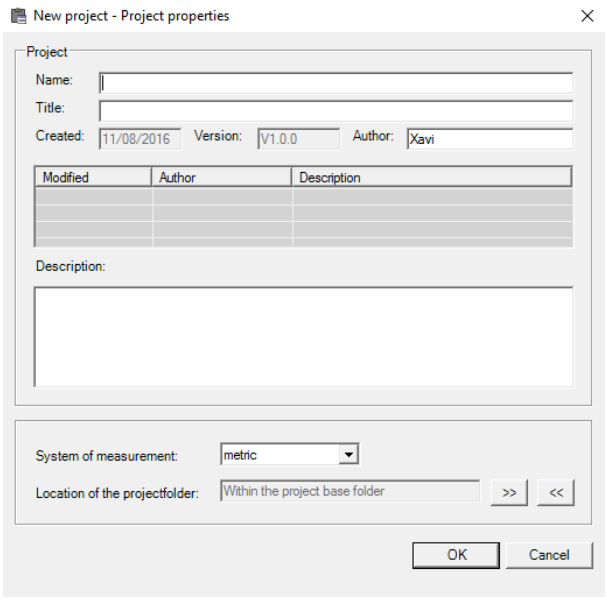
Conexión	Pin	Función	
	1	A[1)]	Señal de encoder incremental A+, polaridad positiva
	2	B[1)]	Señal de encoder incremental B+, polaridad positiva
	3	N[1)]	Señal de encoder incremental impulso cero, polaridad positiva
	4	0 V	Potencial de referencia
	5	+5 V ±10 %	Alimentación del transmisor. Máx. 100 mA, SIN protección ante cortocircuitos.
	6	A/[1)]	Señal de encoder incremental A–, polaridad negativa
	7	B/[1)]	Señal de encoder incremental B–, polaridad negativa
	8	N/[1)]	Señal de encoder incremental impulso cero, polaridad negativa
	9	–	–

Table 8: Function of encoder conection

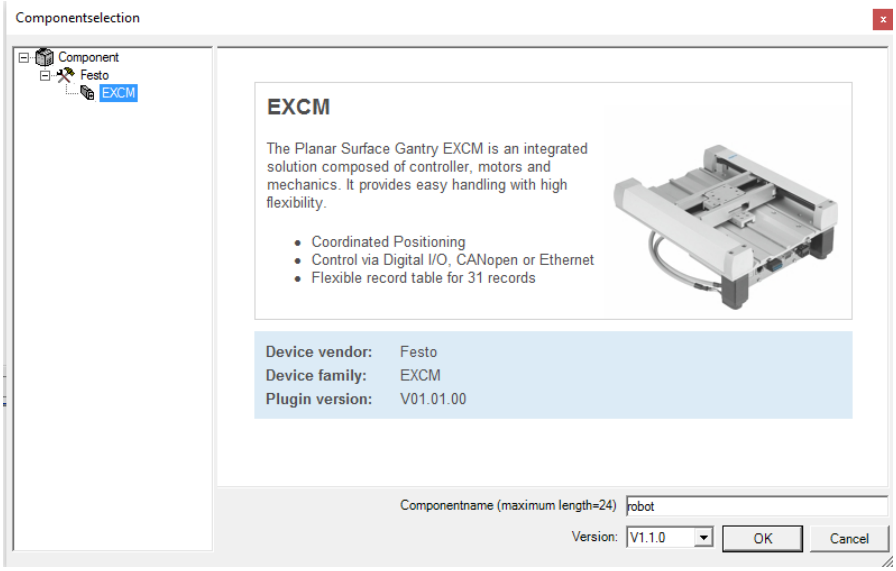
3.1.4.3 Controller configuration

The control will be configurated by the software provided by Festo for its control "*Festo Configuration Tool".*

In the beginning, we will have to creat a new project indicating the name, title and author. We will have to indicate, as well, the mesurement system, which will be *metric* in our case, then push OK.



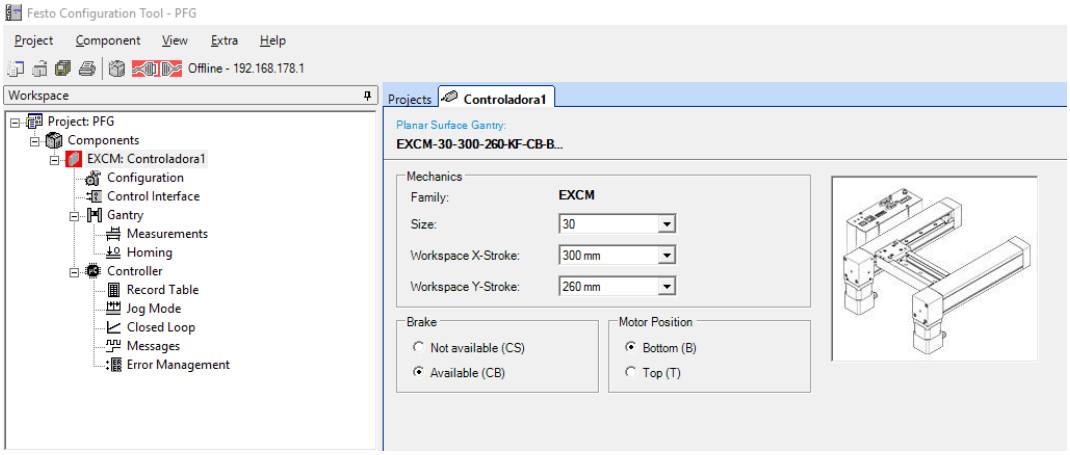
A new window will open so we will have to choose the components of the brand Festo that we want to control. In our case, we will select the EXCM model and we will indicate component name, then push OK.



All the configurable panels will charge, and we will have to go step-by-step to configure each one of them. In the following figure you can see the Configuration panel where you will have to introduce our own configuration:

- *Size:* robot size
- *Workspace X*: workspace in the X axis
- *Workspace Y*: workspace in the Y axis
- *Brake*: we will indicate if motors have brakes or not

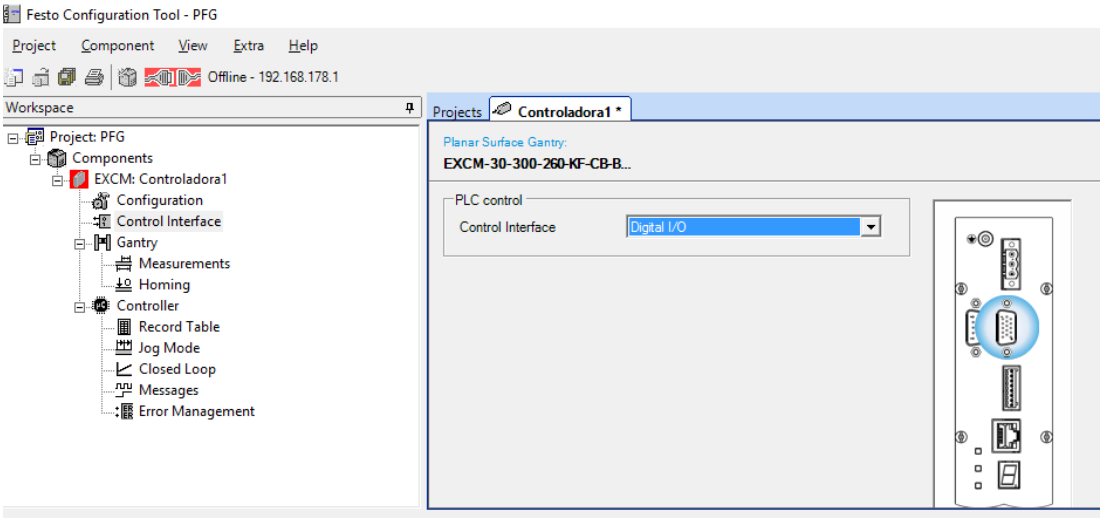
- Motor position: we will indicate the motor position. To help, there will be an image where you can see the configuration by selecting Bottom or Top.



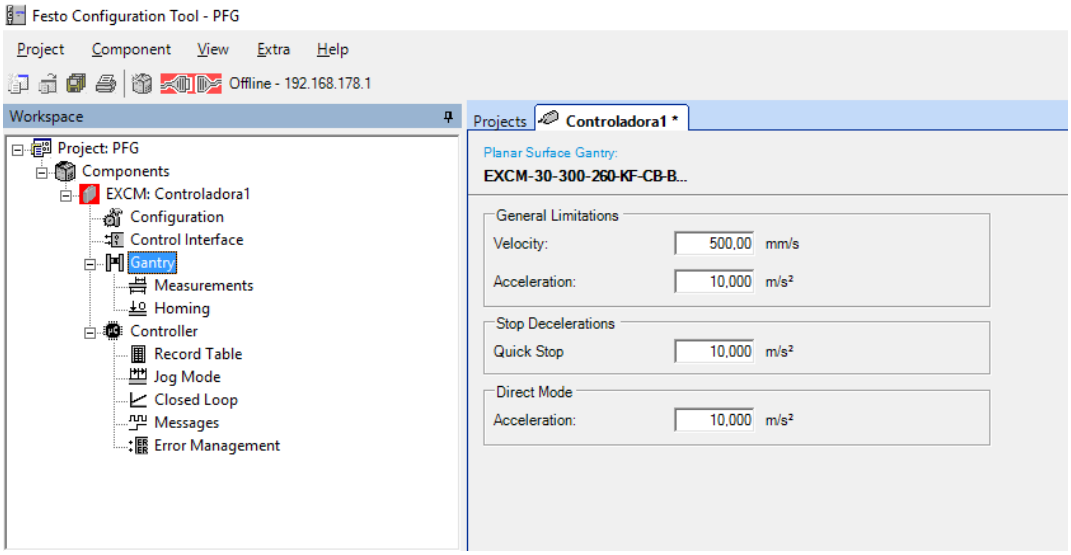
In the tab **Control Interface** we will have to indicate which kind of interface we want to control our robot from PLC or similar.

Digital I/O: control for entrances and exits

- CanBus: control well-known as master-slave canbus
- Ethernet: control by ethernet phrases

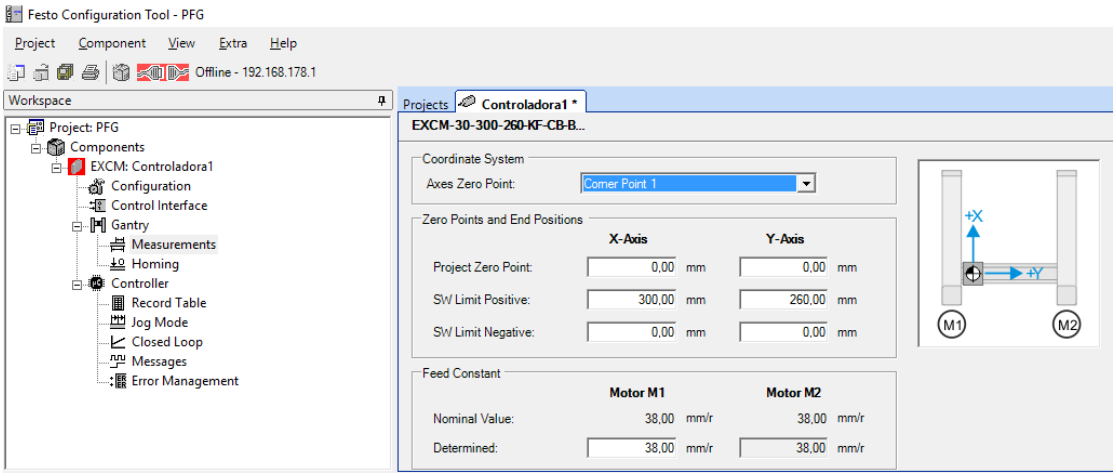


In **Gantry** we will have to introduce all the velocity limitations and acceleration which for security reasons we have to take. It is recommend to let the ones which are by default, unless you would want to modify for any special application.

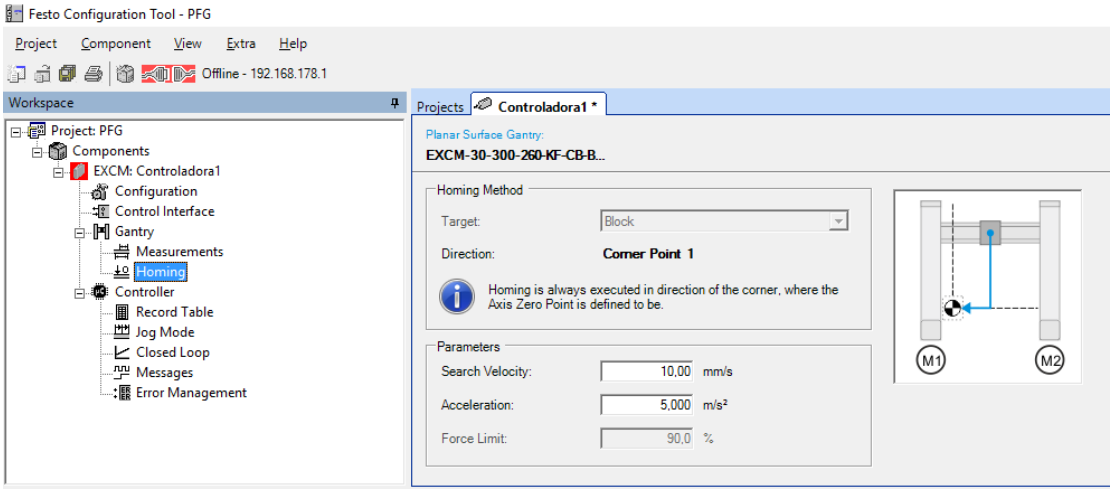


The tab **Measurements** is pretty important by the time you want to fix the movement standings and fix the zero of the robot. There is a chapter for the advance constants of the motor, as well.

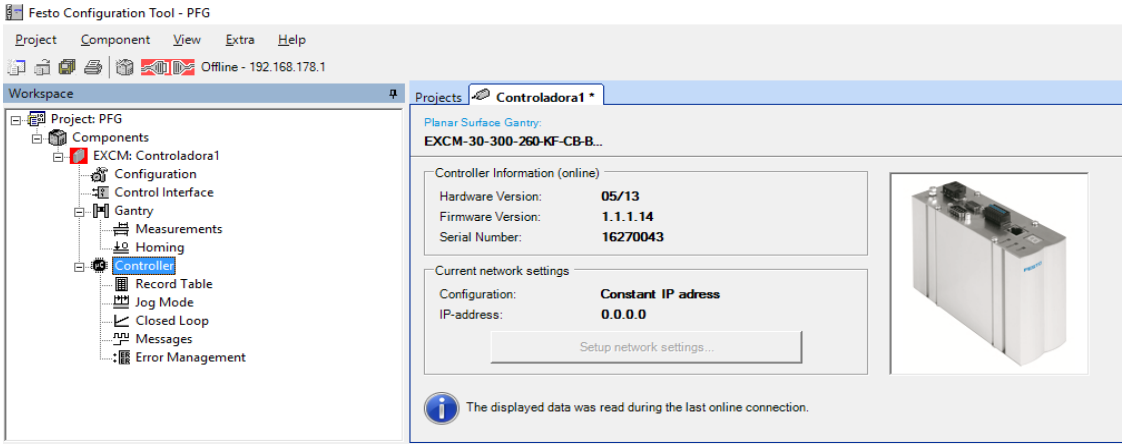
- Axes Zero Point: we will indicate which corner we want to be the zero of the robot.
- Project Zero Point: we will indicate if we want to increase the distance in X or Y to the zero.
- SW Limit Positive: we will indicate the distance limit in positive that we want to move from zero in X or in Y.
- SW Limit Negative: we will indicate the distance limit in negative that we want to move from zero in X or in Y.
- Feed Constant Determinated: the advance constant of the motor. In this case we will let that at 38mm per revolution, that would be its nominal value (recommended).



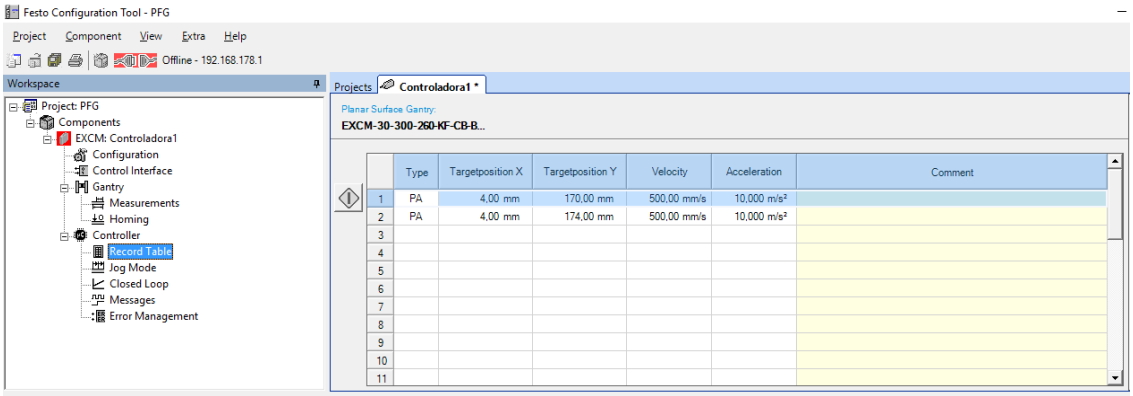
The **Homing** refers to that time which the robot loses its position value due to a new begining, position error or problem with the electri supply. In these cases, the robot has to be in position zero, so we will have to configure its velocity and acceleration for this action.



In this **Controller** panel we can see all the data about the versions, serie number and what Ethernet configuration we have to use to communicate with her.



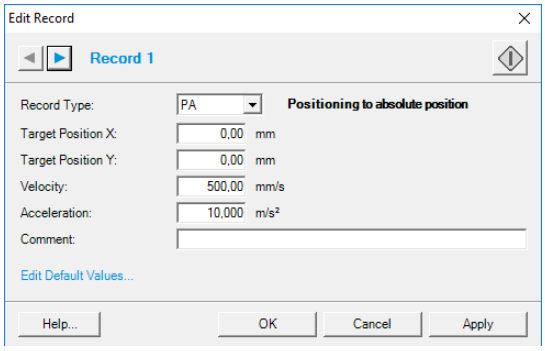
In **Record Table** we can include until 31 posible points of position memory that we could need to do programated movements by the entrance/exit interfaces.

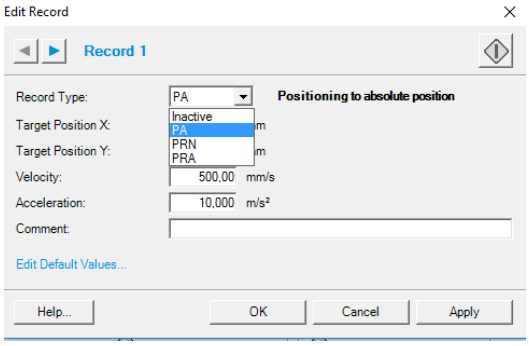


To include a memory position, we only would have to push the left number and it will open a new configuration tab for the position, as the following one:

- Record Type: here we can select the type of move we want
  - PA: Absolute, it is added to the distance from zero (recommended)

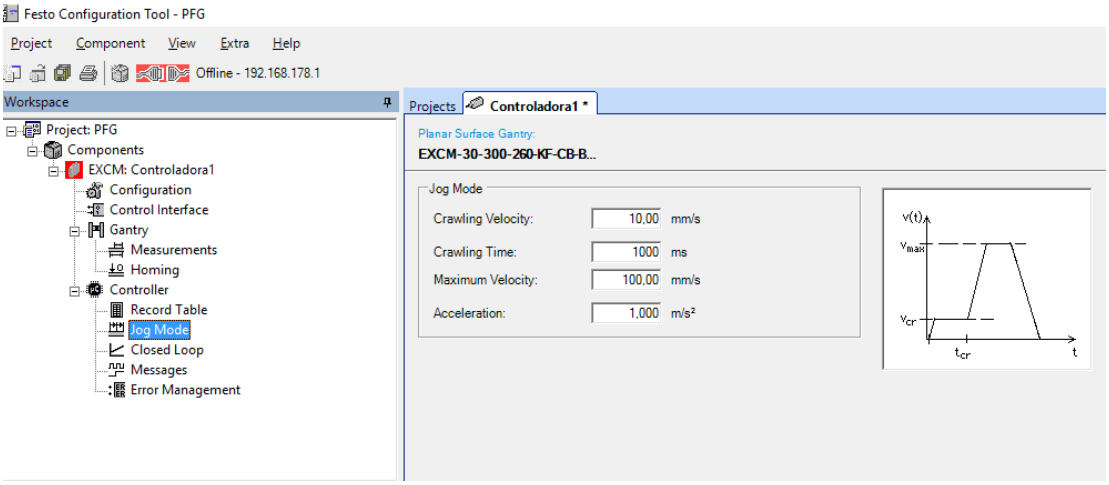
  - PRN: Nominal relative, it is added to the actual distance
  - PRA: Absolute relative , it its added to the absolute distance
- Target Position: the distance we want to add to X and/or Y
- Velocity: the velocity we want to use to go to the added distance
- Acceleration: the acceleration we want to use to go to the added distance



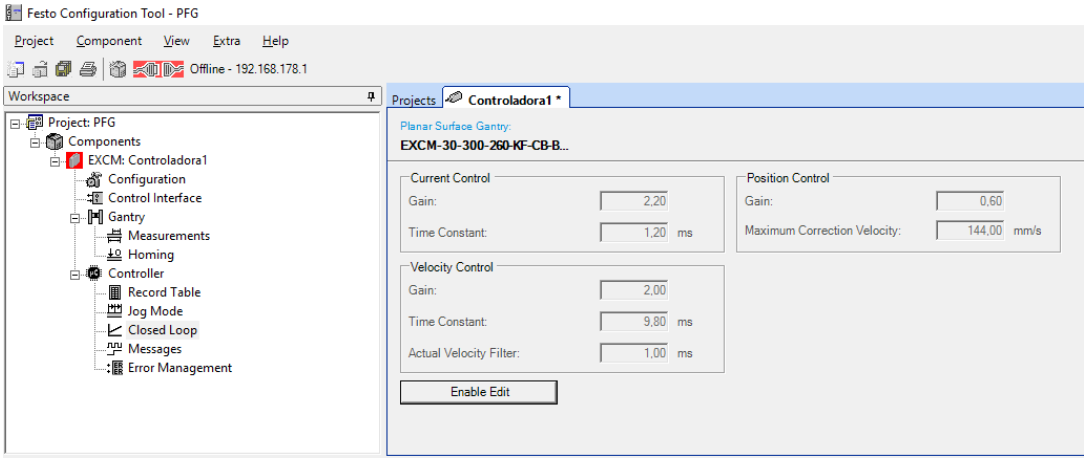


In **Jog Mode** we can configurate the parameters to move the carriage of the robot manualy in case we need to move it. In the figure we can see all the default values (recommended) and, of course, we can modify as we need.

You can see as well a figure which shows how it afects each modificable values.

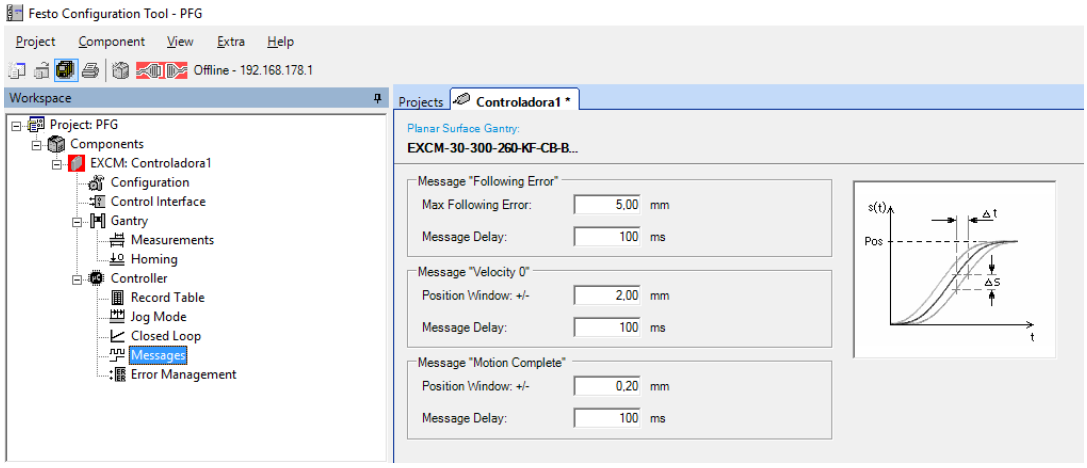


The **Closed Loop** the closed bond control of the controller for its right functioning and errors correction. It is recommend to not modify the default values, but in case you would like to modify you would have to push the *Enable Edit* botton.



In **Massages** we can configurate the warnings we want to notify us depending on the real movement curve compared to the nominal movement curve.

- Following Error: this message will activate if the difference between nominal and real positions is out of the admissible error we have established here.
- Velocity 0: this message is used to monitor if the axis stop, so we will have to introduce a distance rank we want to detect.
- Motion Complete: it indicates if at the final of a positioning operation has variated in nominal distance.



3.1.4.4 Programmed positions

A total of 18 positions have been configurated and programmated. Its distribution has been as you can see below:

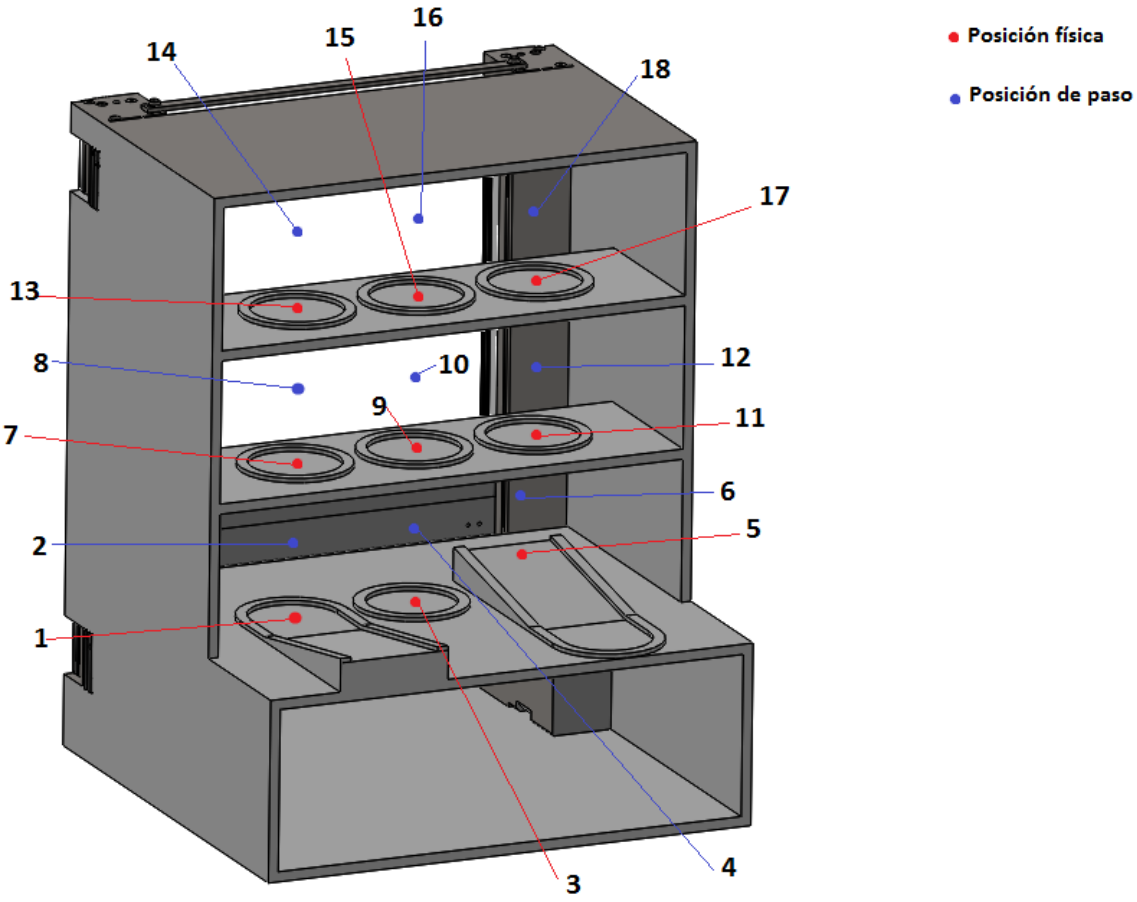


Figure 25: Configurated Points Design

The positions cited previously have been stipulated and configurated as you can see in the following positions table:

Position	Coordinate X (mm)	Coordinate Y (mm)	Velocity (mm/s)
1	46,5	19,6	1
2	51	19,6	15
3	46,5	104	1
4	51	104	15
5	46,5	190	1
6	51	190	15
7	98,5	19,6	1
8	149	19,6	15
9	98,5	104	1
10	149	104	15
11	98,5	190	1
12	149	190	15
13	242,5	19,6	1
14	247	19,6	15
15	242,5	104	1
16	247	104	15
17	242,5	190	1
18	247	190	15

Table 9: Programmed Points of FCT

Physical Positions

Odd positions (1, 3, 5, 7, 9, 11, 13, 15, and 17) are those final positions where you can interact physically with the robot design.

- 1: Entrance of Petri dishes
- 3: Position to take photos
- 5: Exit of Petri dishes
- 7,9,11,13,15 and 17: Memory positions for Petry dishes

Passing through positions

Even positions are passing through positions to its pertinent odd position.
To know what passing through position corresponds to a determinate final position, we will have to sume one to the final position, that is final position +1. An example:

- To go to position 9, we will have to go first to position 10 (9+1)

3.2 RaspberryPi

What is RaspberryPi?

RaspberryPi is a computer but with the size of a credit card, designed originally for the education and inspirited in BBC Micro in 1981.

The point of the creator Eben Upton was to create a low-cost mechanism to improve the programming abilities and hardware understanding at a pre-university level. But thanks to its small size and accessible prize, it was quickly adopted for the handy people, manufacturers and electronic enthusiasts for projects which require something more than a simple controller.

RaspberryPi is a little bit slower than a laptop or a modern computer, but it is still a complet Linux electronic device and it can proportionate all capacities expected and with a really low energy consumption.

Hardware

Raspberry Pi is an open hardware, with the exception of the principal chip, the Broadcomm Soc (system-on-a-chip), which extends to a lot of principal components of the CPU plate, memory, USB controller, etc. Most of the realized projects are open and documented, so you can build and modify yourself.

Software

The Raspberry Pi was designed by the operating system Linux and lots of Linux distributions have an optimized version for it now.

Two of the most popular options are firstly Raspbian, based on the operating system Debian; and secondly Pidora, based on the operating system Fedora. The election is a personal preference.

There are, of course, a lot of options. OpenELEC and RasBMC are both Linux distributions that are directed to be used in Raspberry Pi as a central media. There are as well no Linux systems, such as RISC OS or even Windows 10.

Specifications

Anterior models exist since 2012, but we compare below the actual ones.

	RaspberryPi 1 Model B+	RaspberryPi 2 Modelo B	RaspberryPi 3 Modelo B
Release Date	2014	2015	2016
Target Price	25€	40€	55€
Architecture	ARMv6 (32bits)	ARMv7 (32bits)	ARMv8 (64bits)
System-on-chip	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	700 MHz single-core ARM1176JZF	900 MHz 32-bit quad-core ARM Cortex-A7	1.2 GHz 64-bit quad-core ARM Cortex-A53
GPU	Broadcom VideoCore IV @ 250 MHz OpenGL ES 2.0 MPEG-2 and VC-1 (with license) 1080p30 H.264/MPEG-4 AVC		
Ram Memory	512 MB (shared with GPU)	1 GB (shared with GPU)	
USB Ports	4x USB 2.0		
Input Video	15-pin MIPI camera interface (CSI) connector		
Output Video	HDMI (rev 1.3), composite video (3.5 mm TRRS jack)		
Storage	MicroSD slot		
Network	10/100 Mbit/s Ethernet (8P8C)		10/100 Mbit/s Ethernet 802.11n wireless Bluetooth 4.1
Power Ratings	600 mA (3.0 W)	800 mA (4.0 W)	
Size	85.60 mm × 56.5 mm		
Weight	45 g		

Table 10: RaspBerry Pi comparison

3.2.1 Board configuration

We have selected the Raspberry Pi 1 model B+ because of its characteristics.

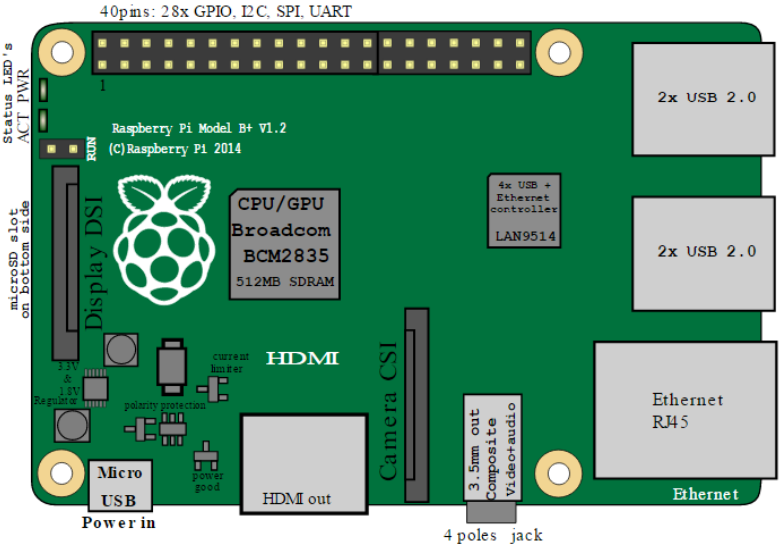


Figure 26: RaspBerry Pi B+ Board

For the right configuration of the Raspberry plate, first of all we should have:

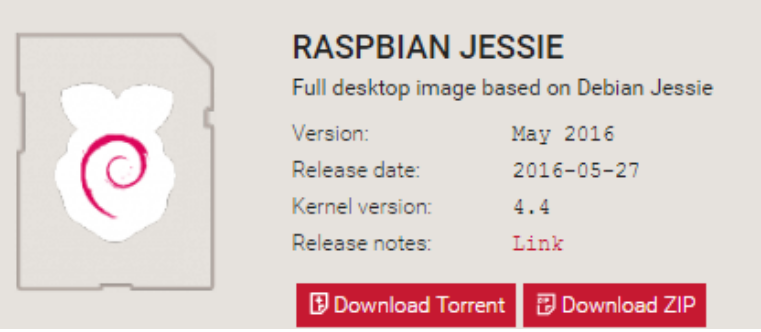
- A microSD target of a minimum of 8gb and class 4
- Wiring connection HDMI
- A screen or display with HDMI entrance
- A mouse and a keyboard with USB connection
- A power source of at least 0.7A and 5V with microUSB connection
- Ethernet wire or USB Wireless

3.2.1.1 Instalation

Before initiating the install, we should have to connect all peripherals cited previously.

Step 1: Download

We will go the official Raspeberry Pi web and we will go to the tab *Downloads*, and there we will select the operative system we want to install.

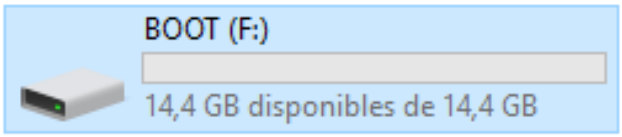


*In this project we have installed **Raspian Jessie** because it is the one recommended officially and it has a lot of pre-installed applications.*

Step 2: Image SD

Insert the SD target in the SD target reader and verify what unit letter is assigned.

You can see easily the unit letter, in this case the F:/ just by looking to the left column of the Windows explorer.



Format the micro SD target with the SDFormatter v4 program.

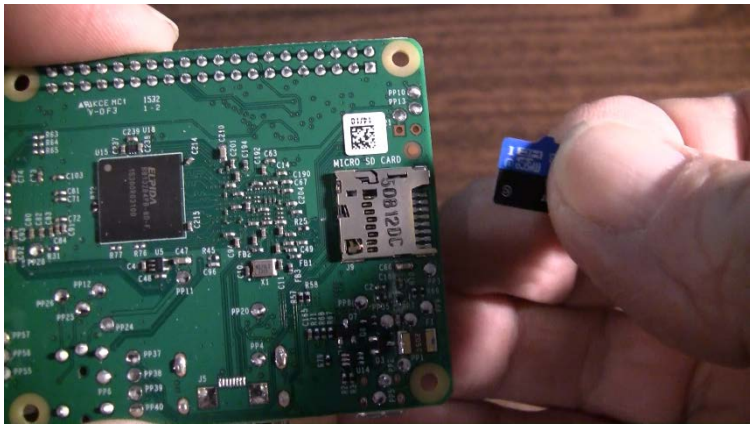
Download the Win32Disklmager utility to unzip the operating syste
m in image to SD target.

Select the image file you have removed previously and select the unit letter of the SD target.

Push in *Write* and wait till the writing is completed.

Step 3: Inserting

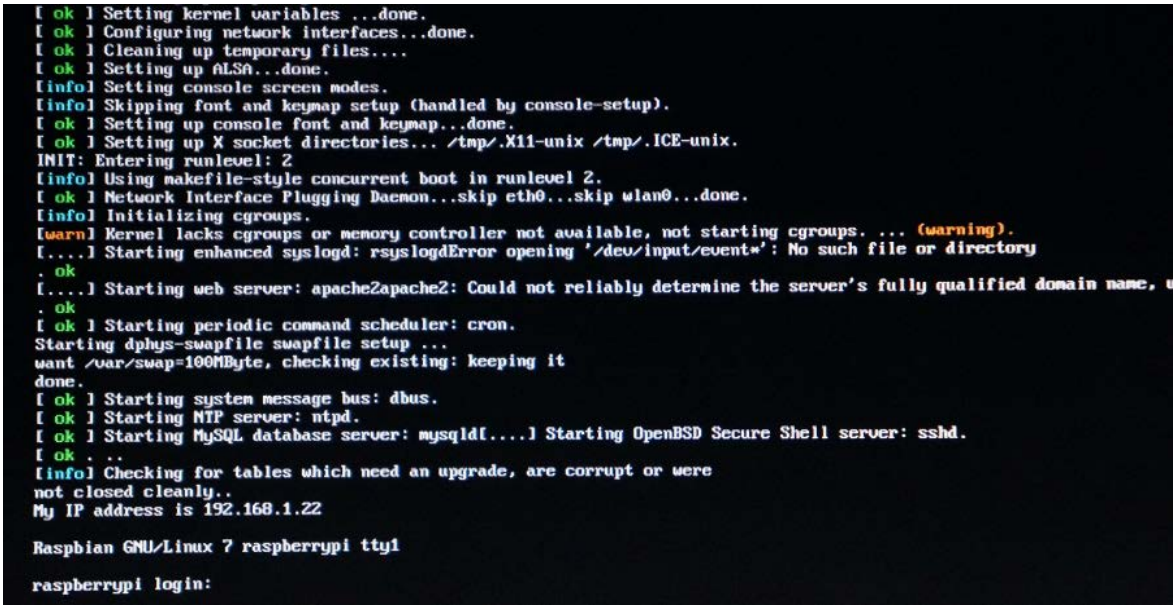
It will be inserted in the slot included in Raspberry Pi at the back and we will connect all peripherals and its supply.



At this moment we can proceed to switch on the Raspberry Pi by connecting the power source to the network.

Step 4: Installing

Directly, it will show in the monitor the install of the operating system inserted in the SD target.



The install will take about 40-50 minutes.

Step 5: Setup

The Raspberry Pi will be configured dependent on our use, so we will introduce the next command in the operating system console:

sudo raspi-config

It will show the following menu:

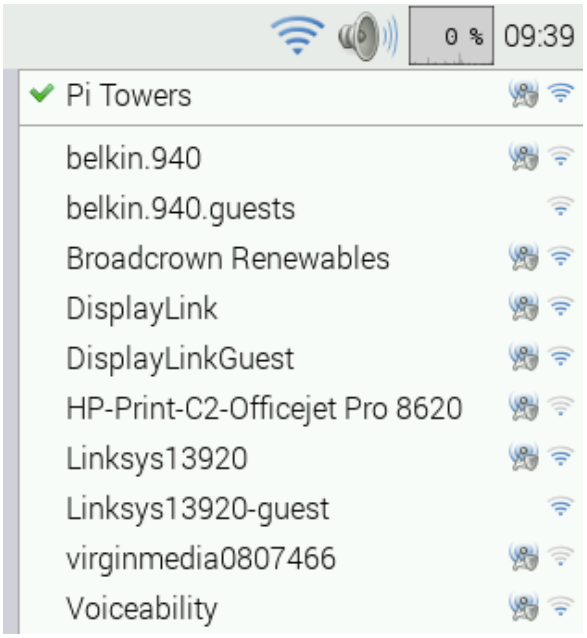
Things that we have to configure:

- Expand Filesystem: we will activate it, but it is already active by default in RASPBIAN
- Change User Password: recommended to set a user and a password different from the default.
- Enable Boot to Desktop/Scratch: we will choose that the raspberry will switch on always in the desktop for the moment, while we program.
- Internationalisation Options: here we will adjust the time zone and the language of the operating system and of the keyboard.
- Enable Camera: we will activate the camera connector.

Then we will push *Finish* and we will reinitiate the Raspberry Pi.

Step 6: Wi-Fi

We will configure what Wi-Fi network we want to connect.



By pushing in the Wi-fi symbol, it will open all available ones and we will have to click that one we have to use by default, in which one we will have to add the password and select *Remember*.

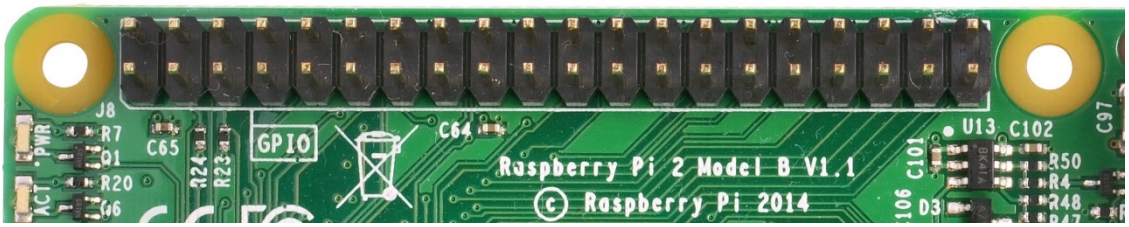
Once we are connected to Wi-fi, we will open the console and we will add the following command in order to update whenever is it necessary:

sudo apt-get update && sudo apt-get upgrade

This will take about 30 minutes, and in the end it will reinitiate Raspberry Pi and we will terminate the install.

3.2.2 GPIO

In the model of Raspberry Pi Model B +, it has GPIO pins, as shown in the following figure;



Starting from the left side of the figure below the pin number 1 and number 2 above.

3.2.2.1 Characteristics

It has a 40-pin configuration

Figure 27: GPIO configuration

- 26-pin to program as inputs or outputs (green, blue, purple and pink)
- 8-pin ground connection (black)
- 2-pin 5v Output Voltage (red)
- 2-pin 3v3 Output Voltage (orange)
- 2-pin reserved for the EEPROM

3.2.2.2 Own Configuration

In this Project we will use a total of 20-pin

- 5-pin of Digital Inputs
- 10-pin of Digital Outputs
- 3-pin GND
- 2-pin of 5v Output Voltage

In the following tables, organized by connections of each hardware that is installed

VGA connector to be used to govern the Festo robot

Phrase for programmed points	5x Digital Output
Security	3x Digital Output
Information from Robot	4x Digital Input

Tabla 11: VGA connexions from GPIO

Emergency Button (Security)

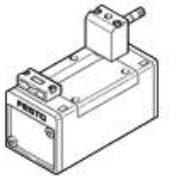
Ground	1x GND
Signal	1x Digital Input
Power Source	1x 5v

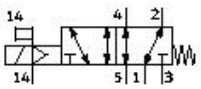
Tabla 12: Emergency Button connexions from GPIO

Electrovalve

Ground	1x GND
Signal	1x Digital Output

Tabla 13: Electrovalve connexions from GPIO





Servomotor for gripper

Signal	1x Digital Output
Power source	1x 5v
Ground	1x GND

Tabla 14: Servomotor connexions from GPIO

3.2.3 Camera

A camera will be installed in the system, and, as it is specify in the proposal, it will take photos of the sample and it will be always fixed in the same position.

It has been used the official Raspberry Pi camera, due to it includes adapted programming bookstore that will help.

Figure 28: Camera

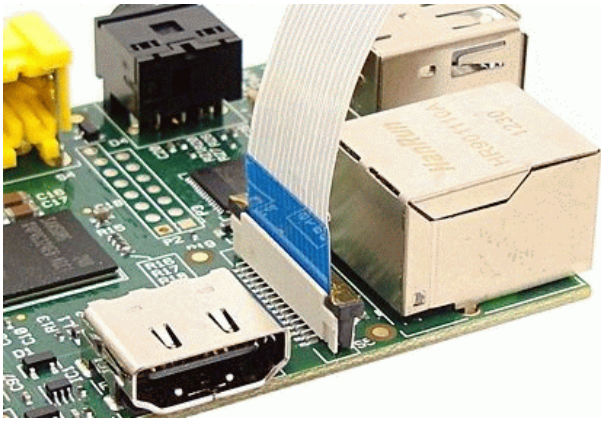
3.2.3.1 Specifications

Image Sensor	Omnivision 5647 CMOS image sensor in a fixed-focus module with integral IR filter
Resolution	5-megapixel (2592 x 1944)
Image transfer rate	1080p: 30fps 720p: 60fps
Connection	15 Pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2)
Control functions	Automatic exposure control, Automatic white balance, Automatic band filter, Automatic 50/60 Hz luminance detection, Automatic black level calibration
Temp Range	Stable image: 0° to 50° Operating: -30° to 70°
Lens size	1/4”
Dimensions	20 x 25 x 10mm
Weight	3g

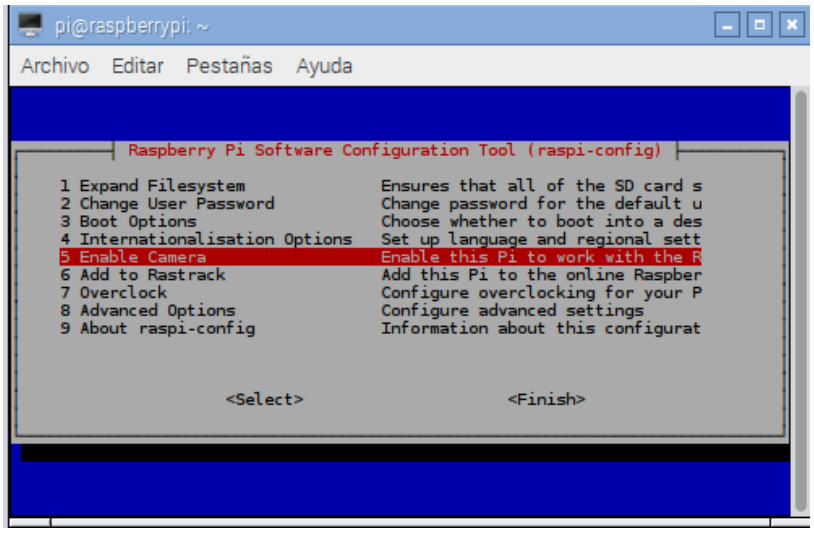
Table 15: Camara Specifications

3.2.3.2 Install

For its install, the camera has a Bus CSI-2 interface connector and the Raspberry Pi has his own entrance to this connection.



Once it is connected, we will make sure that we have able the entrance for the camera in the operative system we have installed.



To check if the camera is detected correctly, we will introduce in the console the next command.

lsusb

Then we will see what peripherals are detected by the Raspberry. Otherwise the connector or the camera are not well connected.

3.2.3.3 Set-up

At this moment, we have to prove it just in case we would have to modify any parameter to get a defined photo of our sample.

raspistill -o test.jpeg

He have to introduce this command in the console to the prove and take a photo of what we want.

In case we would have to modify any parameter, we will use the options of the camera:

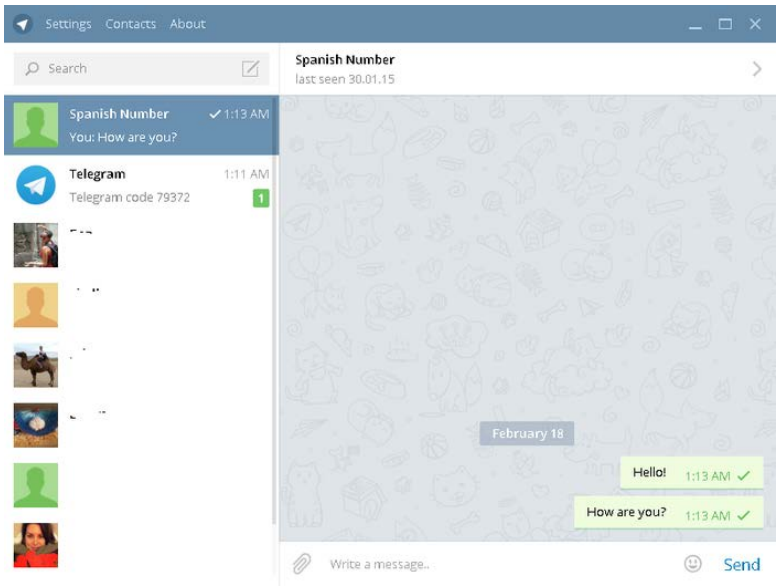
--sharpness, -sh Set image sharpness (-100 to 100)
Set the sharpness of the image, 0 is the default.
--contrast, -co Set image contrast (-100 to 100)
Set the contrast of the image, 0 is the default
--brightness, -br Set image brightness (0 to 100)
Set the brightness of the image, 50 is the default. 0 is black, 100 is white.
--saturation, -sa Set image saturation (-100 to 100)
Set the colour saturation of the image. 0 is the default.
--ISO, -ISO Set capture ISO
Sets the ISO to be used for captures. Range is 100 to 800.
--vstab, -vs Turn on video stabilization
In video mode only, turn on video stabilization.
--ev, -ev Set EV compensation
Set the EV compensation of the image. Range is -10 to +10, default is 0.

3.2.4 Telegram

This application is a free messenger service via Internet, developed by two brothers, Nikolai and Pavel Durov.

It came out at 2013 and initially it was used for mobile phones, but in the next year a lot of versions for multiplataform came out into the open.

That service consist on sending and receiving messages (text message, documents or miscellaneous) through the MTProto architecture, developed by themselves, instead of the popular XMPP.



3.2.4.1 Characteristics

This project requires Telegram to control Raspberry Pi (and in consequence the robot), and this is possible from any Smartphone or PC which have installed that applications.

His principal advantages in relation to its competitor are:

- Security
  In fact, Telegram creators would reward anyone able to hack the application.

- For free
  From its beginnings, this application has been always download without any charge, in contrast to his rivals.

- Contacts
  One of the main characteristics of the applications is that you have the possibility to have a

"nickname", so other users can add you without the need of having a visible telephone number.

- File sending
  The possibility to send or to receive all kind of files. Most interessants and usefuls are PDF, zip files, pictures, gifs and even 1.5GB files.

- Operating Systems
  There is the possibility to work in any operating system.

- Open Source
  One of the advantages that can allow Telegram evolve more broadly is that the API is open, which means you can access the source code, modify it and redistribute it, so if you're a programmer and you have good ideas, can contribute improvements of this application.

- Cloud based
  It lets you access any device chats to continue the conversation or download the file sent to you on the device you want

3.2.4.2 Install

To install the application Telegram in the Raspberry Pi, you must first have the Raspbian software installed and updated correctly and have set up the wireless network. Go to Section 3.2.1 otherwise

Step 1: Update

We will open the console and update the entire system Raspbian:

```
sudo apt-get update
sudo apt-get upgrade
```

We will have to wait between command and command, and it has a total duration of 30 minutes.

Step 2: Install

To install the application, we will use the cloning of the version on the website of github.com, as shown in figure:

Introduce the cloner command in the console:

git clone --recursive https://github.com/vysheng/tg.git

And once installed, we will have to install a set of libraries needed to use the application based on our project:

sudo apt-get install libreadline-dev make
sudo apt-get install libconfig-dev make
sudo apt-get install libssl-dev make
sudo apt-get install lua5.2 make
sudo apt-get install liblua5.2-dev make
sudo apt-get install libevent-dev make

Once installed all the libraries, we will have to access the folder created by telegram with the command:

cd tg

Now inside the folder, configure the application:

./configure

And when we thought all settings, activate the makefile inserting:

Make

This process will take more than 30 minutes, then we will terminate the installation.

3.2.4.3 Setup

Telegram is based on associating a user name to a phone, but only the phone number will be required to authenticate the account, from there it will take only use the user that has successfully authenticated.

To register must be put in the following command console:

bin/telegram-cli -k tg-server.pub

“

In our registration process, the following error has shown, that is because it is a version for 64-bit and 32-bit is our Raspberry Pi.

```
pi@raspberrypi:~/tg $ bin/telegram-cli -k tg-server.pub -W
Telegram-cli version 1.4.1, Copyright (C) 2013-2015 Vitaly Valtman
Telegram-cli comes with ABSOLUTELY NO WARRANTY; for details type `show_license'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show_license' for details.
Telegram-cli uses libtgl version 2.1.0
Telegram-cli includes software developed by the OpenSSL Project
for use in the OpenSSL Toolkit. (http://www.openssl.org/)
I: config dir=[/home/pi/.config/telegram-cli]
> telegram-cli: tgl/mtproto-utils.c:101: BN2ull: Assertion `0' failed.
```

We have solved entering the file and we discussed the line 101 that is giving the error:

In file tgl/mtproto-utils.c, replace in lines 101 and 115:

assert (0); // As long as nobody ever uses this code, assume it is broken.

by

//assert (0); // As long as nobody ever uses this code, assume it is broken.

Once you save the modified file, we will go to the console and we will have to recompile introducing into the console

cd tg
make

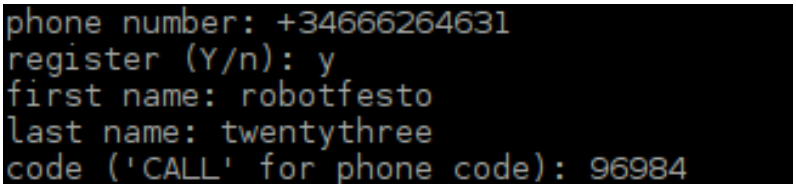
”

Now we can check in without any mistakes.

The symbol “>” will show up and then we will have to push any key and introduce:

1. The telephone number we want to register, with the area code and the sign +.
2. The name
3. The surname
4. The activation code we will recieve in our telephone once we have introduced it.

This figure shows the input data of our raspberry



3.2.4.4 Test

After the installation, we can prove that everything went right by typing in the command console:

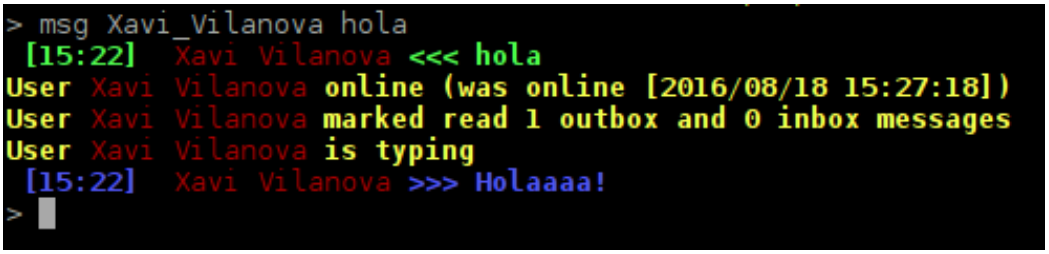
Help

We observe all the commands available to interact with Raspberry Pi-Telegram

We'll try to send a message from the Raspberry Pi to another account, for this we use the following command:

msg Firstname_Lastname Hola!

The following figure shows the test performed successfully seen from the console of the Raspberry Pi



In this figure the test performed successfully since the application installed on a Smartphone

3.3 Programming

For Raspberry Pi programming, you can use different languages and libraries provided govern the GPIO ports and the camera.

In this case it was decided to use the language **Lua** and **Python** language.

The program is divided into two processes:

- *Main Program:* This program has been programmed with Lua language as it contains many functions of different libraries and executions to other programs, and required a language that did not use many resources.

  Lua has the process transparent to the user compilation and can be made in advance to increase performance and reduce memory usage by dispensing compiler.

- *Subrutines Program*: For subprograms that are called pro the main program, has been used Python language, because it is easy and comfortable writing language.

  Note that this language is the most widely used programming with Raspberry Pi

3.3.1 Block Diagram

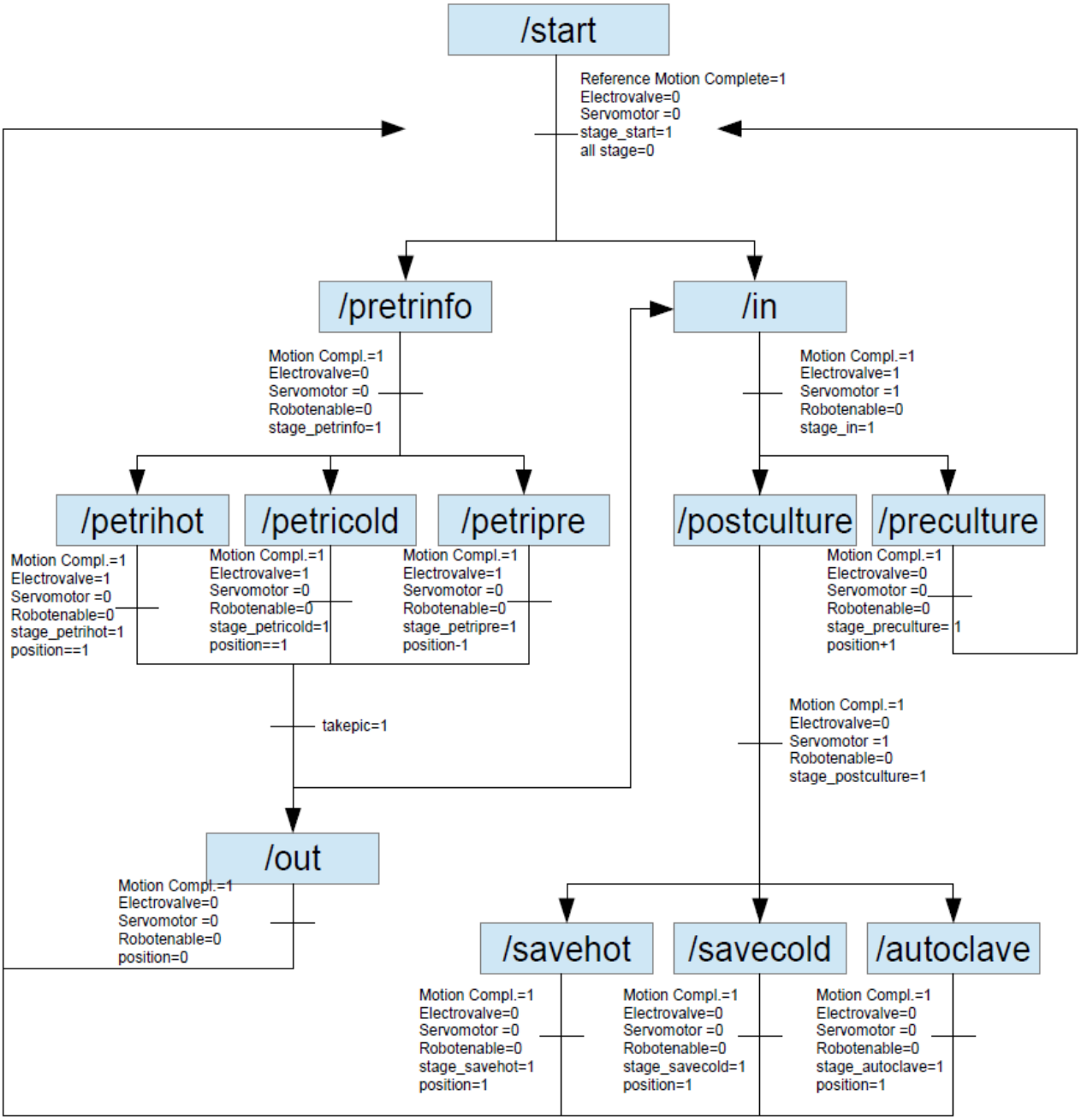


Figure 29: Block Diagram of system

3.3.2 Commands

Listed here the commands, their definitions and their actions that are necessary for control by Telegram application and interact with the robot

/ping
Action: Telegram connectivity check
Answer: pong

/reboot
Action: Reboot the raspberry system
Answer: 'Rebooting the system...'

/start
Action: Boot the system and prepares the robot to be used
Answer: 'The system is READY'
'Available Commands = [/in, /petrinfo]'

/in
Action: Robot prepared to include a petri dish within the system
Answer: 'Got the new plate'
'Available Commands = [/preculture, /postculture]'

/preculture
Action: Move the plate to the preculture points
Answer: 'Saved the plate to the preculture zone'

/postculture
Action: Enables the commands for the postculture options
Answer: 'You have some options for a post culture plate process'
'Available Commands = [/savecold, /savehot, /autoclave]'

/petrinfo
Action: Prepares the robot to pick up a plate within the system
Answer: 'Which plate are you asking for?'
'Available Commands = [/petrihot, /petricold, /petripre]'

/savehot
Action: The robot will save the plate in the calefactor zone
Answer: 'Saved the plate to the hot spot'

/savecold
Action: The robot will save the plate in the refrigerator zone

Answer: 'Saved the plate to the cold spot'

/autoclave

Action: The robot will move to the autoclave
Answer: 'Moved the plate to the autoclave'

/petrihot

Action: The robot picks up the plate from the calefactor and makes a pic
Answer: 'Sent pic to confirm it'
'Available Commands = [/out, /in]'

/petricold

Action: The robot picks up the plate from the refrigerator and makes a pic
Answer: 'Sent pic to confirm it'
'Available Commands = [/out, /in]'

/petripre

Action: The robot picks up a preculture plate and makes a pic
Answer: 'Sent pic to confirm it'
'Available Commands = [/out, /in]'

/out

Action: Leave the plate picked previosly to the departure zone
Answer: 'Left the plate in the departure zone'

/systemstatus

Action: Analize which spots are used
Answer: Show us the status of the spots are we using or not

3.3.3 Main program

The complete program has not been added, the entire program is listed in the annexes to this document, just some pieces have been used for explanation.

We declare the variables that will make through conditions for the program to be more robust.

Variable positions are also reported for the system to have control of what is inside and what not.

```
paso_in = 0
paso_start = 0
paso_petrinfo = 0
paso_preculture = 0
paso_postculture = 0
paso_petrihot = 0
paso_petricold = 0
paso_petripre = 0

posicion_petripre1 = 0
posicion_petripre2 = 0
posicion_petripre3 = 0
posicion_cold = 0
posicion_hot = 0
posicion_autoclave = 0
```

The function get_title:

This function is for the name of who is sending messages to the client Telegram of Raspberry, and identify who is interacting at that time. Information can be used to filter which users we want to use the robot and which ones no.

```
function get_title (P, Q)
  if (Q.type == 'user') then
    return P.first_name .. " " .. P.last_name
  elseif (Q.type == 'chat') then
    return Q.title
  elseif (Q.type == 'encr_chat') then
    return 'Secret chat with ' .. P.first_name .. ' ' .. P.last_name
  else
    return ''
  end
end
```

The function do_notify:

With this feature it will show on the console of the Raspberry that we have received messages and from whom.

```
function do_notify (user, msg)
  local n = notify.Notification.new(user, msg, icon)
  n:show ()
end
```

The function sleep:

Function that implements a pause in the program if needed in the main function.

```
local clock = os.clock

function sleep(n)
  local t0 = clock()
  while clock() - t0 < n do end
end
```

The function on_msg_recieve:

This is the main function of the program because is the one which decides what to do according to the message received. The program consists basically of execution "if", since according to the message received, will enter a "if" or another. In the following piece, simply states that the program remains in the first "if" until the program and connections to take effect Telegram.

```
function on_msg_receive (msg)
  if started == 0 then
    return
  end
  if msg.out then
    return
  end
```

From here the "thread of the program" will come into its corresponding piece depending on the *do_notify* received by the user from the mobile. In the following case, if the user sends a message that contains only "/ ping" the robot will automatically reply with the response function SEND_MSG "pong".

```
do_notify (get_title (msg.from, msg.to), msg.text)

if (msg.text == '/ping') then
  if (msg.to.id == our_id) then
    send_msg (msg.from.print_name, 'pong', ok_cb, false)
  else
    send_msg (msg.from.print_name, 'pong', ok_cb, false)
  end
  return
end
```

The next "if" implements the call subprograms, as seen if the client receives Raspberry Telegram in their chat the word "/ start" , the program will make a os.execute to the program defined in the direction 'path' calling subrutinas.py

```
if (msg.text == '/start') then
  paso_in = 0
  paso_start = 0
  paso_petrinfo = 0
  paso_preculture = 0
  paso_postculture = 0
  paso_petrihot = 0
  paso_petricold = 0
```

```
  paso_petripre = 0
  os.execute('sudo python '..path..'/subrutinas.py start')
  send_msg (msg.from.print_name, 'The system is READY', ok_cb, false)
  send_msg (msg.from.print_name, 'Available Commands = [/in, /petrinfo]', ok_cb, false)
  paso_start=1
  return
end
```

When the functions finish in that subprogram, it will return to where it left off and in this case send the following messages to the SEND_MSG "The system is READY" function and "Available commands = [/ in, / petrinfo]"

In the next "if", it appears as when we write the "/ preculture" command has conditions, and that should come first stage "/ in" because otherwise it will return to "command not allowed"

It should be properly authorized by his stage predecessor command, in this case we include the board in the position that is free, that should be all taken we would return a "the preculture spots are full" and we would return by the exit area.

```
if (msg.text == '/preculture') then
    if (paso_in == 1) then
            if (posicion_petripre1==0) then
                    os.execute('sudo python '..path..'/subrutinas.py preculture1')
                    send_msg (msg.from.print_name, 'Saved the plate to the preculture zone 1', ok_cb, false)
                   posicion_petripre1=1
                   paso_in=0
            elseif (posicion_petripre2==0) then
                    os.execute('sudo python '..path..'/subrutinas.py preculture2')
                    send_msg (msg.from.print_name, 'Saved the plate to the preculture zone 2', ok_cb, false)
                   posicion_petripre2=1
                   paso_in=0
            elseif (posicion_petripre3==0) then
                    os.execute('sudo python '..path..'/subrutinas.py preculture3')
                    send_msg (msg.from.print_name, 'Saved the plate to the preculture zone 3', ok_cb, false)
                   posicion_petripre3=1
                   paso_in=0
            else
                    send_msg (msg.from.print_name, 'The preculture spots are full', ok_cb, false)
                    os.execute('sudo python '..path..'/subrutinas.py out')
                    send_msg (msg.from.print_name, 'Left the plate in the departure zone', ok_cb, false)
                   paso_in=0
            end
    elseif (paso_in==0) then
            send_msg (msg.from.print_name, 'Command not allowed', ok_cb, false)
    end
  return
end
```

3.3.4 Robot Points Comunication

To communicate with the robot from the Raspberry Pi, it has been used digital inputs 5-pin which has enabled the robot in the VGA connector, which are governed by the GPIO of the Raspberry Pi, what is programmed as variables named in Python.

In this table all the nomenclature used in the different components can be seen:

VGA pin connector (Robot Controller)	GPIO pin connector (Raspberry Pi)	Variable (in Python)	Value pin
2	*29*	*phrasepin2*	***1***
3	*31*	*phrasepin3*	***2***
4	*33*	*phrasepin4*	***4***
5	*35*	*phrasepin5*	***8***
6	*37*	*phrasepin6*	***16***

Table 16: Comunication between VGA-GPIO-Programming

Pin values 1,2,4,8 and 16 are values that the combination of them can run the programmed points in the Robot Controller.

The following table shows the relationship of values with the programmed points:

Programmed Point (in FCT)	Combination (in Python)	Combination (in GPIO)
1	Phrasepin2	29
2	Phrasepin3	31
3	Phrasepin2 + phrasepin3	29 + 31
4	Phrasepin4	33
5	Phrasepin4 + phrasepin2	33 + 29
6	Phrasepin3 + phrasepin4	31 + 33
7	Phrasepin2 + phrasepin3 + phrasepin4	29 + 31 + 33
8	Phrasepin5	35
9	Phrasepin5 + phrasepin2	35 + 29
10	Phrasepin5 + phrasepin3	35 + 31
11	Phrasepin5 + phrasepin2 + phrasepin3	35 + 31 + 29
12	Phrasepin5 + phrasepin4	35 + 33
13	Phrasepin5 + phrasepin4 + phrasepin2	35 + 33 + 29
14	Phrasepin5 + phrasepin4 + phrasepin3	35 + 33 + 31
15	Phrasepin5 + phrasepin4 + phrasepin3 + phrasepin2	29 + 31 + 33 + 35
16	Phrasepin6	37
17	Phrasepin6 + phrasepin2	37 + 29
18	Phrasepin6 + phrasepin3	37 + 31
Reference	Phrasepin2+phrasepin3+phrasepin4+phrasepin5+phrasepin6	29 + 31 + 33 + 35 + 37

Table 17: Relationship of values from Python with FCT Points

3.3.5 Subprograms

The complete program has not been added, the entire program is listed in the annexes to this document, just some pieces have been used for explanation.

For the subprogram, which is scheduled in Python because of its comfortable programming, first we have to import the libraries you need for subroutines. In our case we will import the library GPIO ports to control inputs and outputs of the Raspberry Pi, and will also include the libraries *sys*, *time* and *os*.

```
import RPi.GPIO as GPIO
import sys
import time
import os
```

You must declare GPIO mode, which in our case will be used to GPIO.BOARD number with the actual port numbers and name if required.

```
GPIO.setmode(GPIO.BOARD)
```

We declare all pins used as inputs or outputs and give them a name in order to facilitate the interpretation of the program.

```
startbutton= 7
electrovalve= 11
motioncompletebutton= 12
startphrase= 16
robotenable= 18
robotreset= 22
phrasepin2= 29
phrasepin3= 31
phrasepin4= 33
phrasepin5= 35
phrasepin6= 37
infopin11= 32
infopin12= 36
infopin13= 38
infopin14= 40
```

Now is the step of declaring pins as digital inputs or digital output according to your need.

```
GPIO.setup(startbutton,GPIO.IN)
GPIO.setup(electrovalve,GPIO.OUT)
GPIO.setup(motioncompletebutton,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(startphrase,GPIO.OUT)
GPIO.setup(robotenable,GPIO.OUT)
GPIO.setup(robotreset,GPIO.OUT)
GPIO.setup(phrasepin2,GPIO.OUT)
GPIO.setup(phrasepin3,GPIO.OUT)
GPIO.setup(phrasepin4,GPIO.OUT)
GPIO.setup(phrasepin5,GPIO.OUT)
GPIO.setup(phrasepin6,GPIO.OUT)
GPIO.setup(infopin11,GPIO.IN)
```

```
GPIO.setup(infopin12,GPIO.IN)
GPIO.setup(infopin13,GPIO.IN)
GPIO.setup(infopin14,GPIO.IN)
```

Statement Array command commandlist receive from the main program.

```
commandlist=['start', 'reboot','in','preculture', 'postculture', 'petrinfo', 'savehot', 'savecold', 'autoclave', 'petrihot', 'petricold','petripre','out']
```

When a command that is not on the list is received.

```
if sys.argv[1] not in commandlist:
   print "Unrecognized command, try again"
```

When the main program aims to this program subprograms, the eachArg function compares the command sent by the user Telegram with the subprogram, when both agree then the program is in that subprogram is performed.

```
elif eachArg == 'in':
   GPIO.output(electrovalve,0)
   GPIO.output(robotenable,1)
   GPIO.output(robotreset,1)
   time.sleep(3)
   GPIO.output(phrasepin2,0)
   GPIO.output(phrasepin3,1)
   GPIO.output(phrasepin4,0)
   GPIO.output(phrasepin5,0)
   GPIO.output(phrasepin6,0)
   time.sleep(3)
   GPIO.output(startphrase,1)
   GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
   GPIO.output(startphrase,0)
   GPIO.output(electrovalve,1)
   time.sleep(3)
   GPIO.output(phrasepin2,1)
   GPIO.output(phrasepin3,0)
   GPIO.output(phrasepin4,0)
   GPIO.output(phrasepin5,0)
   GPIO.output(phrasepin6,0)
   time.sleep(3)
   GPIO.output(startphrase,1)
   GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
   GPIO.output(startphrase,0)
   GPIO.output(servomotor,1)
   GPIO.output(robotenable,0)
   GPIO.output(robotreset,0)
   GPIO.output(startphrase,0)
   time.sleep(3)
```

In the "reboot" subprogram, we will use the library *os* to use a sudo command so you can restart the Raspberry.

```
elif eachArg == 'reboot':
   os.system('sudo shutdown -r now')
```

Chapter 4
Wiring Design

4.1 General Wiring

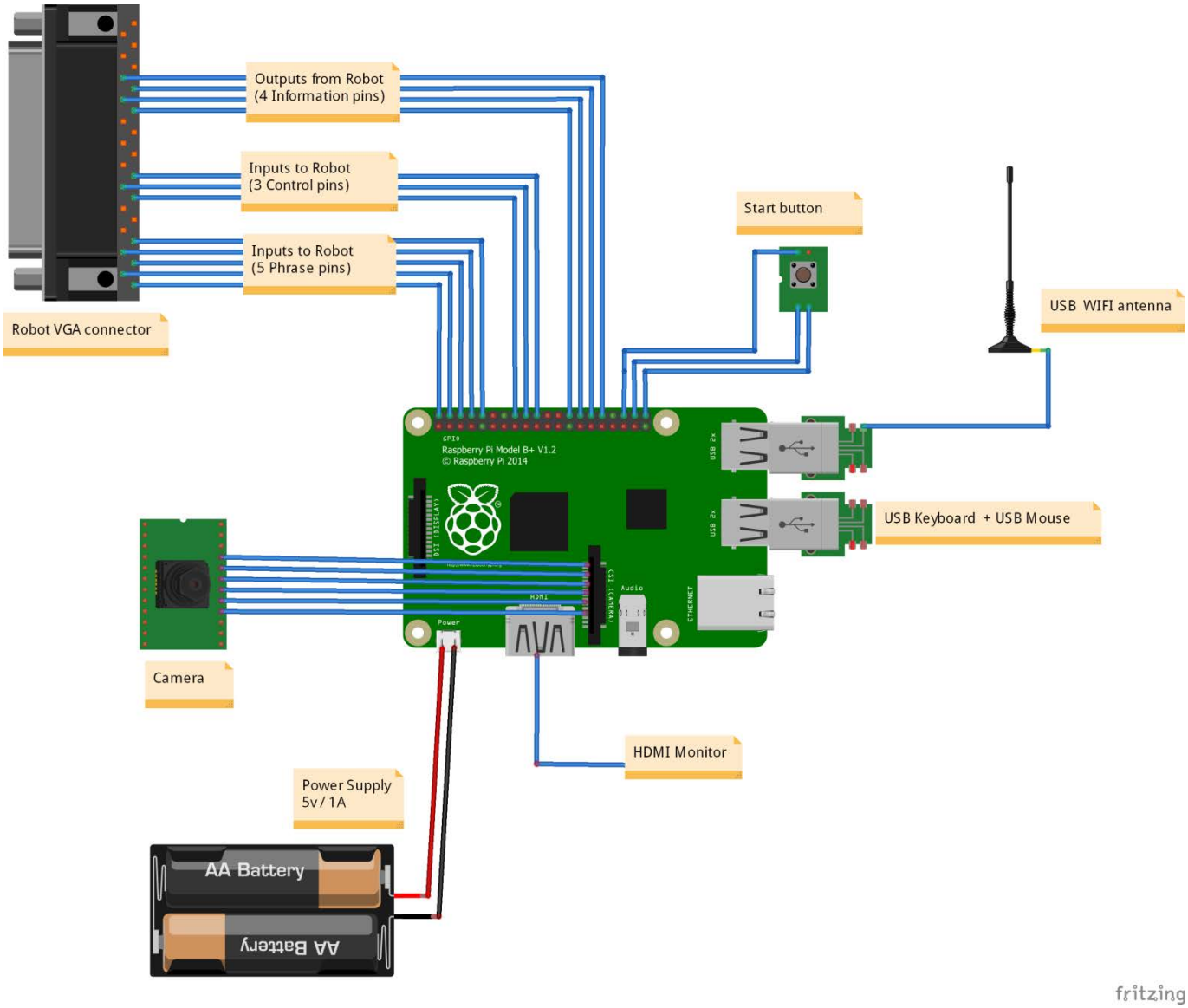


Figure 30: General Wiring

4.2 Simulation RPi Wiring

4.2.1 Protoboard

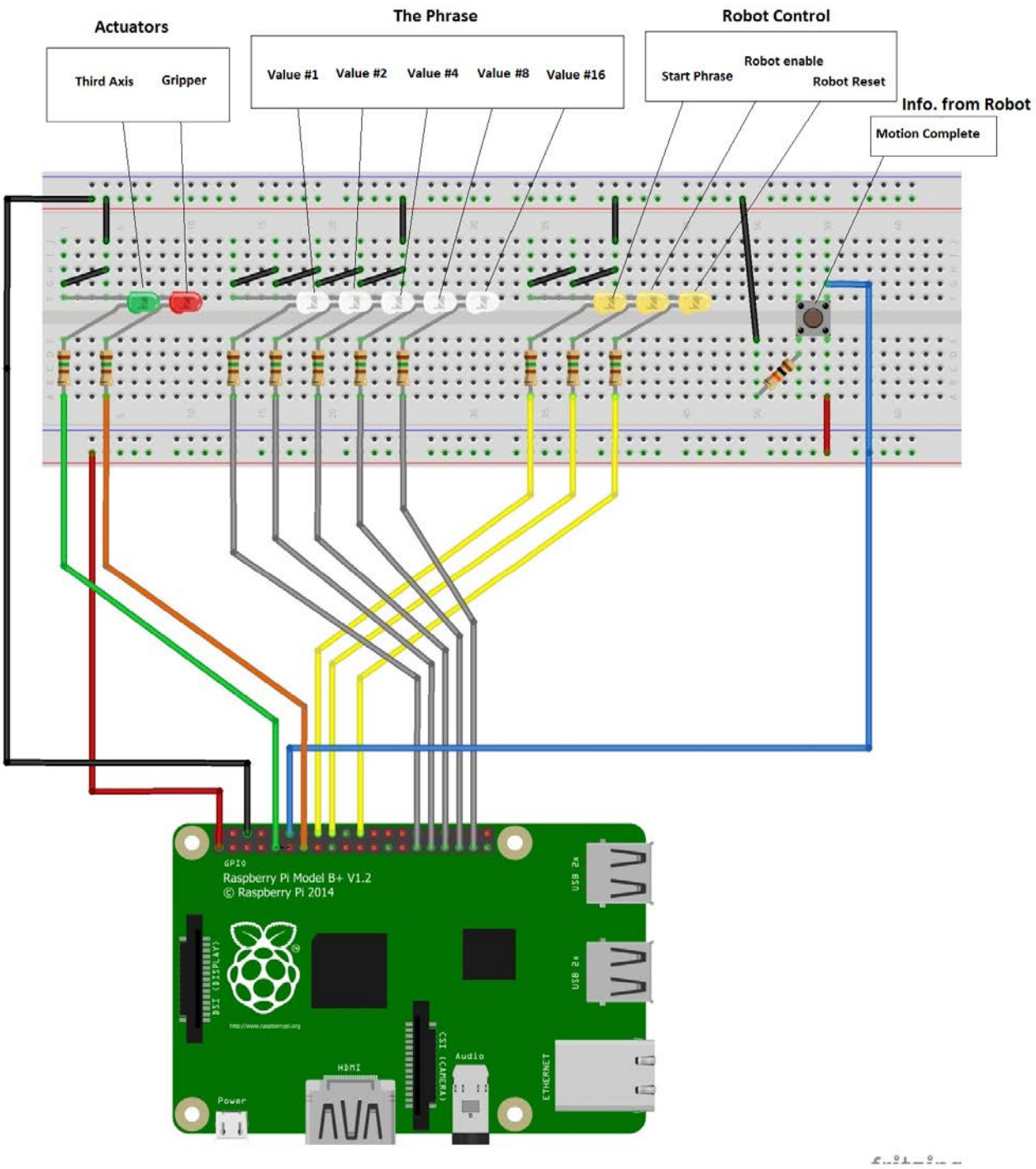


Figure 31: Wiring Simulation Protoboard

4.2.2 Schematic

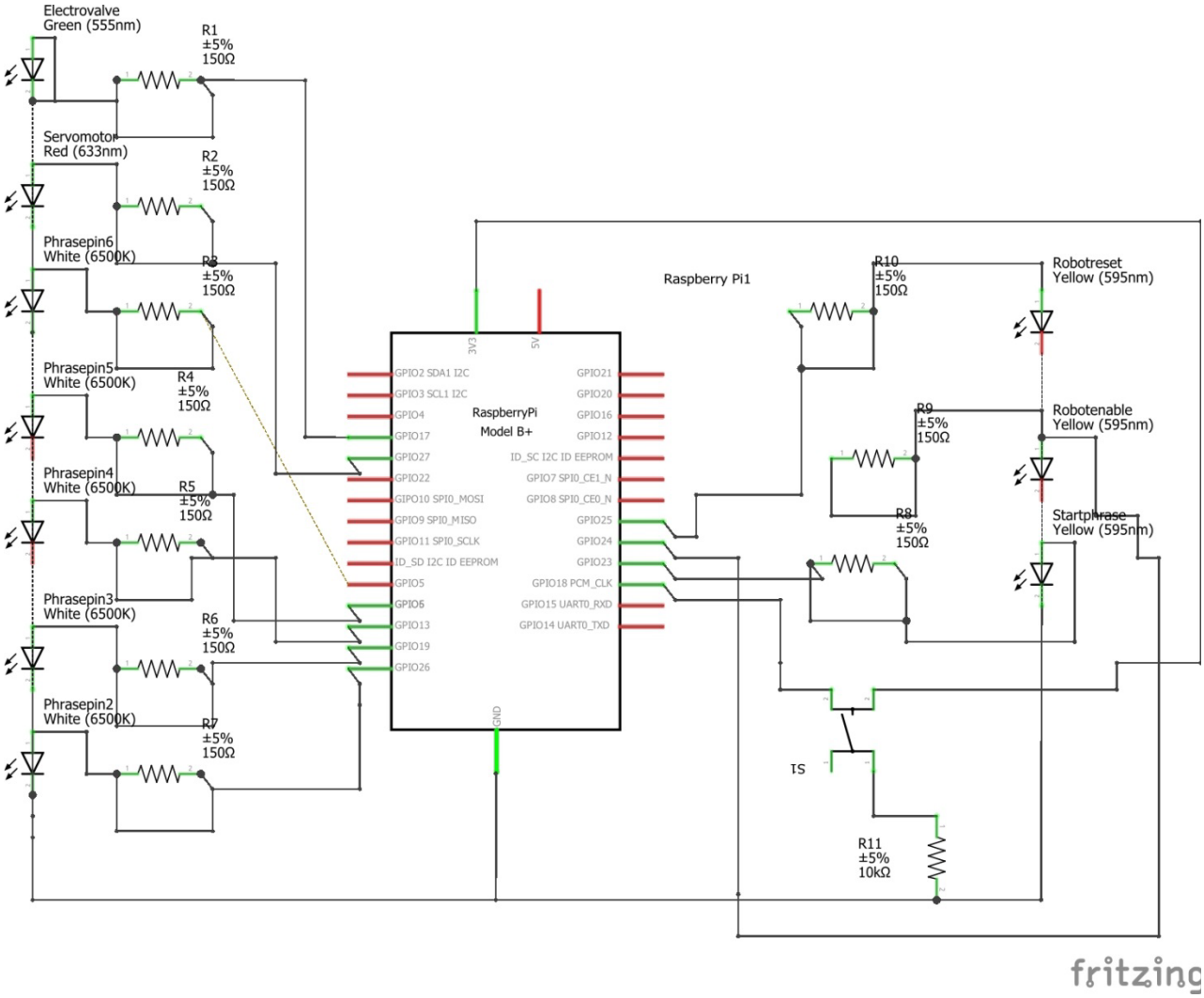
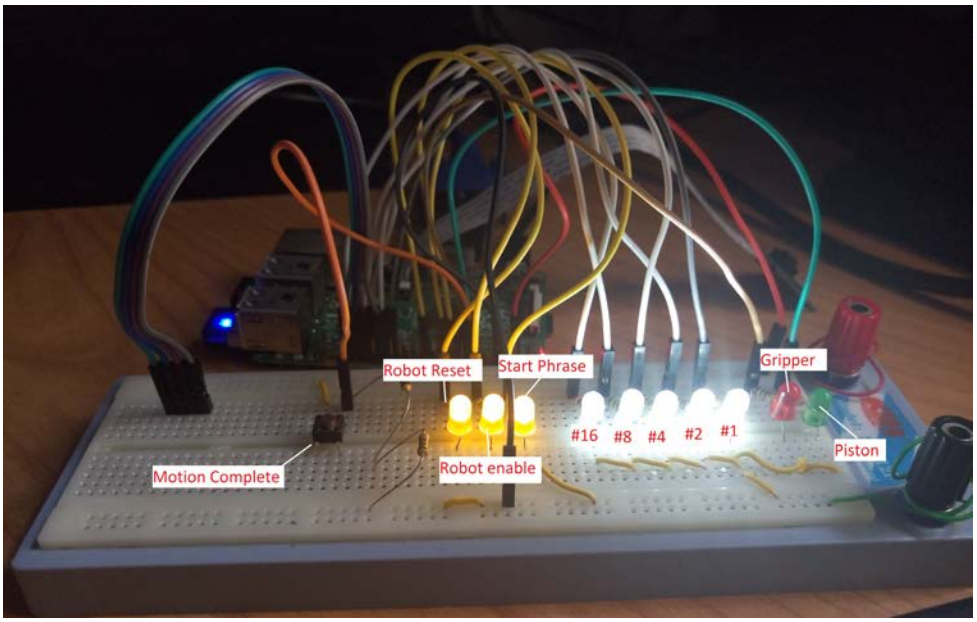


Figure 32: Wiring Simulation Schematic

Chapter 5
Test and Results

Little introduction for interpretate the nexts steps:



The button motion complete, simulates the robot and is when the robot do the movement

The robot reset and the Robot enable is always before send the 5 leds phrase
The Start phrase is always after we sent the 5 leds phrase combination
The red led shine is when we close the Gripper
The green led shine is when we extend the piston
The white leds are the phrase, in function which we use, its diferent value. For exemple we switch on the led #2 and the led #8, the phrase will be: 2+8=10 so, the robot will going to the point 10 (programmed in the FCT)

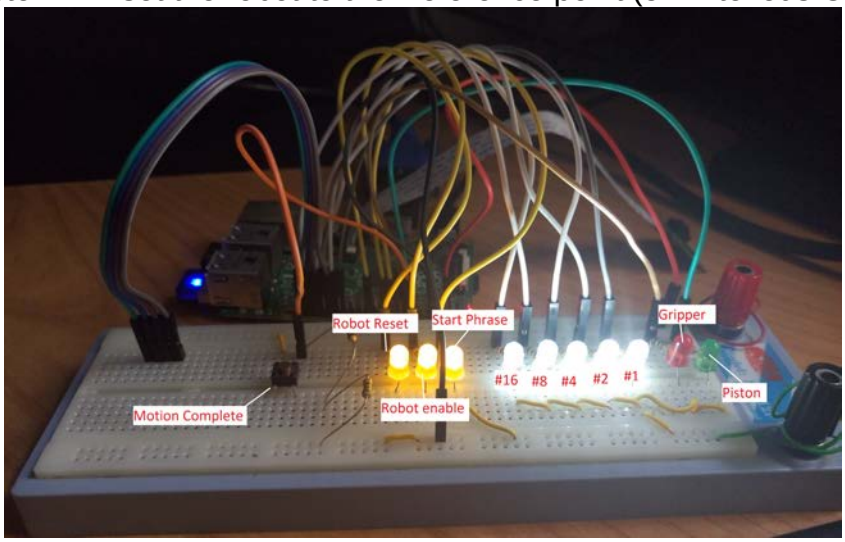
Step1: Start the system and get the ready

Lets start to understand how this system works. I going to show only a few sequences of program.

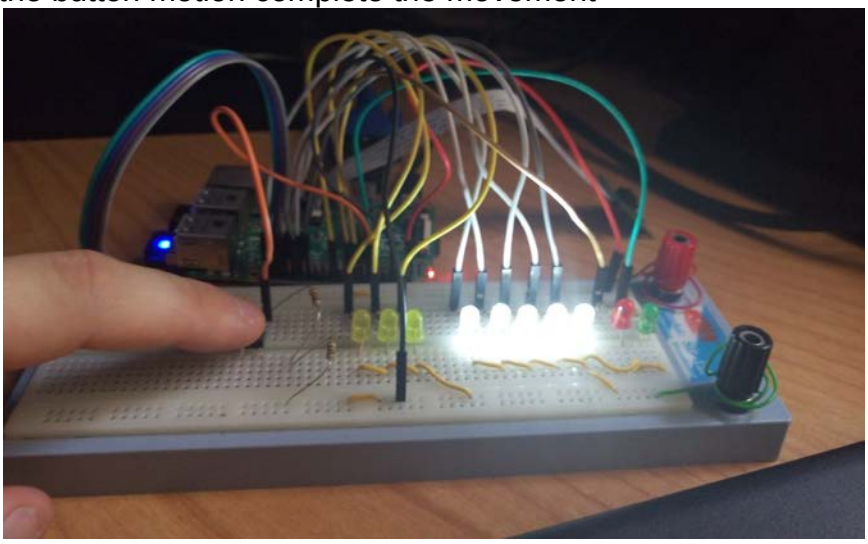
When Robot is on, we can switch on the Raspberry Pi, the program will start automatically just waiting for the initiate command from the Telegram

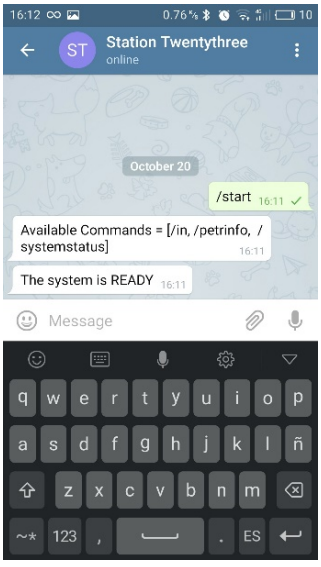
For start the system we must to type the correct command “/start” if not the system will give us a “This command is not allowed”

Now the system will set the robot to the Reference point (5 white leds ON).



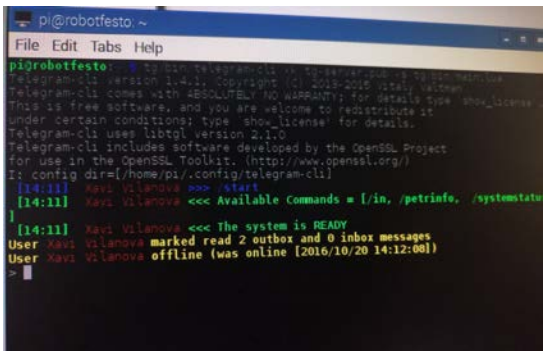
When the Start phrase is on is when the phrase has been sent to the robot, so we have to press the button motion complete the movement





In this part, is when the system will say toy in Telegram that the system is ready to start any process, and also will give you a tip of commands available to do the next step

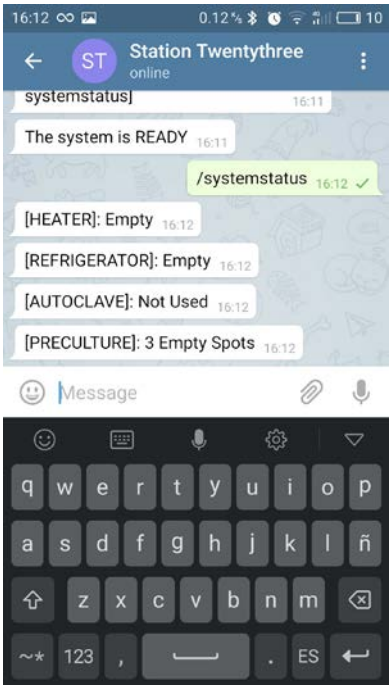
We can see how the Telegram "talks" with the Raspberry in the command screen of the Debian in the next figure:



Step2: Systemstatus

Before introduce a new plate, we can view the system status and see if the system has petri dish using some application or not.

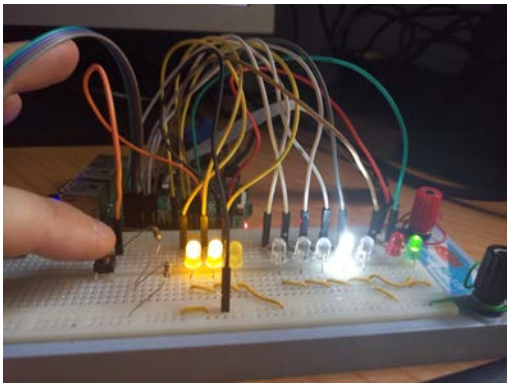
Also we can see the date and the time we introduced the petri plate in the application



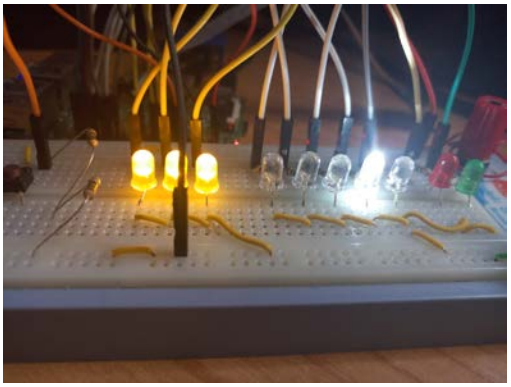
In this case, when we type the command “/systemstatus” the Telegram said us that all spots are empty, so we can introduce a plate in all of them

Step3: Introduce a new plate in the system

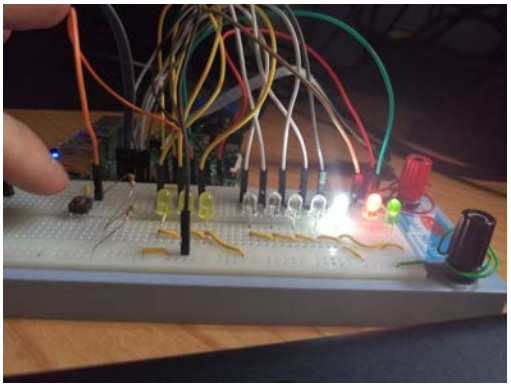
For introduce a new plate into the system we have to do a “/in” command
The robot will go to the point 2



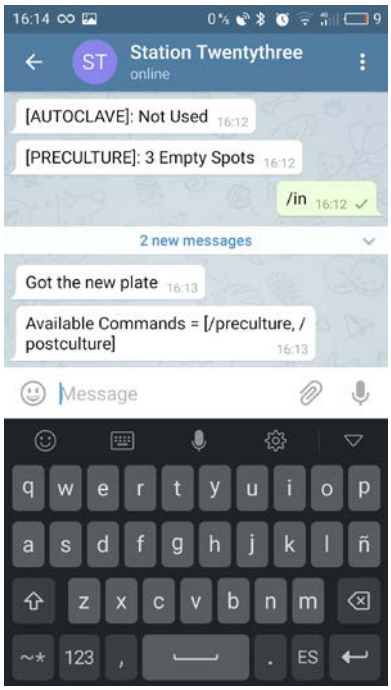
When the motion complete is pressed



The robot will down to the point 1 to take the new plate, if you see in the pic, first active the piston and after that the Gripper close

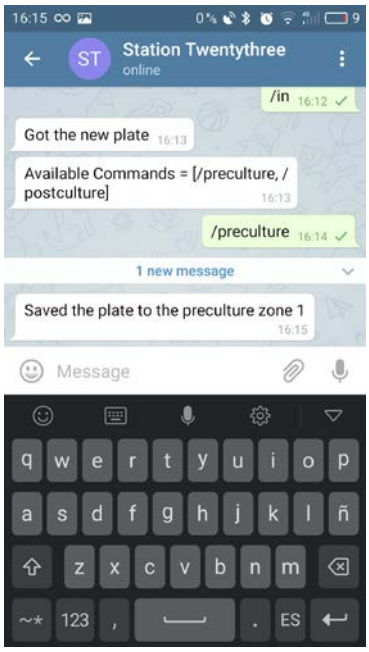


After that we will receive a new message in the Telegram application saying us that the plate is in the system waiting for do the next step:



Step4: Save the plate in the preculture zone

So now we are going to type the "/preculture" command and the system will save the plate in any free spot in the preculture zone.



In this case the plate is saved in the spot 1, but the system have the spot 1, 2 & 3. The system will save the plate automatically in one of the free spot

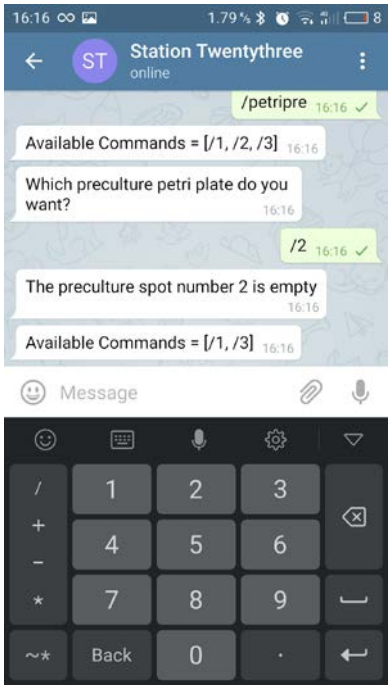
.

If the all the spots are full the system will say us that are the spots are not available, so the system will return us the plate.

Step 5: Take out a plate from the system

We have to use the command "/petrinfo" to take out a plate

The system will ask us which we are interested to take out, so in this case we are going to use the “/petripre” to take out a preculture petri from the system that we introduced in the last example



The system will ask us which of the three we are interested. If we type one that is empty, the system will say us that is empty and we have other options

If we put the correct one, the system will get the plate and will go to the camera zone to send us a pic to confirm that the plate is the correct one

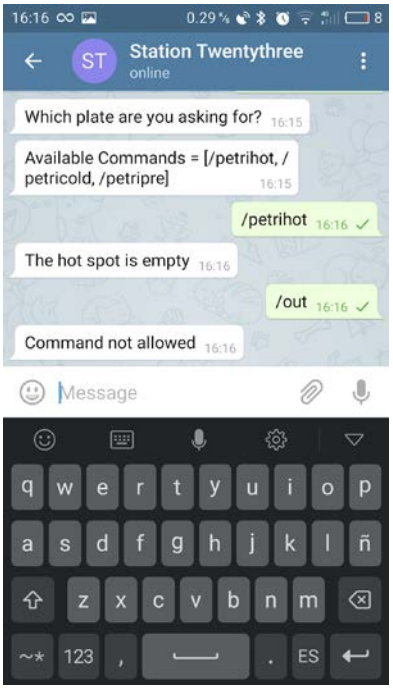
If the plate was incorrect, we can introduce again the plate in the system typing “/in”.

So in this case is the correct, we can type “/out” to get it

Step6: wrong command

What’s up if we type an incorrect command?

The system will say us that the command is not allowed to do this command:



Step7: Try to take out an empty spot

What's up if we are trying to take out an inexisting plate?

The system will have the control all the moment of the plate are in, so if the spot don't have any plate the system always will say you that is empty

Example:

Chapter 6
Real implementation

For the implementation of the simulation to the real robot connection, you will need to make some modifications:

6.1 Modification of the wiring diagram

- Add optocouplers, because the robot works at 24v, and works on 3-5v raspberry and we could damage it.
- Add the 4 inputs information robot

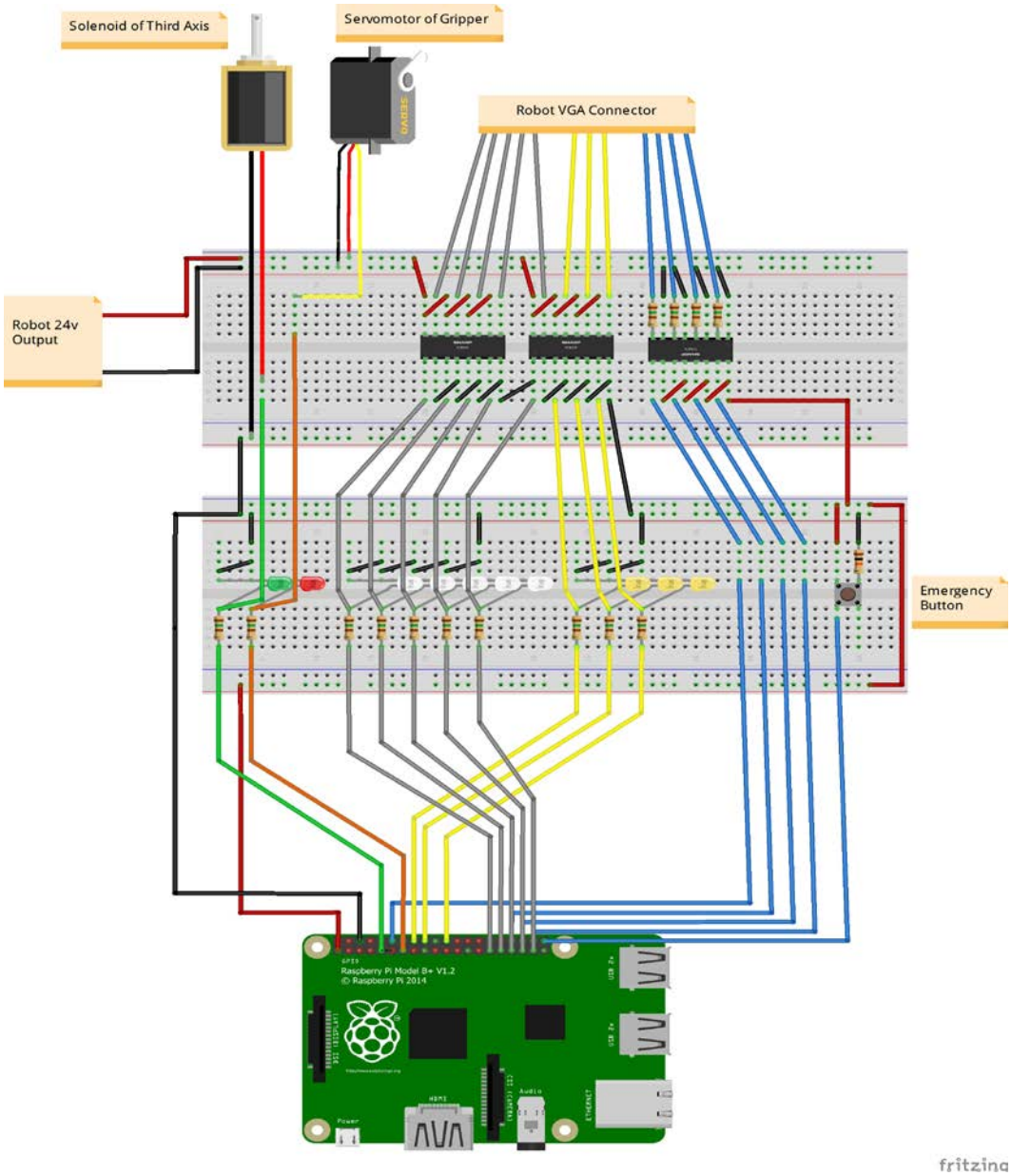


Figure 33: Modification of the wiring diagram

6.2 Modification of the Python programming

- Reverse logic, because the robot is NPN→ All 0 will be 1 and 1 are all 0
- Add new program for controlling the servomotor

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(13,GPIO.OUT)    #Ponemos el pin 13 como salida
p = GPIO.PWM(13,50)        #Ponemos el pin 13 en modo PWM y frecuencia 50
p.start(7.5)               #Enviamos un pulso del 7.5% para centrar la pinza


    p.ChangeDutyCycle(4.5)    #Enviamos un pulso del 4.5% para cerrar pinza
    p.ChangeDutyCycle(10.5)   #Enviamos un pulso del 10.5% para abrir pinza
```

Chapter 7

Budget

7.1 Manpower Budget

The estimation oficial price per hour is:

- 50€/h Industrial Engineer
- 35€/h Informatic Engineer
- 25€/h 3D Draftsman

Concept	Role	Estimated Hours	Estimated Cost
Research	Industrial Engineer	40	2000€
Robot Design	3D Draftsman	104	2600€
Robot configuration	Industrial Engineer	64	3200€
Third Axis Design	3D Draftsman	48	1200€
Raspberry Pi Configuration	Industrial Engineer	104	5200€
Programming	Informatic Engineer	200	7000€
Wiring Design	Industrial Engineer	40	2000€
Documentation	Industrial Engineer	64	3200€
Testing	Industrial Engineer	40	2000€
	TOTAL =	**704h**	**28.400€**

7.2 Hardware Budget

Concept	Price	Qty	Cost
Festo Robot EXCM30	6.198€	1	6.198€
RaspBerry Pi 1 B+	25€	1	25€
Emergency Button	10€	1	10€
Cylinder	88,2€	1	88,2€
Electrovalve	135€	1	135€
Gripper 3D printed	50€	1	50€
Camera	22€	1	22€
1m2 of Metracrylate	35€	5	175€
USB Wifi antenne	10€	1	10€
USB Mouse	9€	1	9€
USB Keyboard	11€	1	11€
HDMI TV Monitor	105€	1	105€
LED	0,10€	10	1€
Servomotor 9g	8€	1	8€
VGA Connector	3€	1	3€
24v/2A Power Source	70€	1	70€
5v/1A Power Source	7€	1	7€
Optocoupler PC847	2€	3	6€
8gb SD Card	5€	1	5€
150ohm Resistence	0,015€	15	0,23€
10kohm Resistence	0,015€	10	0,15€
Wiring pack - 20pin	5€	1	5€
TOTAL =	**6.943,58€**		

7.3 Software Budget

Concept	Price	Licencia	Utilizado	Coste
Debian Jessie for Raspberry	0€	Free	N/A	0€
SD Formatter	0€	Free	N/A	0€
Python IDK	0€	Free	N/A	0€
Fritzing	0€	Free	N/A	0€
Telegram	0€	Free	N/A	0€
Solidworks	3995€	12 Months	4 Months	998,7€
Microsoft Office 2016	9,99€	1 Month	4 Months	39,96€
Festo FCT Software	0€	Free	N/A	0€
TOTAL	**1038,66€**			

7.4 Total Budget

Concept	Price
Manpower Budget	28.400,00€
Hardware Budget	6.943,58€
Software Budget	1.038,66€
TOTAL	**36.382,24€**

Chapter 8
Conclusions

In conclusion it is noteworthy that the project has been ambitious from the beginning and throughout the project has been checking that involved great technical difficulty, since it involves a multidisciplinary development that included part of programming, part design, part of mechanical and electronic part.

Mainly aim was to create a prototype which capture the idea and solution we wanted to do, and I think I accomplished successfull.

The value of the lessons learned from the implementation of this project is very large, as on countless occasions I have had to solve problems and find alternatives, in addition to do a good research on programming "Lua" as it had previously never used it.

Note that as time planning, I've had several problems, since the project covered more than I really expected, and added to my job situation I found it laborious although ultimately successful, anyway I learned to be better managing the time of the projects.

And research on the idea has been a success, as contrasted with medical personal working in a lab this type of procedures in addition to finding information in the internet

About Software design has finally been successful, and although it is very changeable and I have many ideas for improvement and debugging, has successfully passed the goal I intended.

On the hardware design has done all successfully and within our goals except the part joining the piston with the robot, and finally I had to give an alternative to other types of piston since I was unworkable go that my way into CAD design concepts.

In brief, I think the project has been generally satisfactory, because as I mentioned before, I have reflected and have managed to teach the idea and solution the idea and solution to the problem as I expected. Perhaps as a negative point, that the project has covered more than expected and could not make a final product and has had to leave as a product prototype

Chapter 9
Future Development

To change the future, I have many in mind that I want to share because I find interesting and that the final product would be much more robust and attractive

Organized from less difficult to high difficult of implementation

Database

A database where you can include all the memories and states of the robot so that it was more robust and could use that database for any platform and type of programming. Currently are local variables that can only read the Rapsberry.

Add permissions to contacts.

Implementing the system has Telegram available to give permission to those contacts to messages that the user sends, to be read by the raspberry. In brief, give permissions to users so that not everyone who "add" the robot can control it.

Implement security program and debugging of errors.

Implement a program to manage the time all the movements and processes provided that can be displayed for errors and debug as well.

Perform Mobile Application.

This would be to design a robot exclusive application of which shall contain buttons instead of typing commands as is currently done with a more intuitive interface for handling robot, as it would be more comfortable for the user.

Development of interaction with the aplications

Develop interaction with refrigerator, heater and the autoclave for that to further automate the process and keep control over them as main applications of a laboratory.

Chapter 10
Annexs

10.1 Main entery program

```
paso_in = 0
paso_start = 0
paso_petrinfo = 0
paso_preculture = 0
paso_postculture = 0
paso_petrihot = 0
paso_petricold = 0
paso_petripre = 0

posicion_petripre1 = 0
posicion_petripre2 = 0
posicion_petripre3 = 0
posicion_cold = 0
posicion_hot = 0
posicion_autoclave = 0

started = 0
our_id = 0
path = os.getenv("HOME").."/tg/scripts"
function vardump(value, depth, key)
  local linePrefix = ""
  local spaces = ""

  if key ~= nil then
    linePrefix = "["..key.."] = "
  end

  if depth == nil then
    depth = 0
  else
    depth = depth + 1
    for i=1, depth do spaces = spaces .. "  " end
  end

  if type(value) == 'table' then
    mTable = getmetatable(value)
    if mTable == nil then
```

```
      print(spaces ..linePrefix.."(table) ")
    else
      print(spaces .."(metatable) ")
        value = mTable
    end
    for tableKey, tableValue in pairs(value) do
      vardump(tableValue, depth, tableKey)
    end
  elseif type(value)    == 'function' or
      type(value)       == 'thread' or
      type(value)       == 'userdata' or
      value             == nil
  then
    print(spaces..tostring(value))
  else
    print(spaces..linePrefix.."("..type(value)..") "..tostring(value))
  end
end


function ok_cb(extra, success, result)
end

-- Notification code {{{

function get_title (P, Q)
  if (Q.type == 'user') then
    return P.first_name .. " " .. P.last_name
  elseif (Q.type == 'chat') then
    return Q.title
  elseif (Q.type == 'encr_chat') then
    return 'Secret chat with ' .. P.first_name .. ' ' .. P.last_name
  else
    return ''
  end
end

local lgi = require ('lgi')
local notify = lgi.require('Notify')
notify.init ("Telegram updates")
local icon = os.getenv("HOME") .. "/.telegram-cli/telegram-pics/telegram_64.png"

function do_notify (user, msg)
```

```
  local n = notify.Notification.new(user, msg, icon)
  n:show ()
end

-- }}}

local clock = os.clock

function sleep(n)
  local t0 = clock()
  while clock() - t0 < n do end
end

function on_msg_receive (msg)


  if started == 0 then
    return
  end
  if msg.out then
    return
  end
  do_notify (get_title (msg.from, msg.to), msg.text)


  if (msg.text == '/ping') then
    if (msg.to.id == our_id) then
      send_msg (msg.from.print_name, 'pong', ok_cb, false)
    else
      send_msg (msg.from.print_name, 'pong', ok_cb, false)
    end
    return
  end


  if (msg.text == '/takepic') then
    send_msg (msg.from.print_name, 'Sending pic..', ok_cb, false)
    os.execute('python '..path..'/hacerfoto.py')
    send_photo (msg.from.print_name, path..'/fotoplaca.jpg',ok_cb,false)
    return
  end
```

```
if (msg.text == '/start') then
  paso_in = 0
  paso_start = 0
  paso_petrinfo = 0
  paso_preculture = 0
  paso_postculture = 0
  paso_petrihot = 0
  paso_petricold = 0
  paso_petripre = 0
  os.execute('sudo python '..path..'/subrutinas.py start')
  send_msg (msg.from.print_name, 'The system is READY', ok_cb, false)
  send_msg (msg.from.print_name, 'Available Commands = [/in, /petrinfo,
   /systemstatus]', ok_cb, false)
  paso_start=1
  return
end


if (msg.text == '/reboott') then
  send_msg (msg.from.print_name, 'Rebooting the system...', ok_cb, false)
  os.execute('sudo python '..path..'/subrutinas.py reboot')
  return
end


if (msg.text == '/shutdownn') then
  send_msg (msg.from.print_name, 'Shutdown in 3..2..1..', ok_cb, false)
  os.execute('sudo python '..path..'/subrutinas.py shutdown')
  return
end

if (msg.text == '/in') then
  paso_in = 0
  paso_petrinfo = 0
  paso_preculture = 0
  paso_postculture = 0
  paso_petrihot = 0
  paso_petricold = 0
  paso_petripre = 0
   if (paso_start == 1 or paso_petripre == 1) then
        os.execute('sudo python '..path..'/subrutinas.py in')
        send_msg (msg.from.print_name, 'Got the new plate', ok_cb, false)
```

```
        send_msg (msg.from.print_name, 'Available Commands = [/preculture,
  /postculture]', ok_cb, false)
        paso_start=0
        paso_in=1
  elseif (paso_start == 0) then
        send_msg (msg.from.print_name, 'Command not allowed', ok_cb, false)
  end
  return
end


if (msg.text == '/preculture') then
  if (paso_in == 1) then
        if (posicion_petripre1==0) then
                os.execute('sudo python '..path..'/subrutinas.py preculture1')
                send_msg (msg.from.print_name, 'Saved the plate to the
  preculture zone 1', ok_cb, false)
                posicion_petripre1=1
                paso_in=0
                paso_start=1
        elseif (posicion_petripre2==0) then
                os.execute('sudo python '..path..'/subrutinas.py preculture2')
                send_msg (msg.from.print_name, 'Saved the plate to the
  preculture zone 2', ok_cb, false)
                posicion_petripre2=1
                paso_in=0
                paso_start=1
        elseif (posicion_petripre3==0) then
                os.execute('sudo python '..path..'/subrutinas.py preculture3')
                send_msg (msg.from.print_name, 'Saved the plate to the
  preculture zone 3', ok_cb, false)
                posicion_petripre3=1
                paso_in=0
                paso_start=1
        else
                send_msg (msg.from.print_name, 'The preculture spots are full',
  ok_cb, false)
                os.execute('sudo python '..path..'/subrutinas.py out')
                send_msg (msg.from.print_name, 'Left the plate in the departure
  zone', ok_cb, false)
                paso_in=0
                paso_start=1
        end
```

```
  elseif (paso_in==0) then
        send_msg (msg.from.print_name, 'Command not allowed', ok_cb, false)
  end
 return
end


if (msg.text == '/postculture') then
  if (paso_in==1) then
       send_msg (msg.from.print_name, 'You have some options for a post
  culture plate process', ok_cb, false)
       send_msg (msg.from.print_name, 'Available Commands = [/savecold,
  /savehot, /autoclave]', ok_cb, false)
       paso_in=0
       paso_postculture=1
  elseif (paso_in==0) then
       send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
  end
 return
end


if (msg.text == '/savehot') then
    if (paso_postculture==1) then
      if (posicion_hot==0) then
        posicion_hot=1
        os.execute('sudo python '..path..'/subrutinas.py savehot')
        send_msg (msg.from.print_name, 'Saved the plate to the hot spot',
  ok_cb, false)
        paso_postculture=0
       paso_start=1
      elseif (posicion_hot==1) then
        send_msg (msg.from.print_name, 'The preculture spots are full', ok_cb,
  false)
       os.execute('sudo python '..path..'/subrutinas.py out')
       send_msg (msg.from.print_name, 'Left the plate in the departure zone',
  ok_cb, false)
       paso_postculture=0
       paso_start=1
     end
    elseif (paso_postculture==0) then
      send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
    end
```

```
end


if (msg.text == '/savecold') then
    if (paso_postculture==1) then
        if (posicion_cold==0) then
            os.execute('sudo python '..path..'/subrutinas.py savecold')
            send_msg (msg.from.print_name, 'Saved the plate to the cold spot',
  ok_cb, false)
            posicion_cold=1
            paso_postculture=0
        paso_start=1
        elseif (posicion_cold==1) then
            send_msg (msg.from.print_name, 'The preculture spots are full', ok_cb,
  false)
        os.execute('sudo python '..path..'/subrutinas.py out')
        send_msg (msg.from.print_name, 'Left the plate in the departure zone',
  ok_cb, false)
        paso_postculture=0
        paso_start=1
     end
    elseif (paso_postculture==0) then
        send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
    end
 return
end


if (msg.text == '/autoclave') then
    if (paso_postculture==1) then
        if (posicion_autoclave==0) then
            os.execute('sudo python '..path..'/subrutinas.py autoclave')
            send_msg (msg.from.print_name, 'Moved the plate to the autoclave',
  ok_cb, false)
            posicion_autoclave=1
            paso_postculture=0
        paso_start=1
        elseif (posicion_autoclave==1) then
            send_msg (msg.from.print_name, 'The preculture spots are full', ok_cb,
  false)
        os.execute('sudo python '..path..'/subrutinas.py out')
        send_msg (msg.from.print_name, 'Left the plate in the departure zone',
  ok_cb, false)
```

```
            paso_postculture=0
            paso_start=1
         end
       elseif (paso_postculture==0) then
          send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
       end
    return
end


if (msg.text == '/petrinfo') then
   if (paso_start == 1) then
          os.execute('sudo python '..path..'/subrutinas.py petrinfo')
          send_msg (msg.from.print_name, 'Which plate are you asking for?',
   ok_cb, false)
          send_msg (msg.from.print_name, 'Available Commands = [/petrihot,
   /petricold, /petripre]', ok_cb, false)
          paso_start=0
          paso_petrinfo=1
   elseif (paso_start==0) then
          send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
   end
  return
end


if (msg.text == '/petrihot') then
   if (paso_petrinfo == 1) then
          if (posicion_hot==1) then
             os.execute('sudo python '..path..'/subrutinas.py petrihot')
             send_msg (msg.from.print_name, 'Sent pic to confirm it', ok_cb, false)
             os.execute('python '..path..'/hacerfoto.py')
             send_photo (msg.from.print_name, path..'/fotoplaca.jpg',ok_cb,false)
             send_msg (msg.from.print_name, 'Available Commands = [/out, /in]',
   ok_cb, false)
             paso_petrinfo=0
             posicion_hot=0
             paso_petrihot=1
          paso_start=1
          elseif (posicion_hot==0) then
             send_msg (msg.from.print_name, 'The hot spot is empty', ok_cb, false)
             paso_petrinfo=1
          end
```

```
    elseif (paso_petrinfo==0) then
        send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
  end
 return
end


if (msg.text == '/petricold') then
  if (paso_petrinfo == 1) then
      if (posicion_cold==1) then
         os.execute('sudo python '..path..'/subrutinas.py petricold')
         send_msg (msg.from.print_name, 'Sent pic to confirm it', ok_cb, false)
         os.execute('python '..path..'/hacerfoto.py')
         send_photo (msg.from.print_name, path..'/fotoplaca.jpg',ok_cb,false)
         send_msg (msg.from.print_name, 'Available Commands = [/out, /in]',
  ok_cb, false)
         paso_petrinfo=0
         posicion_cold=0
         paso_petricold=1
       paso_start=1
      elseif (posicion_cold==0) then
         send_msg (msg.from.print_name, 'The cold spot is empty', ok_cb,
  false)
         paso_petrinfo=1
      end
    elseif (paso_petrinfo==0) then
       send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
  end
 return
end




if (msg.text == '/petripre') then
   if (paso_petrinfo==1) then
      send_msg (msg.from.print_name, 'Which preculture petri plate do you
  want?', ok_cb, false)
      send_msg (msg.from.print_name, 'Available Commands = [/1, /2, /3]',
  ok_cb, false)
      paso_petripre=1
   elseif (paso_petrinfo==0) then
```

```
      send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
    end
  return
end


if (msg.text == '/1') then
    if (paso_petripre==1) then
       if (posicion_petripre1==1) then
            os.execute('sudo python '..path..'/subrutinas.py petripre1')
            send_msg (msg.from.print_name, 'Sent pic to confirm it', ok_cb, false)
            os.execute('python '..path..'/hacerfoto.py')
            send_photo (msg.from.print_name, path..'/fotoplaca.jpg',ok_cb,false)
            send_msg (msg.from.print_name, 'Available Commands = [/out, /in]',
   ok_cb, false)
            posicion_petripre1=0
          paso_start=1
            paso_petrinfo=0
       elseif (posicion_petripre1==0) then
            send_msg (msg.from.print_name, 'The preculture spot number 1 is
   empty', ok_cb, false)
            send_msg (msg.from.print_name, 'Available Commands = [/2, /3]',
   ok_cb, false)
          paso_start=1
            paso_petrinfo=0
       end
   elseif (paso_petripre==0) then
      send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
   end
  return
end


if (msg.text == '/2') then
    if (paso_petripre==1) then
       if (posicion_petripre2==1) then
            os.execute('sudo python '..path..'/subrutinas.py petripre2')
            send_msg (msg.from.print_name, 'Sent pic to confirm it', ok_cb, false)
            os.execute('python '..path..'/hacerfoto.py')
            send_photo (msg.from.print_name, path..'/fotoplaca.jpg',ok_cb,false)
            send_msg (msg.from.print_name, 'Available Commands = [/out, /in]',
   ok_cb, false)
```

```
            posicion_petripre2=0
          paso_start=1
       elseif (posicion_petripre2==0) then
            send_msg (msg.from.print_name, 'The preculture spot number 2 is
   empty', ok_cb, false)
            send_msg (msg.from.print_name, 'Available Commands = [/1, /3]',
   ok_cb, false)
          paso_start=1
            paso_petrinfo=0
       end
   elseif (paso_petripre==0) then
      send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
   end
  return
end


if (msg.text == '/3') then
    if (paso_petripre==1) then
       if (posicion_petripre3==1) then
            os.execute('sudo python '..path..'/subrutinas.py petripre3')
            send_msg (msg.from.print_name, 'Sent pic to confirm it', ok_cb, false)
            os.execute('python '..path..'/hacerfoto.py')
            send_photo (msg.from.print_name, path..'/fotoplaca.jpg',ok_cb,false)
            send_msg (msg.from.print_name, 'Available Commands = [/out, /in]',
   ok_cb, false)
            posicion_petripre3=0
          paso_start=1
       elseif (posicion_petripre3==0) then
            send_msg (msg.from.print_name, 'The preculture spot number 3 is
   empty', ok_cb, false)
            send_msg (msg.from.print_name, 'Available Commands = [/1, /2]',
   ok_cb, false)
          paso_start=1
            paso_petrinfo=0
       end
   elseif (paso_petripre==0) then
      send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
   end
  return
end
```

```
if (msg.text == '/out') then
   if (paso_petripre==1 or paso_petricold==1 or paso_petrihot==1) then
        os.execute('sudo python '..path..'/subrutinas.py out')
        send_msg (msg.from.print_name, 'Left the plate in the departure zone',
  ok_cb, false)
        paso_petripre=0
        paso_petricold=0
        paso_petrihot=0
      paso_start=1
        paso_petrinfo=0
   else
        send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
   end
 return
end


if (msg.text == '/systemstatus') then
   if (paso_start==1) then
        if (posicion_cold==1) then
                send_msg (msg.from.print_name, '[HEATER]: Saved since #Date
  at #Hour', ok_cb, false)
        elseif (posicion_cold==0) then
                send_msg (msg.from.print_name, '[HEATER]: Empty', ok_cb,
  false)
        end
        if (posicion_hot==1) then
                send_msg (msg.from.print_name, '[REFRIGERATOR]: Saved
  since #Date at #Hour', ok_cb, false)
        elseif (posicion_hot==0) then
                send_msg (msg.from.print_name, '[REFRIGERATOR]: Empty',
  ok_cb, false)
        end
        if (posicion_autoclave==1) then
                send_msg (msg.from.print_name, '[AUTOCLAVE]: Being Used',
  ok_cb, false)
        elseif (posicion_autoclave==0) then
                send_msg (msg.from.print_name, '[AUTOCLAVE]: Not Used',
  ok_cb, false)
        end
        if (posicion_petripre1==0 and posicion_petripre2==0 and
  posicion_petripre3==0) then
```

```
                send_msg (msg.from.print_name, '[PRECULTURE]: 3 Empty
    Spots', ok_cb, false)
        elseif (posicion_petripre1==0 and posicion_petripre2==0 and
    posicion_petripre3==1) then
                send_msg (msg.from.print_name, '[PRECULTURE]: 2 Empty
    Spots', ok_cb, false)
        elseif (posicion_petripre1==0 and posicion_petripre2==1 and
    posicion_petripre3==1) then
                send_msg (msg.from.print_name, '[PRECULTURE]: 1 Empty
    Spot', ok_cb, false)
        elseif (posicion_petripre1==0 and posicion_petripre2==1 and
    posicion_petripre3==0) then
                send_msg (msg.from.print_name, '[PRECULTURE]: 2 Empty
    Spots', ok_cb, false)
        elseif (posicion_petripre1==1 and posicion_petripre2==0 and
    posicion_petripre3==0) then
                send_msg (msg.from.print_name, '[PRECULTURE]: 2 Empty
    Spots', ok_cb, false)
        elseif (posicion_petripre1==1 and posicion_petripre2==1 and
    posicion_petripre3==0) then
                send_msg (msg.from.print_name, '[PRECULTURE]: 1 Empty
    Spots', ok_cb, false)
        elseif (posicion_petripre1==1 and posicion_petripre2==1 and
    posicion_petripre3==1) then
                send_msg (msg.from.print_name, '[PRECULTURE]: Full', ok_cb,
    false)
        end

     else
          send_msg (msg.from.print_name, 'Command not allowed ', ok_cb, false)
     end
   return
  end

end

function on_our_id (id)
  our_id = id
end

function on_user_update (user, what)
  --vardump (user)
end
```

```
function on_chat_update (chat, what)
  --vardump (chat)
end

function on_secret_chat_update (schat, what)
  --vardump (schat)
end

function on_get_difference_end ()
end

function cron()
  -- do something
  postpone (cron, false, 1.0)
end

function on_binlog_replay_end ()
  started = 1
  postpone (cron, false, 1.0)
end
```

10.2 Subroutines entery prgram

```
#!/usr/bin/python

import RPi.GPIO as GPIO
import sys
import time
import os

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

#declaracion de pins
startbutton= 7 #Inicializar sistema
electrovalve= 11 #electrovalvula piston
motioncompletebutton= 12 #Boton emergencia
servomotor= 13 #pulso servomotor pinza

startphrase= 16 #Necesario antes de cada envio de frase
robotenable= 18 #Desbloquea el robot para que se pueda mover
robotreset= 22 #Resetea si ha habido fallo

phrasepin2= 29
phrasepin3= 31
phrasepin4= 33
phrasepin5= 35
phrasepin6= 37

infopin11= 32 #Informa que el robot ha sido desbloqueado
infopin12= 36 #Informa que ha habido un fallo
infopin13= 38 #Acknoledge
infopin14= 40 #Secuencia completada

GPIO.setup(startbutton,GPIO.IN)
GPIO.setup(electrovalve,GPIO.OUT)
GPIO.setup(motioncompletebutton,GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(servomotor,GPIO.OUT)
GPIO.setup(startphrase,GPIO.OUT)
GPIO.setup(robotenable,GPIO.OUT)
GPIO.setup(robotreset,GPIO.OUT)
GPIO.setup(phrasepin2,GPIO.OUT)
GPIO.setup(phrasepin3,GPIO.OUT)
GPIO.setup(phrasepin4,GPIO.OUT)
GPIO.setup(phrasepin5,GPIO.OUT)
GPIO.setup(phrasepin6,GPIO.OUT)
GPIO.setup(infopin11,GPIO.IN)
GPIO.setup(infopin12,GPIO.IN)
GPIO.setup(infopin13,GPIO.IN)
GPIO.setup(infopin14,GPIO.IN)
```

```
    #declaracion de comandos subrutinas
    commandlist=['start', 'reboott','shutdownn','in','preculture', 'postculture', 'petrinfo',
'savehot', 'savecold', 'autoclave', 'petrihot',
'petricold','petripre','out','petripre1','petripre2','petripre3']

    #Error si el comando es desconocido
    if len(sys.argv) < 2:
        print "Unrecognized command, try again"
    elif sys.argv[1] not in commandlist:
        print "Unrecognized command, try again"

    #subrutinas en funcion del comando
    for eachArg in sys.argv:
        if eachArg == 'start':
            GPIO.output(electrovalve,0)
            GPIO.output(servomotor,0)
            GPIO.output(robotenable,1)
            GPIO.output(robotreset,1)
            time.sleep(3)
            GPIO.output(phrasepin2,1)
            GPIO.output(phrasepin3,1)
            GPIO.output(phrasepin4,1)
            GPIO.output(phrasepin5,1)
            GPIO.output(phrasepin6,1)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(robotenable,0)
            GPIO.output(robotreset,0)
            GPIO.output(startphrase,0)
            time.sleep(3)


        elif eachArg == 'reboott':
            os.system('sudo shutdown -r now')


        elif eachArg == 'shutdownn':
            os.system('sudo shutdown -h 0.5')

        elif eachArg == 'in':
            GPIO.output(electrovalve,0)
            GPIO.output(robotenable,1)
            GPIO.output(robotreset,1)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,1)
            GPIO.output(phrasepin4,0)
            GPIO.output(phrasepin5,0)
            GPIO.output(phrasepin6,0)
```

```
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,1)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        GPIO.output(startphrase,0)
        time.sleep(3)


    elif eachArg == 'preculture1':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
```

```
        GPIO.output(electrovalve,0)
        time.sleep(3)


    elif eachArg == 'preculture2':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,1)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)


    elif eachArg == 'preculture3':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,1)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
```

```
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,1)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)



    elif eachArg == 'savehot':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        GPIO.output(electrovalve,0)
```

```
        time.sleep(3)

    elif eachArg == 'savecold':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)

    elif eachArg == 'autoclave':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
```

```
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)

    elif eachArg == 'petrihot':
        GPIO.output(electrovalve,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,0)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
```

```
            GPIO.output(startphrase,0)
            GPIO.output(electrovalve,0)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,0)
            GPIO.output(phrasepin4,1)
            GPIO.output(phrasepin5,0)
            GPIO.output(phrasepin6,0)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(electrovalve,1)
            GPIO.output(robotenable,0)
            GPIO.output(robotreset,0)
            time.sleep(3)

        elif eachArg == 'petricold':
            GPIO.output(electrovalve,0)
            GPIO.output(servomotor,0)
            GPIO.output(robotenable,1)
            GPIO.output(robotreset,1)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,0)
            GPIO.output(phrasepin4,0)
            GPIO.output(phrasepin5,1)
            GPIO.output(phrasepin6,0)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(electrovalve,1)
            time.sleep(3)
            GPIO.output(phrasepin2,1)
            GPIO.output(phrasepin3,1)
            GPIO.output(phrasepin4,1)
            GPIO.output(phrasepin5,0)
            GPIO.output(phrasepin6,0)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(servomotor,1)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,0)
            GPIO.output(phrasepin4,0)
            GPIO.output(phrasepin5,1)
            GPIO.output(phrasepin6,0)
```

```
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(electrovalve,0)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,0)
            GPIO.output(phrasepin4,1)
            GPIO.output(phrasepin5,0)
            GPIO.output(phrasepin6,0)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(electrovalve,1)
            GPIO.output(robotenable,0)
            GPIO.output(robotreset,0)
            time.sleep(3)

        elif eachArg == 'petripre1':
            GPIO.output(electrovalve,0)
            GPIO.output(servomotor,0)
            GPIO.output(robotenable,1)
            GPIO.output(robotreset,1)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,1)
            GPIO.output(phrasepin4,1)
            GPIO.output(phrasepin5,1)
            GPIO.output(phrasepin6,0)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(electrovalve,1)
            time.sleep(3)
            GPIO.output(phrasepin2,1)
            GPIO.output(phrasepin3,0)
            GPIO.output(phrasepin4,1)
            GPIO.output(phrasepin5,1)
            GPIO.output(phrasepin6,0)
            time.sleep(3)
            GPIO.output(startphrase,1)
            GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
            GPIO.output(startphrase,0)
            GPIO.output(servomotor,1)
            time.sleep(3)
            GPIO.output(phrasepin2,0)
            GPIO.output(phrasepin3,1)
```

```
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        time.sleep(3)


    elif eachArg == 'petripre2':
        GPIO.output(electrovalve,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
```

```
        GPIO.output(servomotor,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        time.sleep(3)


    elif eachArg == 'petripre3':
        GPIO.output(electrovalve,0)
        GPIO.output(servomotor,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
```

```
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(servomotor,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,1)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,0)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        GPIO.output(robotenable,0)
        GPIO.output(robotreset,0)
        time.sleep(3)

    elif eachArg == 'out':
        GPIO.output(electrovalve,0)
        GPIO.output(robotenable,1)
        GPIO.output(robotreset,1)
        time.sleep(3)
        GPIO.output(phrasepin2,0)
        GPIO.output(phrasepin3,1)
        GPIO.output(phrasepin4,1)
        GPIO.output(phrasepin5,0)
        GPIO.output(phrasepin6,0)
        time.sleep(3)
        GPIO.output(startphrase,1)
        GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
        GPIO.output(startphrase,0)
        GPIO.output(electrovalve,1)
        time.sleep(3)
        GPIO.output(phrasepin2,1)
        GPIO.output(phrasepin3,0)
        GPIO.output(phrasepin4,1)
```

```
GPIO.output(phrasepin5,0)
GPIO.output(phrasepin6,0)
time.sleep(3)
GPIO.output(startphrase,1)
GPIO.wait_for_edge(motioncompletebutton, GPIO.BOTH)
GPIO.output(startphrase,0)
GPIO.output(servomotor,0)
time.sleep(3)
GPIO.output(electrovalve,0)
GPIO.output(robotenable,0)
GPIO.output(robotreset,0)
time.sleep(3)
```

10.3 Take a pic program

```
import time
import picamera
import os

path=os.getenv("HOME")+"/tg/scripts"
with picamera.PiCamera() as picam:
    picam.rotation=90
    picam.start_preview()
    picam.capture(path+'/fotoplaca.jpg',resize=(640,480))
    time.sleep(2)
    picam.stop_preview()
    picam.close()
```

Bibliografy

FLORES J. LAS PLACAS PETRI, EL GRAN INVENTO DE LA MICROBIOLOGÍA [INTERNET]. MUYINTERESANTE.ES. 2016. AVAILABLE FROM: HTTP://WWW.MUYINTERESANTE.ES/CIENCIA/ARTICULO/MICROBIOLOGIA-HOMENAJE-DE-GOOGLE-AL-CREADOR-DE-LAS-PLACAS-DE-PETRI-401369996467

BACTERIA GROWING EXPERIMENTS IN PETRI PLATES [INTERNET]. SCIENCECOMPANY.COM. 2016. AVAILABLE FROM: HTTP://WWW.SCIENCECOMPANY.COM/BACTERIA-GROWING-EXPERIMENTS-IN-PETRI-PLATES.ASPX

RASPBERRY PI - TEACH, LEARN, AND MAKE WITH RASPBERRY PI [INTERNET]. RASPBERRY PI. 2016. AVAILABLE FROM: HTTPS://WWW.RASPBERRYPI.ORG/

TELEGRAM – A NEW ERA OF MESSAGING [INTERNET]. TELEGRAM. 2016. AVAILABLE FROM: HTTPS://TELEGRAM.ORG/

VYSHENG/TG [INTERNET]. GITHUB. 2016. AVAILABLE FROM: https://github.com/vysheng/tg

FESTO ESPAÑA [INTERNET]. FESTO.COM. 2016. AVAILABLE FROM: HTTPS://WWW.FESTO.COM/CMS/ES_ES/INDEX.HTM

PIGPIO LIBRARY [INTERNET]. ABYZ.CO.UK. 2016. AVAILABLE FROM: HTTP://ABYZ.CO.UK/RPI/PIGPIO/PYTHON.HTML

RBNRPI [INTERNET]. RBNRPI. 2016. AVAILABLE FROM: HTTPS://RBNRPI.WORDPRESS.COM/

RASPBERRY PI - ARCHWIKI [INTERNET]. WIKI.ARCHLINUX.ORG. 2016 . AVAILABLE FROM: HTTPS://WIKI.ARCHLINUX.ORG/INDEX.PHP/RASPBERRY_PI

ARIZAGA M. CONFIGURANDO UN ORDENADOR PARA UTILIZAR SU TECLADO Y PANTALLA PARA MANEJAR LA RASPBERRY PI [INTERNET]. MARKEL ARIZAGA. 2013 AVAILABLE FROM: HTTPS://MARKELARIZAGA.WORDPRESS.COM/2013/10/18/CONFIGURANDO-UN-ORDENADOR-PARA-UTILIZAR-SU-TECLADO-Y-PANTALLA-PARA-MANEJAR-LA-RASPBERRY-PI/

TIENDA DE ELECTRÓNICA ONLINE. VENTA DE COMPONENTES ELECTRÓNICOS E INFORMÁTICOS AL POR MAYOR [INTERNET]. DIOTRONIC. 2016. AVAILABLE FROM: HTTP://WWW.DIOTRONIC.COM/

THE PYTHON STANDARD LIBRARY — PYTHON 2.7.12 DOCUMENTATION [INTERNET]. DOCS.PYTHON.ORG. 2016. AVAILABLE FROM: HTTPS://DOCS.PYTHON.ORG/2/LIBRARY/

LUA-USERS WIKI: LIBRARIES AND BINDINGS [INTERNET]. LUA-USERS.ORG. 2016 AVAILABLE FROM: HTTP://LUA-USERS.ORG/WIKI/LIBRARIESANDBINDINGS

PYTHONHOSTED.ORG. (2016). WELCOME TO RPIO'S DOCUMENTATION! — RPIO 0.10.0 DOCUMENTATION. [ONLINE] AVAILABLE AT: HTTPS://PYTHONHOSTED.ORG/RPIO/