Tecnológico de Monterrey

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MÉXICO

Design Report: "Drunk Mixer"

Teacher: Alf Kjartan Halvorsen

Subject: Robotics Project

Team D:

Luis Andrés Medina Calderón A01379628
Leonardo Valencia Benítez A01378568
Aldo Fuentes Mendoza A01373294
Enrique Romero Vazquez A01373298

Index

I. Problem Definition……...……..4

II. What is "Drunk Mixer"?……...4

III. Design Requirements ……4

A. Functional Requirements ……4
B. Performance Requirements ……..5

IV. Mechanical design……...……..5

A. Robotic arms and Smart Gripper ……5
1. CAD ……6
2. Drawings ……..9
B. Dispenser System ……....11
1. CAD ……...……...12
2. Drawings. ……...16
C. Conveyor belt ……...18
1. CAD ……...18
2. Drawing.……19
D. Rack of Glasses ……...20
1. CAD ……...20
2. Drawing……..21

V. Electrical design ……...22

A. Solenoid Electro Valve ……..22
1. CAD ……...22
2. Drawing.……23
B. Circuit ……24
C. Components description ……..26
D. Requirements covered for Electrical Design ……..27
E. Sensors used in RobotStudio ……...27
F. Actuators ……..28

VI. Software……..28

A. UML Diagram……....31
B. Web Page ……...32
C. Server ……33

VII. Control Design……..33

A. Flow Chart …...……..33
B. Rapid (Robot Studio) ……37
C. Station Logic ……...37

VIII. Performance Tests ……..40

A. Importing CAD's designs to Robot Studio ……40
B. Simulating electrical components on Proteus ……....41
C. Web application and the Server ……..43
D. Simulation RobotStudio ……48
E. Time Performance.……52

IV. References ……...54

I. Problem Definition

The world's current situation has limited human interaction in every aspect of our lives. Although humans have dealt with different pandemics before, modern technology has made it easier for people to continue with their everyday activities. Wireless connections and automated services such as grocery shopping without a cashier, car wash, food services, among others help stop the spread of the virus.Taking this into consideration, this project aims to solve the need of a bartender to serve simple drinks. This way the risk of spreading the virus through the glasses is reduced. Also, this project will increment efficiency and reduce time for serving beverages, as well as reducing the cost of having an extra bartender behind the bar. The machine would only need maintenance every amount of time.

It is often a problem that bartenders get wrong orders or spill a considerable amount of liquid while serving large amounts of customers. With this project we are avoiding any annoying confusion, this machine will always get their order right and there will not be any liquid wasted.

Nevertheless, this device does not reduce human interaction completely since it still needs someone to refill the supplies such as the soft drinks, the alcohol and cups/glasses, as well as doing maintenance. Making a machine like this means implementing the infrastructure right on the bar, which costs money and requires appropriate planning. Finally, while it may solve many issues it may raise different problems such as differentiating which drink belongs to which customer, the machine cannot know this. Also, since the amount of liquid per glass will be predetermined, the customer is unable to ask for an extra shot or a reduced amount of something. If it were capable of doing so, it may cause further problems such as a decrease in the amount of beverages expected per bottle.

II. What is "Drunk Mixer"?

It is an embedded system capable of serving drinks in a bar. The client will make the order through a website which is connected to the system. It has 7 containers, one for ice, 3 for alcohol and 3 for sodas. The mixer is able to make 9 different alcoholic drinks while keeping track of how many drinks it has served and how much liquid is left to make them.

III. Design Requirements

A. Functional Requirements

1. The system shall be able to access our web server and attend the clients' requests.
2. The system shall have an interface to manually select the drink.
3. The system shall store the alcohol and mixer needed for the drinks.

4. The system shall be able to identify if it has enough ingredients to prepare the selected drink.
    a. If it does, start the process shown below.
    b. If it doesn't send a signal and display the ingredients level.
5. The system shall be able to take a glass from a special rack and place it on the start of the preparing sub-system.
6. The system should be able to transport the filled glasses without spilling the drink or breaking the glass.
7. The system shall be able to stop in the right position according to the liquid needed.
8. The system shall be able to pour the exact amount of alcohol and mixer needed for each beverage.
9. The system shall be able to identify when to start the next drink after finishing one.
10. The system shall be able to pick up the finished drink and transport it to the bar.
11. The system shall create a report of the served drinks.

B. Performance Requirements

1. The system shall be capable of serving the required drink in 50 seconds or less.
2. The system shall be able to serve 9 different beverages.
3. The system's server shall be able to receive up to 60 customers' requests per queue.
4. The system shall allow up to 1 customer at a time to use the interface and order as many drinks as available product there is.
5. The system's glass rack shall be able to contain up to 12 glasses.
6. The system's first arm shall be able to take one glass at a time and place it on the start of the preparing sub-system.
7. The system's second arm shall be able to take one glass out of the preparing sub-system and place it on the bar.
8. The system's controller shall handle the process for 1 drink at a time.
9. The preparing sub-system shall have 3 different positions for the 3 different steps of the processes.
10. The system shall pour 100ml of soda and 40ml of alcohol into the glass for any required beverage.
11. The system shall count the number of requests for each drink made in the whole day.

IV. Mechanical design

A. Robotic arms and Smart Gripper

The following diagrams and designs were obtained from ABB website. The system implements two robotic arms, specifically the IRB 120 from Robot Studio's library. According to the design requirements (A.5 and A.10) the system shall take the glass from a rack of glasses and place it on the conveyor belt, as well as move the final drink to the bar so the customer may pick it up. Both of these robotic arms along with the smart gripper on each one complies with this functionality. According to the performance requirements (B.6) the first arm shall take one glass at a time and place it on the start of the preparing sub-system and it is with the smart gripper that this is able to be done. The second arm must take the finished drink out of the process and place it on the bar (B.7). The degrees of freedom in each robotic arm allows the necessary movements for the system to comply with every requirement it was intended for. It can rotate almost 360 degrees from the base, 180º for the second joint, 90º for the third one and the tool can open up to 80mm and close all the way. In section VI of the Design Requirements Document, it was stated that the gripper would open up to 10 cm, but this was unnecessary. Since the diameter of the glass is approximately 6cm, the smart gripper can pick up the glass with no problem.

Although this robotic arm has six degrees of freedom, more than needed, it was selected because Robot Studio contains this arm with all its libraries already configured and the movements defined. Since the system is going to be simulated this one was the best option, although if the system were to be built, the robotic arm could be smaller and with only three degrees of freedom will be enough.

1. CAD

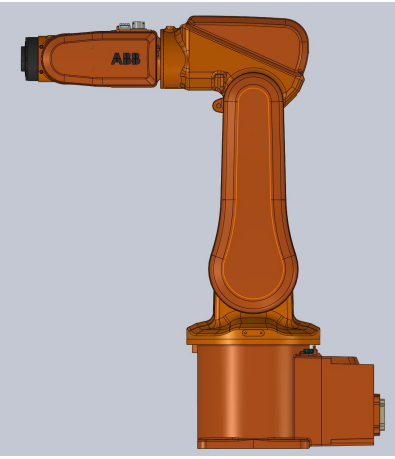


Fig. 1. ABB Robotic Arm model IRB 120 (CAD-Side View).[1]

[1] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

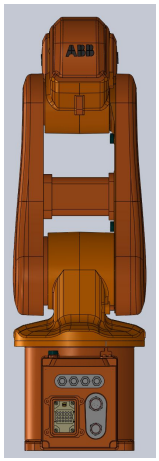


Fig. 2. ABB Robotic Arm model IRB 120 (CAD - Back View).[2]

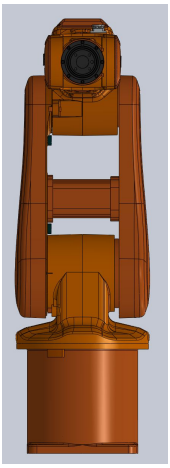


Fig. 3. ABB Robotic Arm model IRB 120 (CAD - Front View).[3]

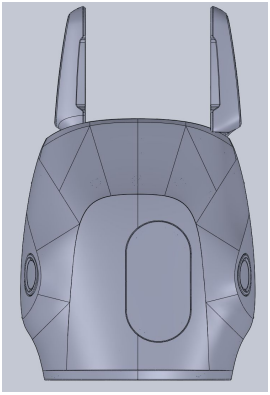


Fig. 4. ABB Smart Gripper (CAD - Upper View).[4]

---

[2] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

[3] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

[4] IRB 14000 YuMi Data - ABB's Collaborative Robot -YuMi (Industrial Robots from ABB Robotics). (2020). Retrieved 1 November 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data

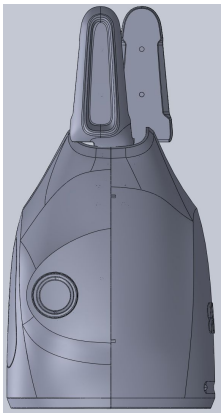


Fig. 5. ABB Smart Gripper (CAD - Side View).[5]

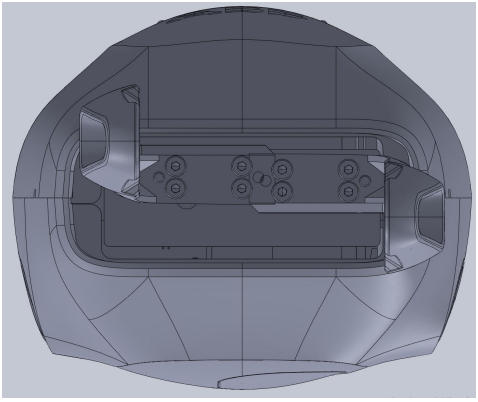


Fig. 6. ABB Smart Gripper (CAD - Front View).[6]

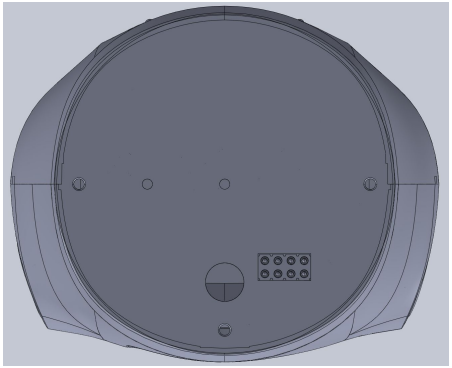


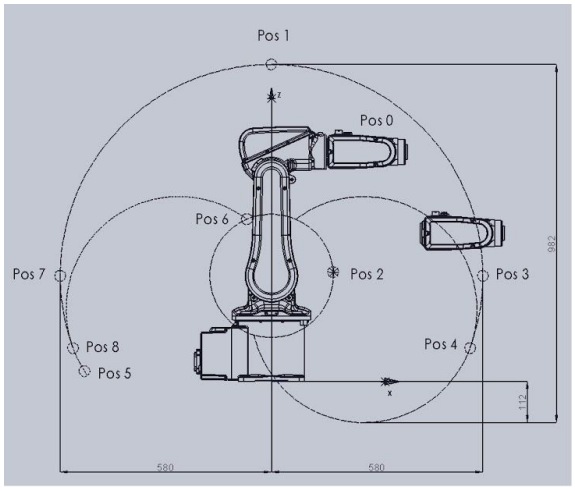
Fig. 7. ABB Smart Gripper (CAD - Bottom View).[7]

[5] IRB 14000 YuMi Data - ABB's Collaborative Robot -YuMi (Industrial Robots from ABB Robotics). (2020). Retrieved 1 November 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data

[6] IRB 14000 YuMi Data - ABB's Collaborative Robot -YuMi (Industrial Robots from ABB Robotics). (2020). Retrieved 1 November 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data

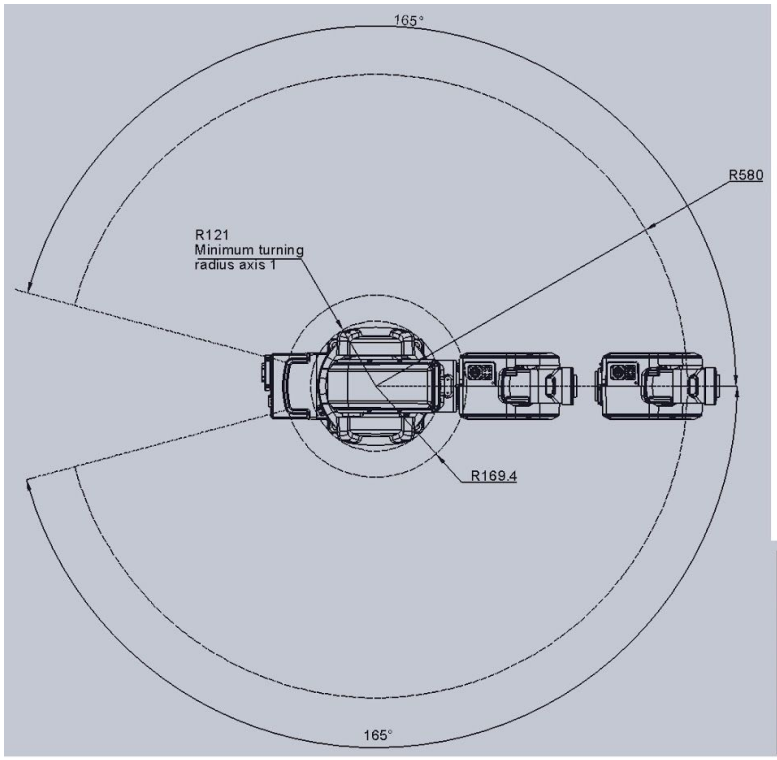
[7] IRB 14000 YuMi Data - ABB's Collaborative Robot -YuMi (Industrial Robots from ABB Robotics). (2020). Retrieved 1 November 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data

2. Drawings



Pos	Position at wrist center (mm)		Angle (degrees)	
	X	Z	axis 2	axis 3
0	302	630	0°	0°
1	0	870	0°	-76.9°
2	169	300	0°	+70.0°
3	580	270	+90°	-76.9°
4	545	91	+110°	-76.9°
5	-513	28	-110°	-90.0°
6	-67	445	-110°	+70.0°
7	-580	270	-90°	-76.9°
8	-545	91	-110°	-76.9°

Fig. 8. ABB Robotic Arm model IRB 120 (Drawing - Degrees of Freedom).[8]



Working range		
Ax.1	+165°	-165°
Ax.2	+110°	-110°
Ax.3	+70°	-90°
Ax.4	+160°	-160°
Ax.5	+120°	-120°
Ax.6	+400°	-400°

Fig. 9. ABB Robotic Arm model IRB 120 (Drawing - Degrees of Freedom 2).[9]

[8] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

[9] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

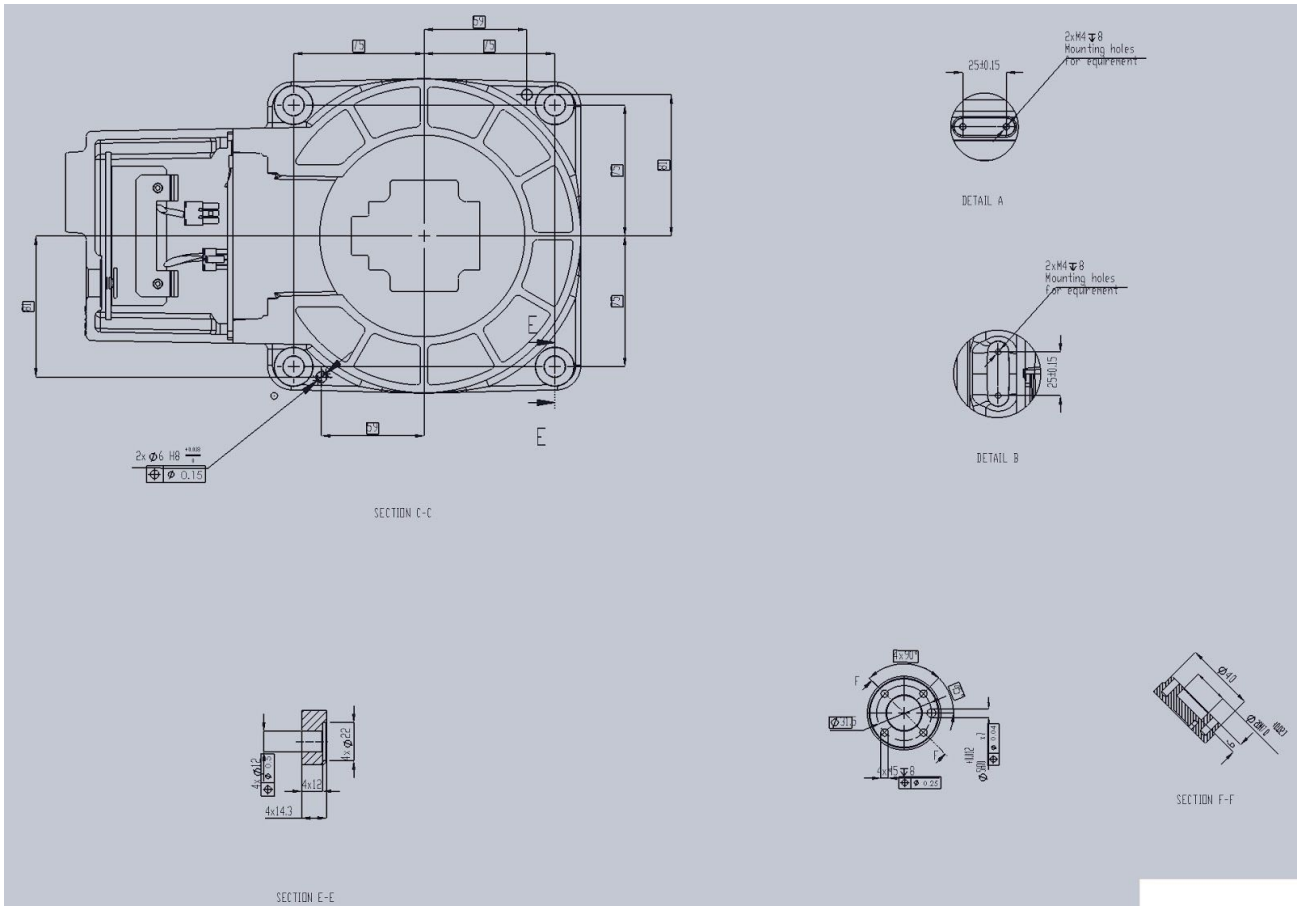


Fig. 10. ABB Robotic Arm model IRB 120 (Drawing - Base).[10]

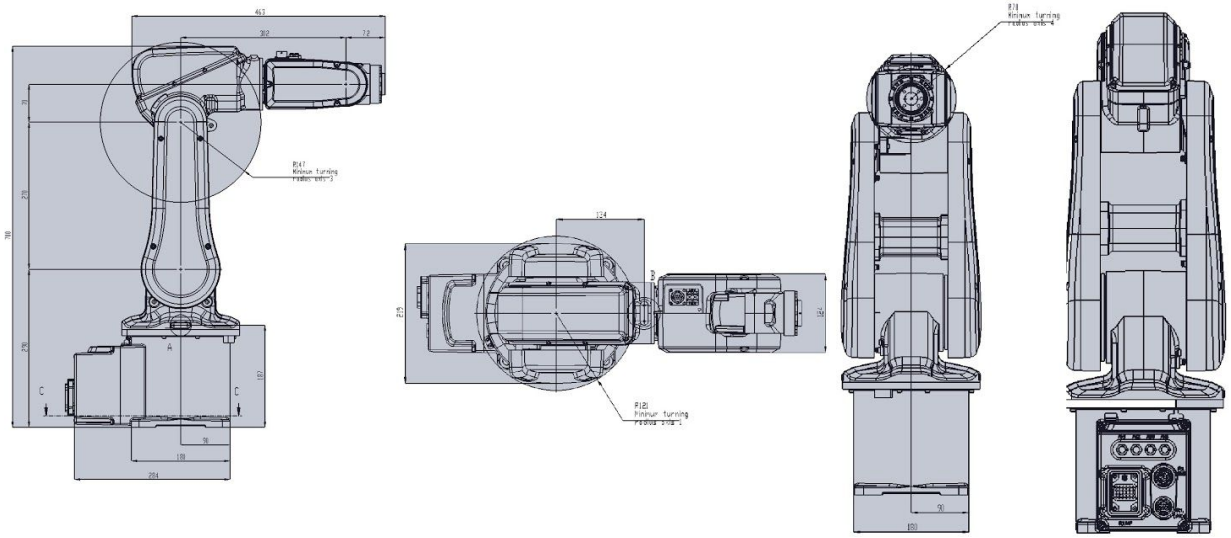


Fig. 11. ABB Robotic Arm model IRB 120 (Drawing - Dimensions).[11]

[10] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

[11] IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad

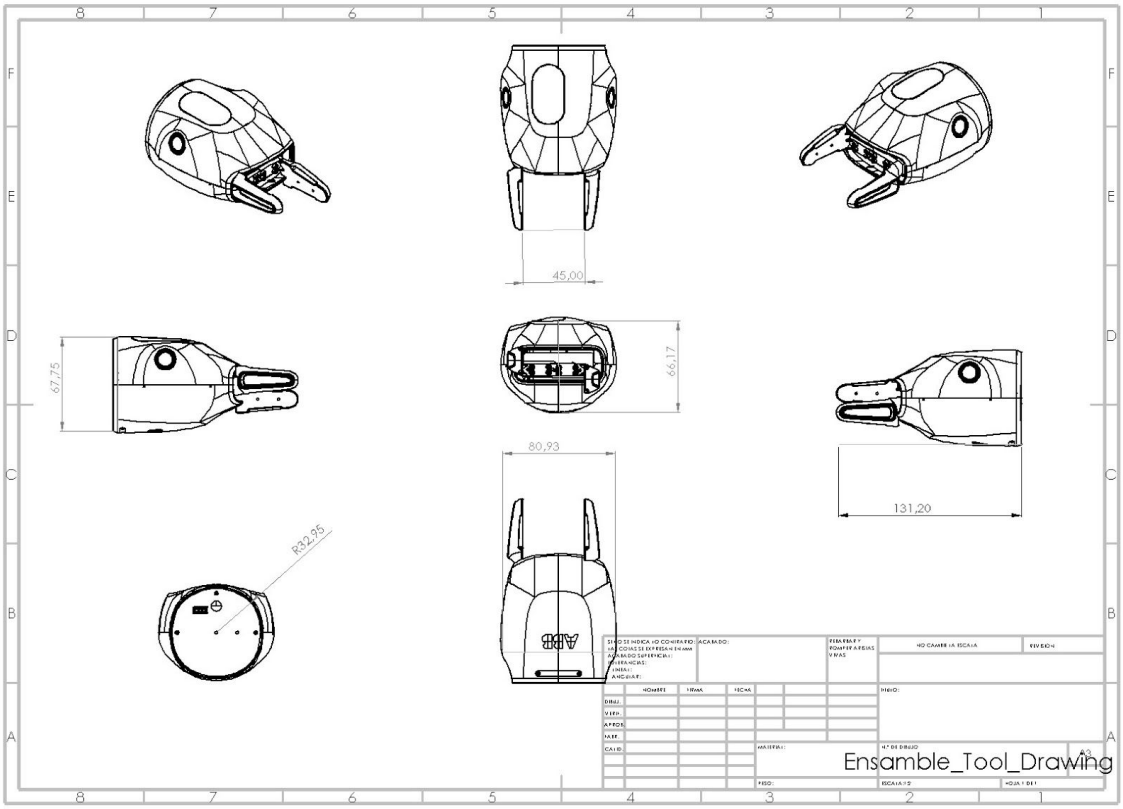


Fig. 12. ABB Smart Gripper (Drawing - Measurements).[12]

B. Dispenser System

The main system is the combination of two seperate parts, the base and the containers. The final designs for this system were based upon the Top Level Technical Requirements (IV), originally the measurements were established as follows: 100cm for the base, 20cm of length and 65cm of height (100x20x65cm). Nevertheless, some modifications were made in order to reduce the space gap between the liquid filler and the glasses placed on the conveyor belt. A more detailed explanation for the final designs is provided in Section B.2. Drawings of this document.

[12]IRB 14000 YuMi Data - ABB's Collaborative Robot -YuMi (Industrial Robots from ABB Robotics). (2020). Retrieved 1 November 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data

1. CAD

a) Base

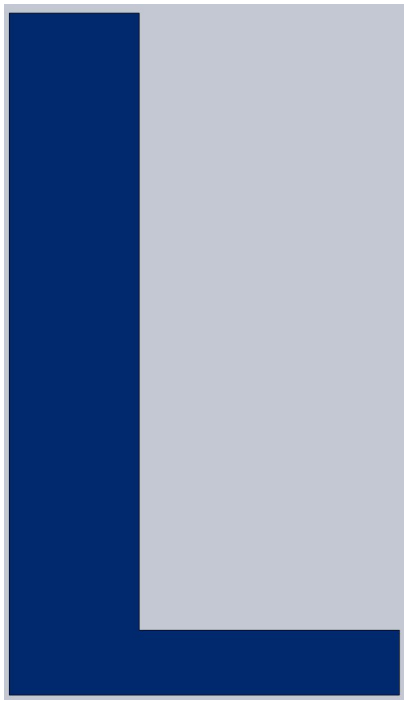


Fig. 13. Base CAD - Side View.

Fig. 14. Base CAD - Front View.

Fig. 15. Base CAD - Upper View.

b) Containers

Fig. 16. Containers CAD - Front View.

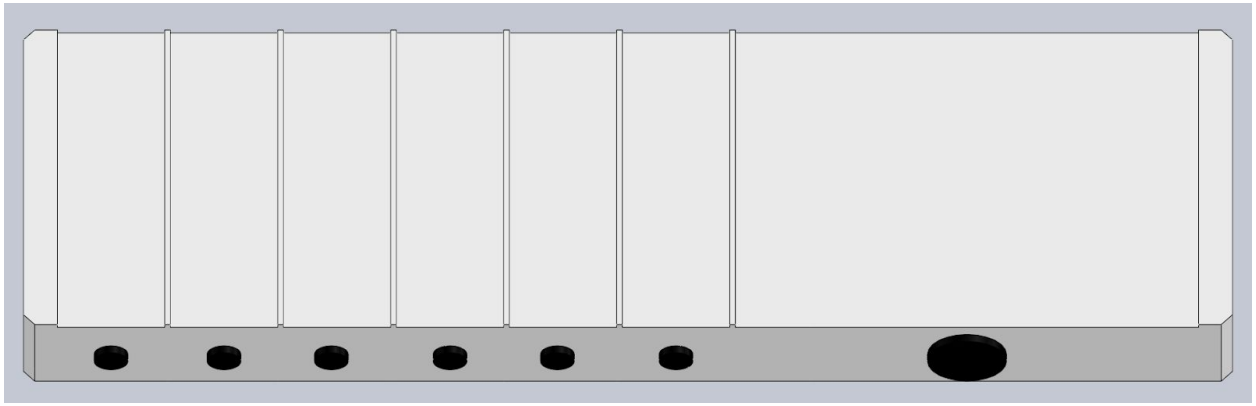


Fig. 17. Base CAD - Bottom View.

Fig. 18. Base CAD - Upper View.

c) Final CAD

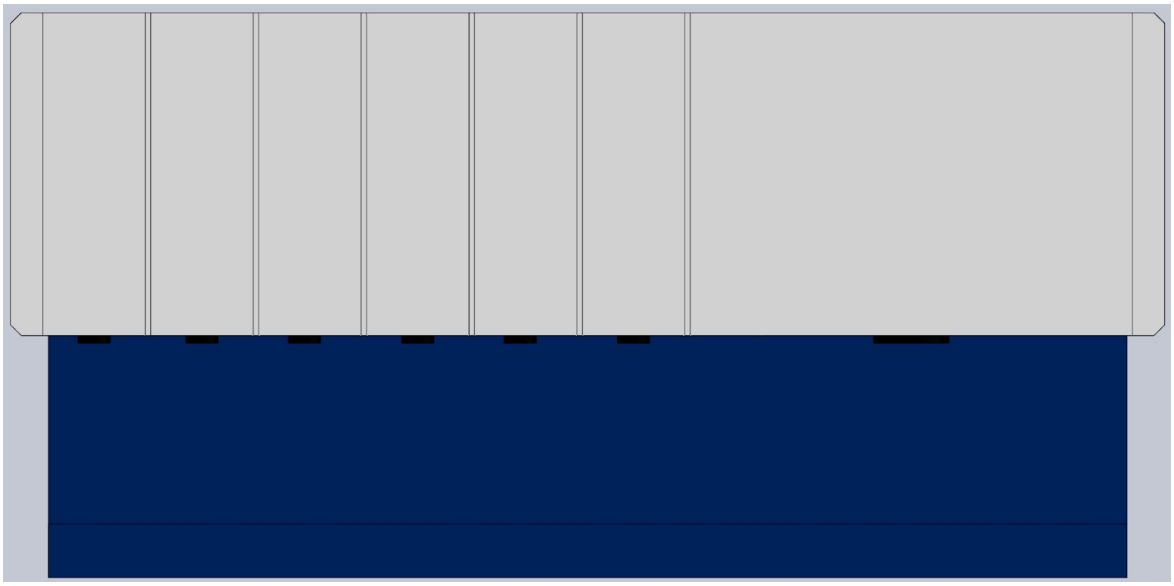


Fig. 19. Base CAD - Front View.

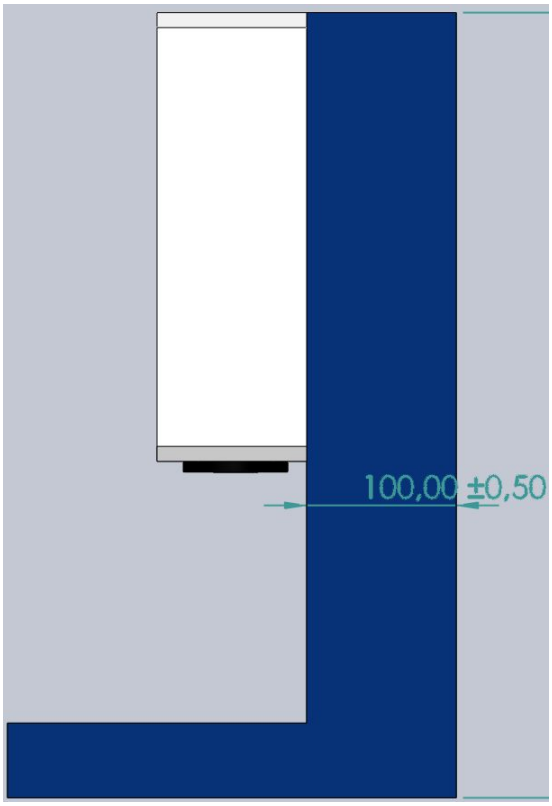


Fig. 20. Base CAD - Side View.

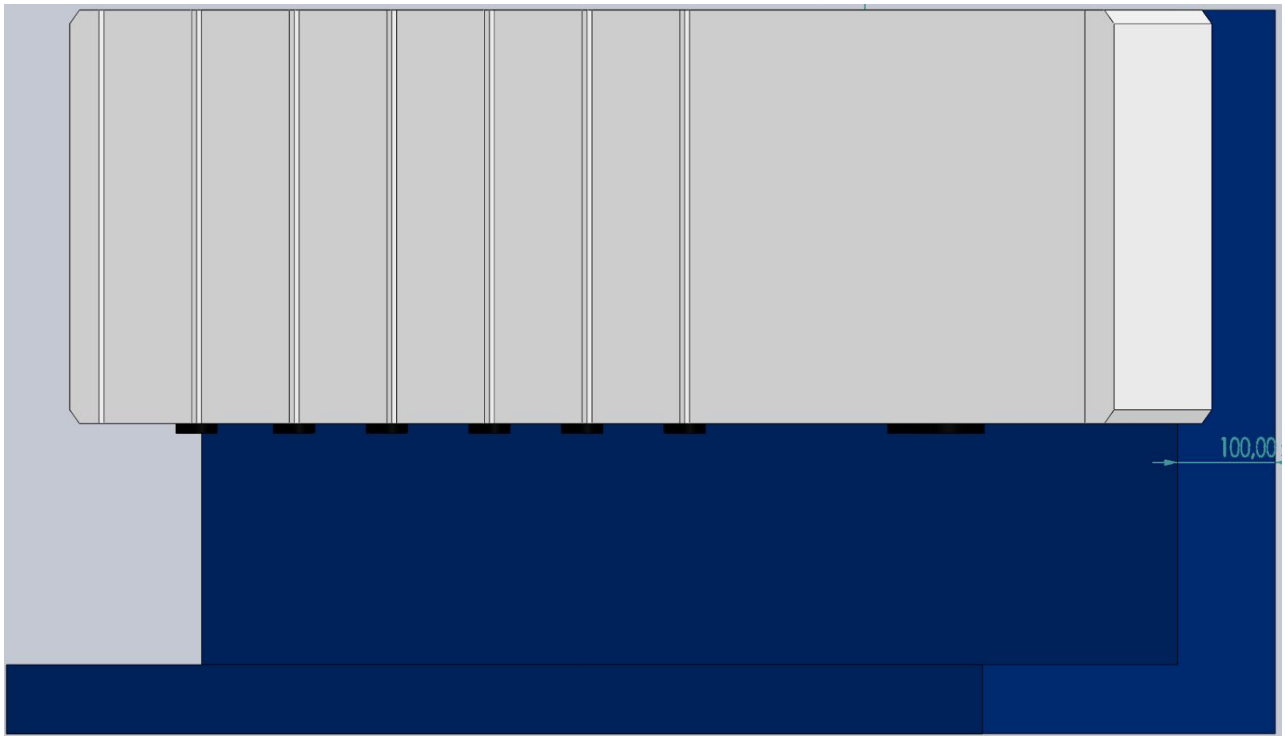


Fig. 21. Base CAD - Diagonal View.

2. Drawings.

a) Base: The original design (IV. Design Requirements Document) established that the base had 65 cm of height. Nevertheless, once the CAD was designed it seemed that the distance between the valves and the glass was more than the expected. This is why the height was modified to 52.5 cm as it can be seen in the following drawing. Besides that, the other aspects of the design were not modified. It still maintained the same "L" shape with a length of 100cm. The width of the main wall was of 10cm in order to have enough space for the tubes, electrovalves, and any circuitry needed for the system.

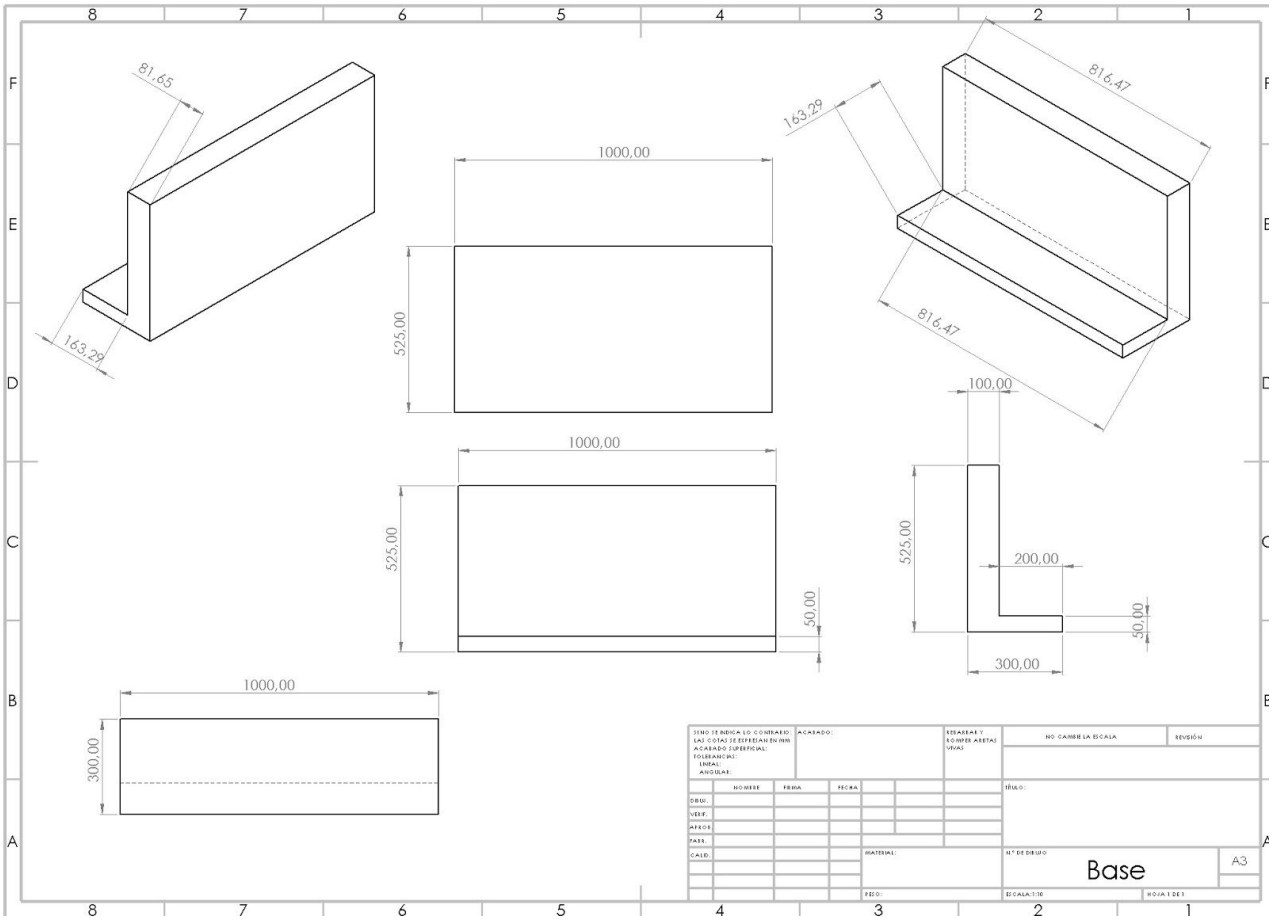


Fig. 22. Base Drawing.

b) Containers: As the Design Requirements A.3 and B.2 established, the final system has 7 different containers. The first six containers are meant for alcohol and sodas, three for each one. The

size of these containers was designed for 3 liters of liquid, so the measurements complied with this (10x30x10)cm. The 5mm between each container was added to consider the gap from the width of the material. The last container had to be the biggest one since it is the one containing the ice, the final measurements were as follows (10x40.5x10)cm. On the bottom of each container there is a valve for the liquid and the ice to fall down. The valve for the ice container is wider so the ice cubes can fall, it was implemented as an open-close gate similar to the ice dispenser on refrigerators. Furthermore, each container has a level sensor and an electrovalve, to comply with functional requirement A.4 and B.10 this way the system can identify when there are not enough ingredients to prepare another drink.

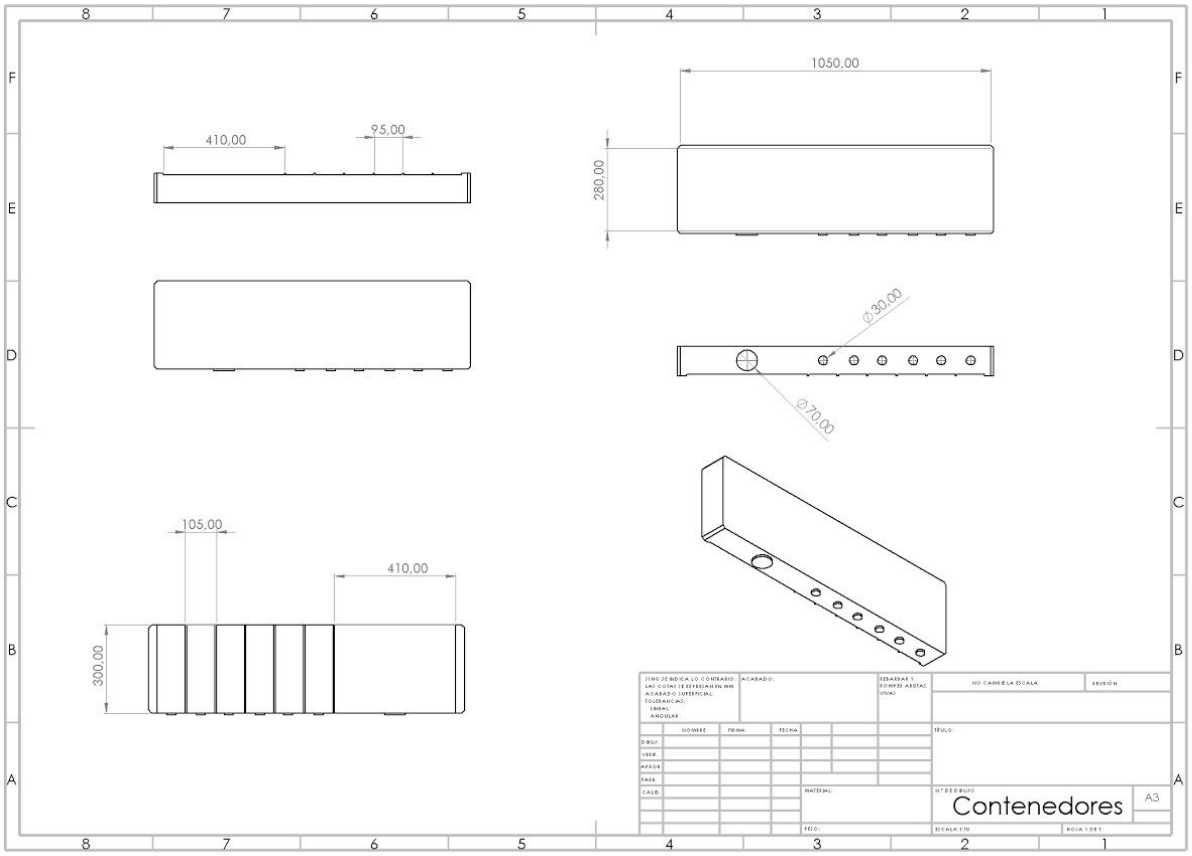


Fig. 23. Containers Drawing.

c) Main Systems:

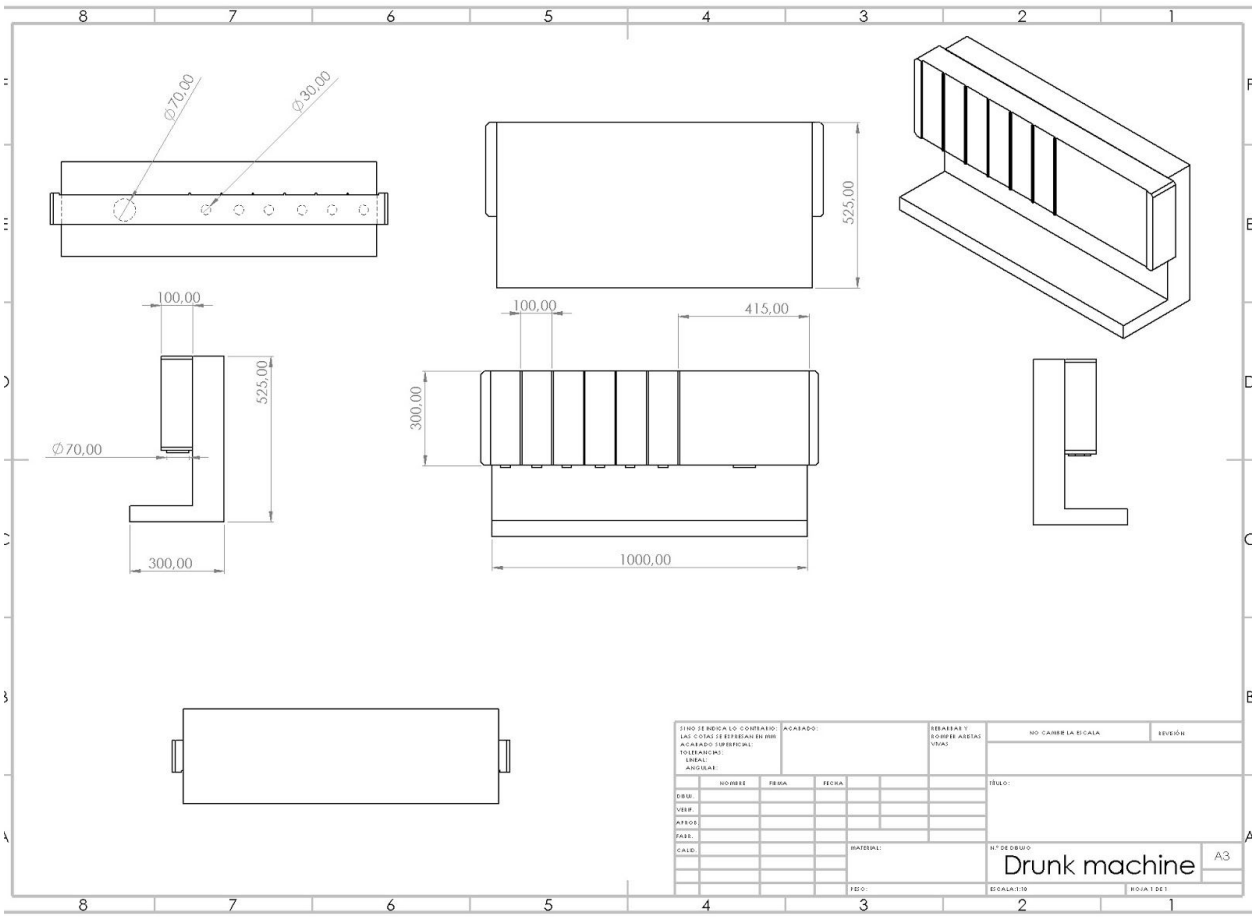


Fig. 24. Main System Drawing.

C. Conveyor belt

1. CAD

Fig. 25. Conveyor Belt CAD - Side View.[13]

[13]Valli, Ricardo. Conveyor Belt with Stepper Motor. (2020). Retrieved 3 November 2020, from: https://grabcad.com/library/conveyor-belt-with-stepper-motor-1

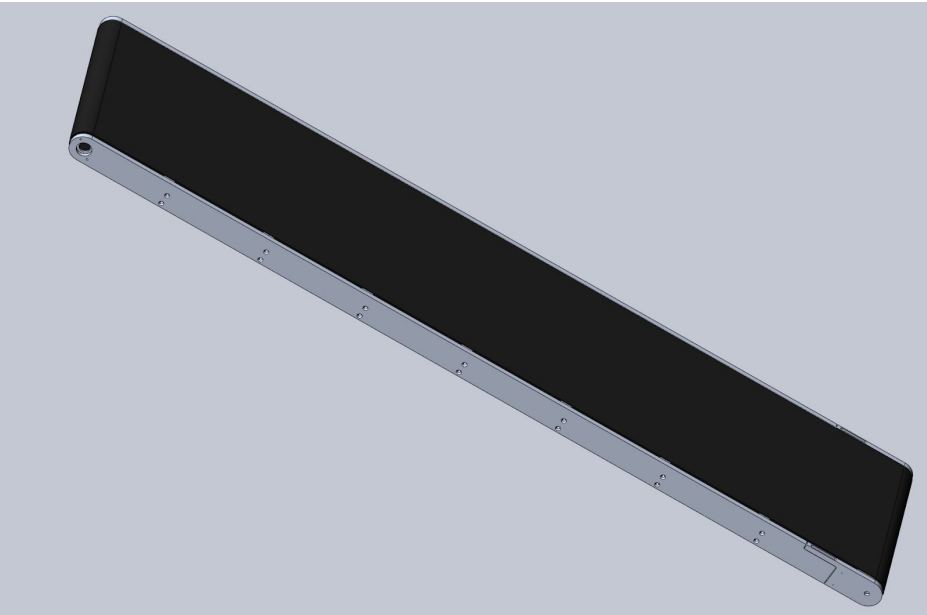


Fig. 26. Conveyor Belt CAD - Diagonal View.[14]

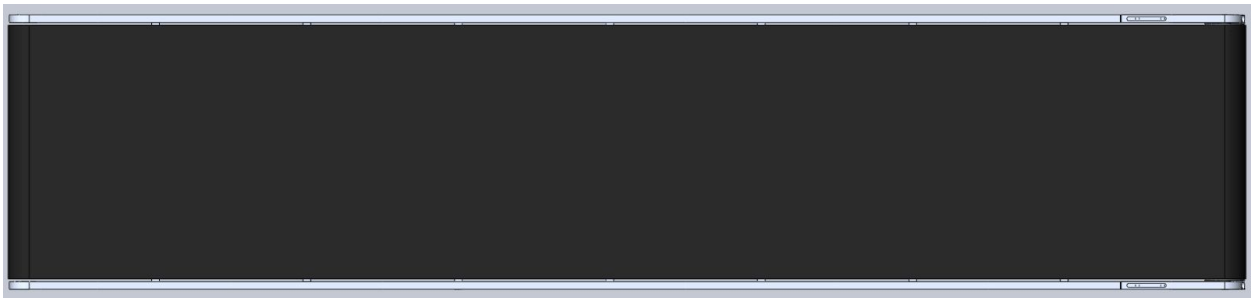


Fig. 27. Conveyor Belt CAD - Bottom View.[15]

2. Drawing.

a. The conveyor belt was designed in order to transport the filled glasses without spilling the drink, as the requirement number A.6 establishes. It has a stepper motor inside the whole mechanism, as considered in the initial budget, so that it can stop in the right position according to the liquid needed, this complies with section B.7 of the design requirements. By controlling the cinematic of the stepper motor, it also complies with the B.9 because the system will be able to stop in three different positions every drink.

[14]Valli, Ricardo. Conveyor Belt with Stepper Motor. (2020). Retrieved 3 November 2020, from: https://grabcad.com/library/conveyor-belt-with-stepper-motor-1

[15]Valli, Ricardo. Conveyor Belt with Stepper Motor. (2020). Retrieved 3 November 2020, from: https://grabcad.com/library/conveyor-belt-with-stepper-motor-1

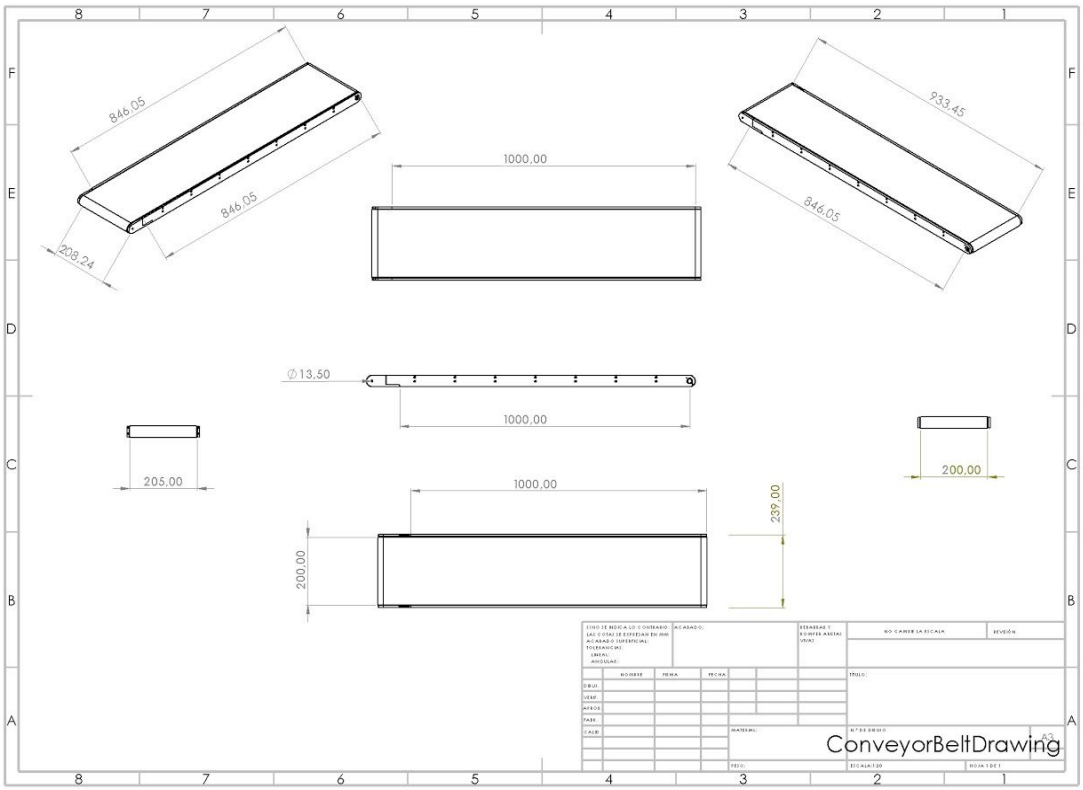


Fig. 28. Conveyor Belt Drawing..[16]

D. Rack of Glasses

1. CAD

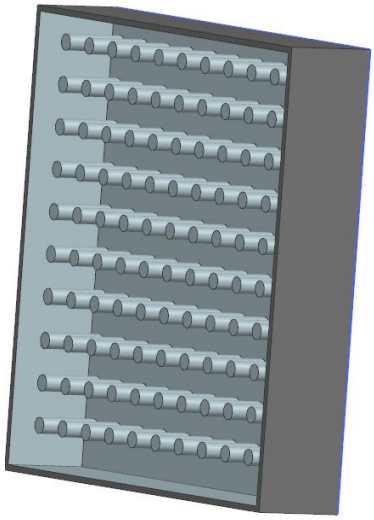


Fig. 29. Rack of Glasses CAD - Side View.

[16] Valli, Ricardo. Conveyor Belt with Stepper Motor. (2020). Retrieved 3 November 2020, from: https://grabcad.com/library/conveyor-belt-with-stepper-motor-1

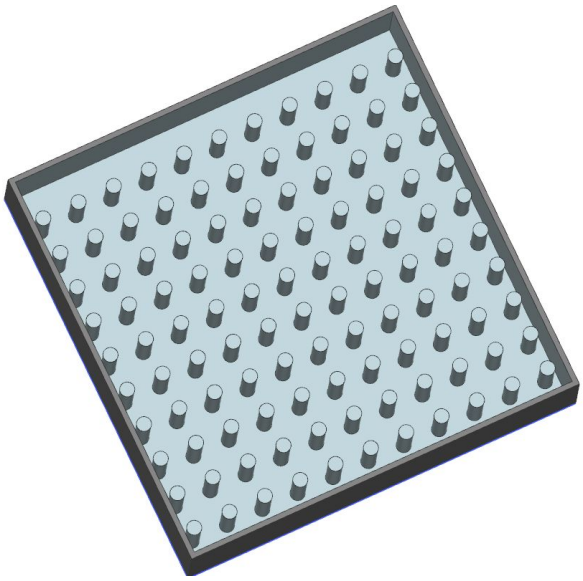


Fig. 30. Rack of Glasses CAD - Upper View.

2. Drawing

a. as the requirement B.5 specifies, the rack is able to contain up to 12 glasses. The radius of each hole is 3.5 cm, which is a little bit greater than the radius of a glass so the robot arm is able to take the glass out of the rack with no difficulty as the A.5 requirement establishes.

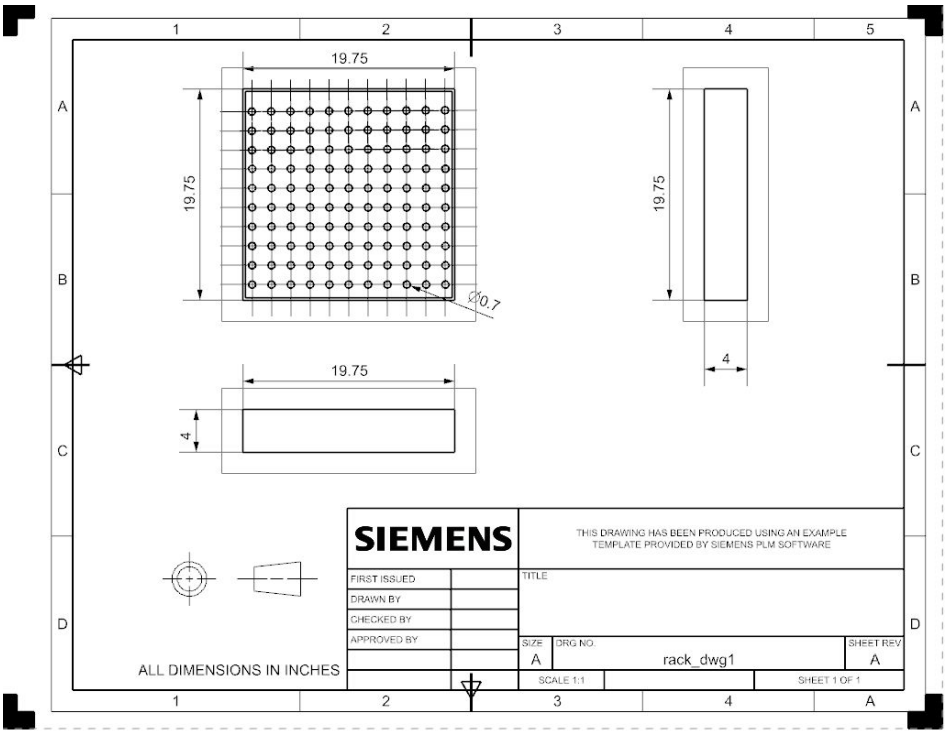


Fig. 31. Rack of Glasses Drawing.

V. Electrical design

A. Solenoid Electro Valve

1. CAD

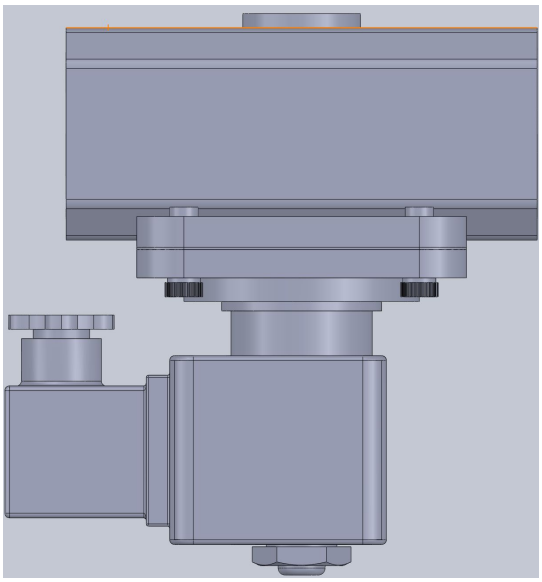


Fig. 32. Solenoid Electro-Valve CAD - Side View.[17]

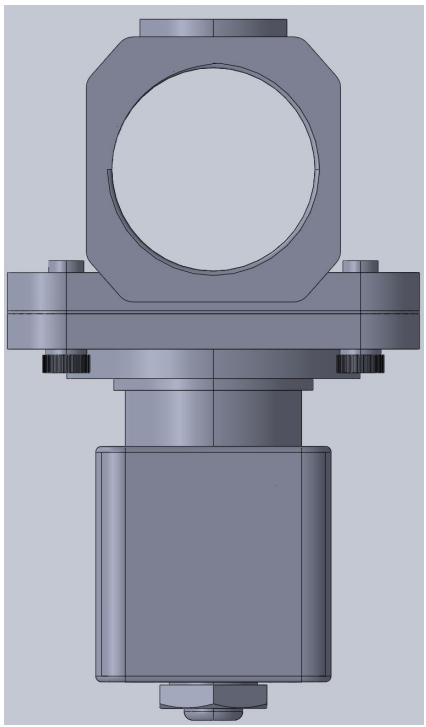


Fig. 33. Solenoid Electro-Valve CAD - Front View.[18]

[17] Abhijeet. Solenoid Valve. (2011). Retrieved 2 November 2020, from: https://grabcad.com/library/solenoid-valve--1

[18] Abhijeet. Solenoid Valve. (2011). Retrieved 2 November 2020, from: https://grabcad.com/library/solenoid-valve--1

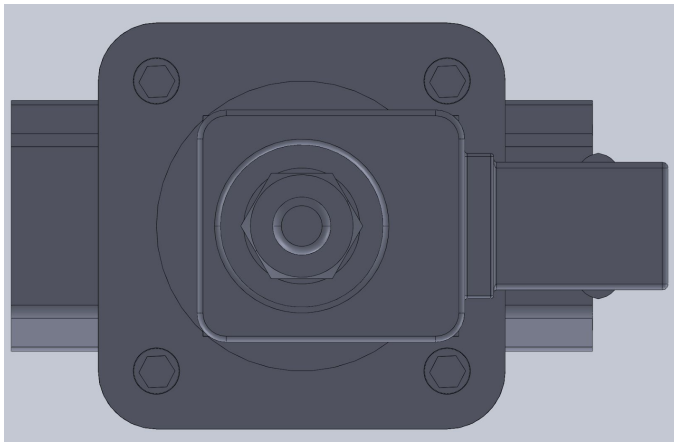


Fig. 34. Solenoid Electro-Valve CAD - Bottom View.[19]

2. Drawing.

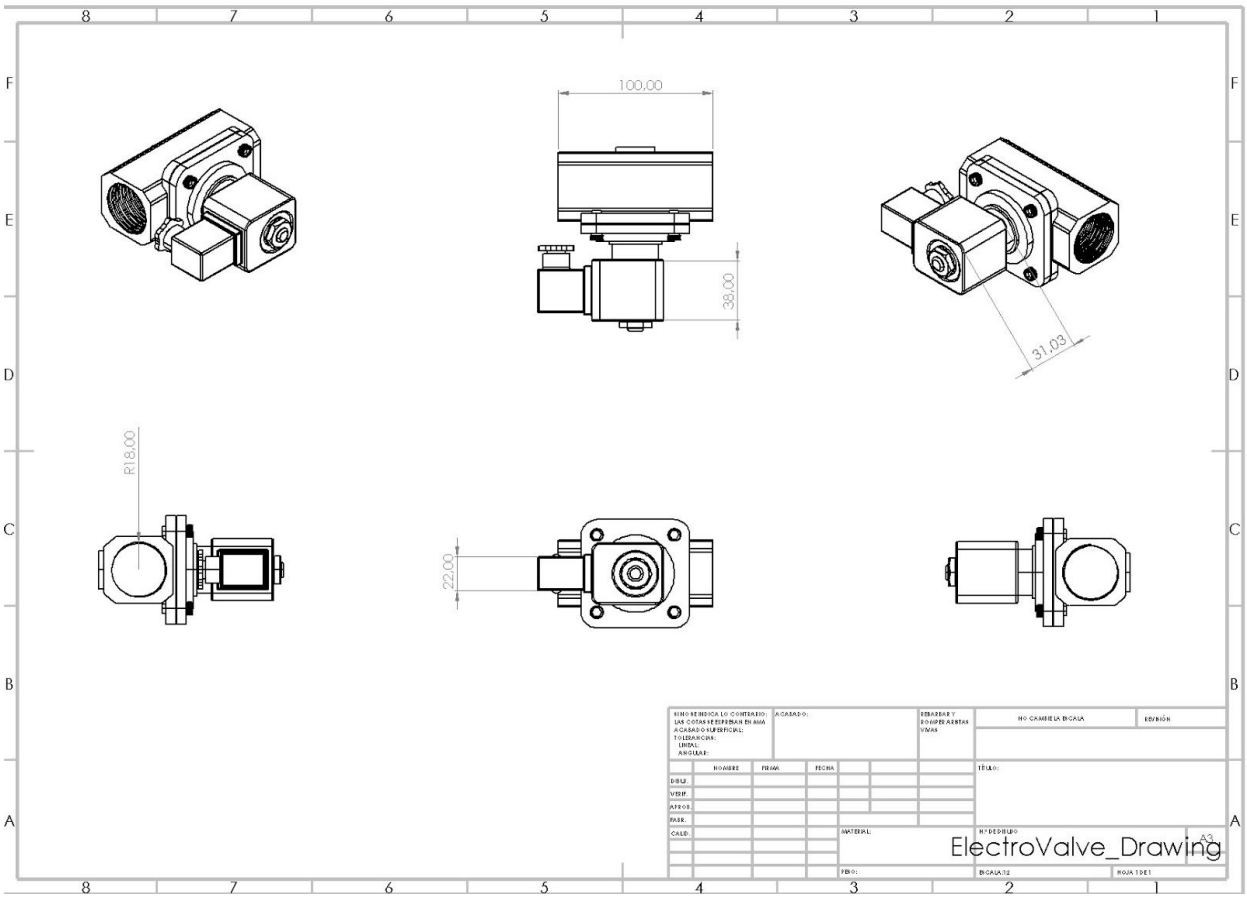


Fig. 35. Solenoid Electro-Valve Drawing.

[19] Abhijeet. Solenoid Valve. (2011). Retrieved 2 November 2020, from: https://grabcad.com/library/solenoid-valve--1

B. Circuit

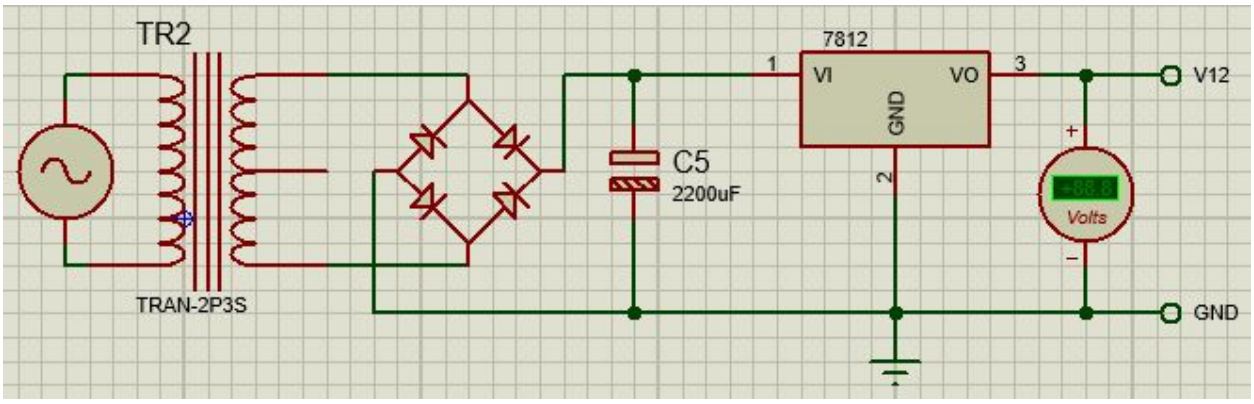


Fig. 36. 12V Power Supplier (Circuit Designed in Proteus).

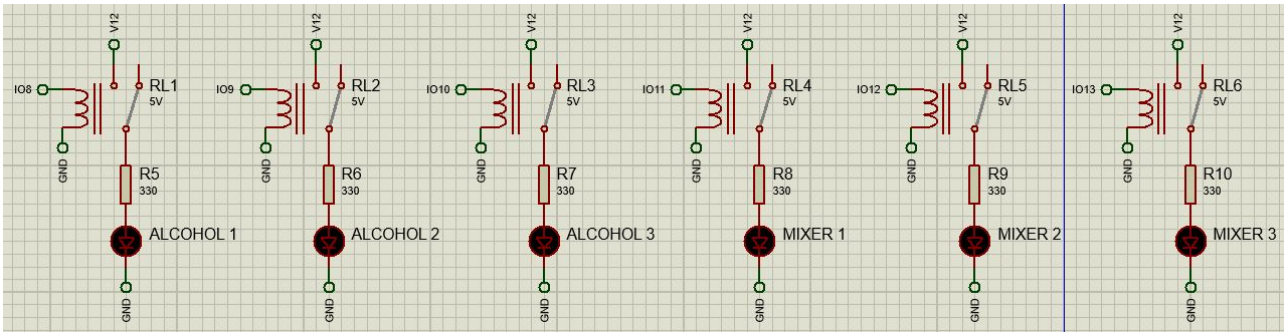


Fig. 37. Electrovalve Simulation (Circuit Designed in Proteus).

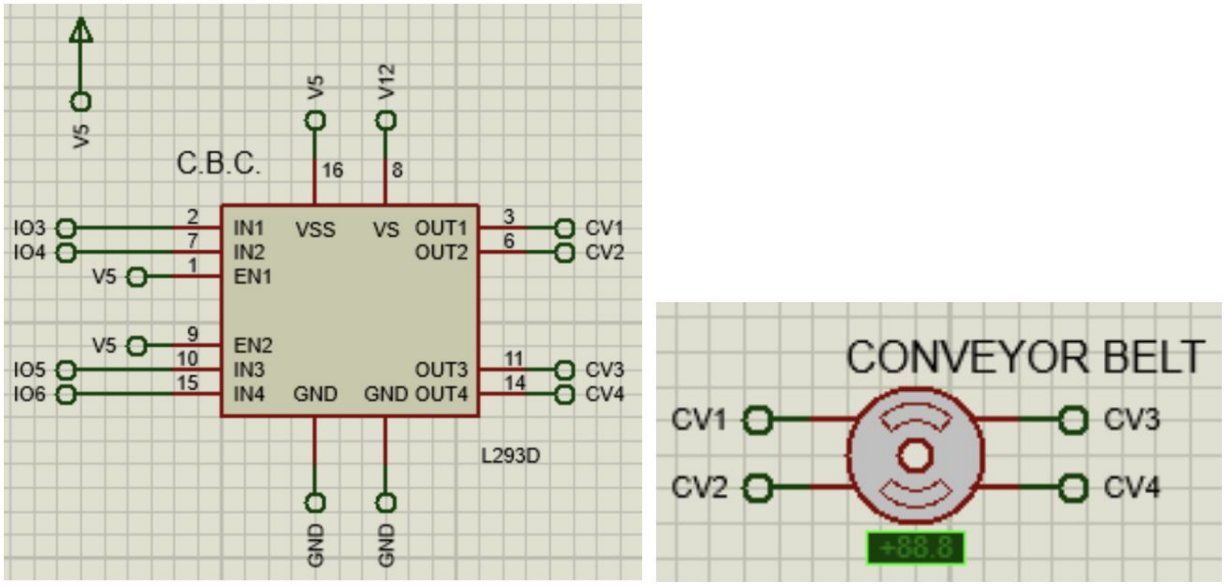


Fig. 38. Conveyor Belt Stepper Motor (Circuit Designed in Proteus).

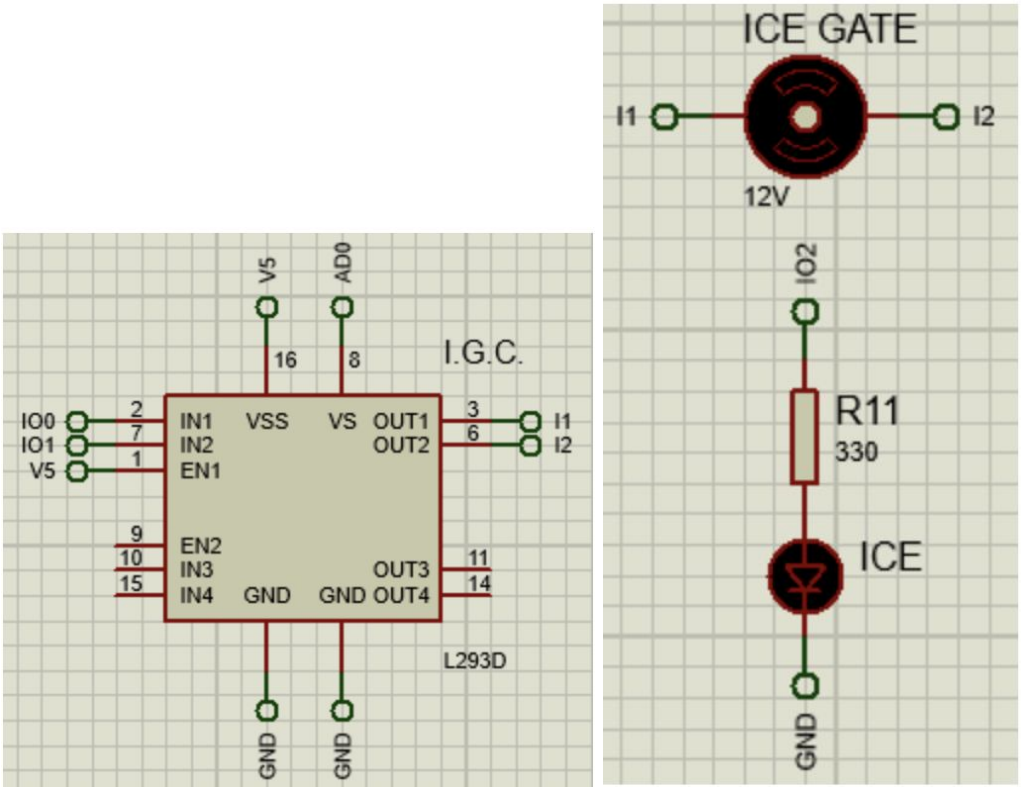


Fig. 39. Ice Gate Stepper Motor (Circuit Designed in Proteus).

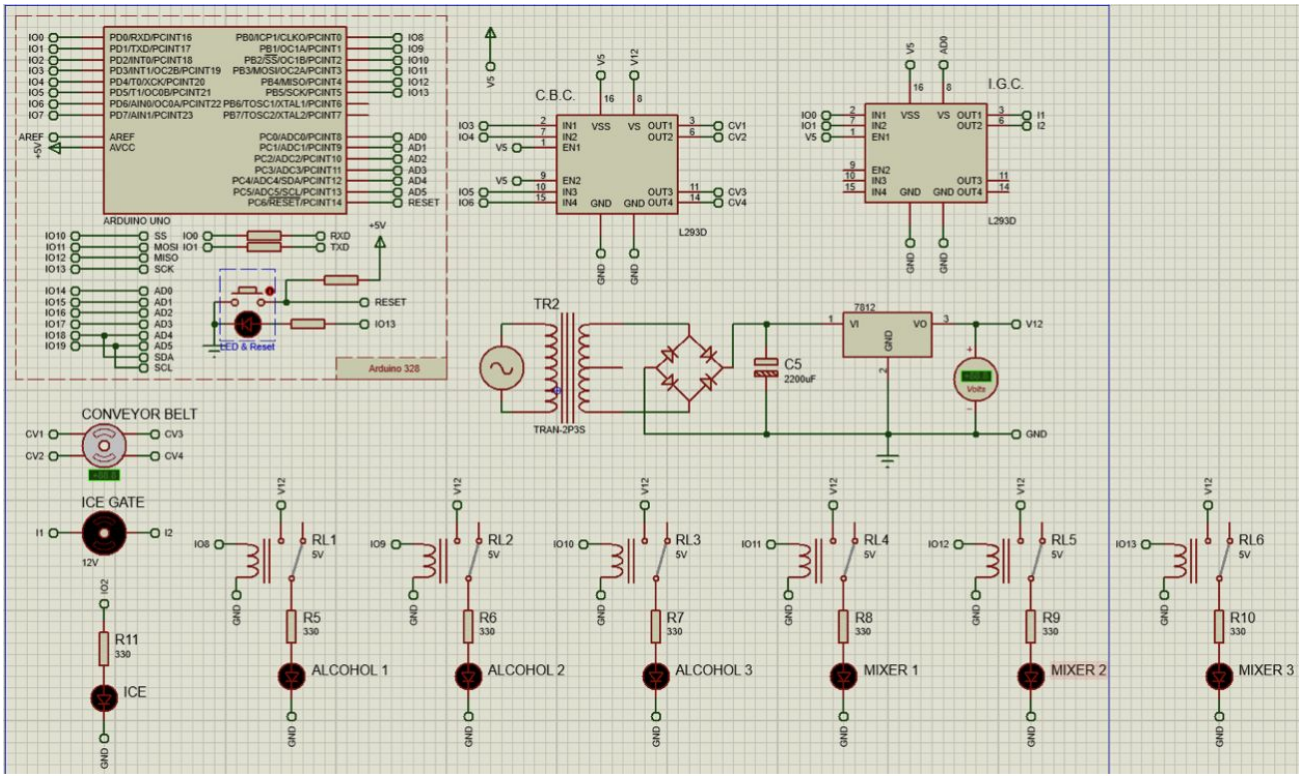


Fig. 40. Final Electric Circuit (Circuit Designed in Proteus).

C. Components description

1. 12v Power Supplier: Designed to supply voltage for the electrovalves and servomotors.
    - 127v-12v Transformer.
    - Diode Bridge
    - Capacitor of 2200uF
    - Voltage Regulator LM7812
2. Electrovalves: Parallel connection of all the electrovalves.
    - Relay srd-5vdc-sl-c
    - Resistor of 330ohms
    - Electrovalve - Tailonz Pneumatic 3/8 Inch NPT 12V Brass Electric Solenoid Valve 2W040-10, Normally Closed, 2 Position, 2 Way Connection Type
    - Raspberry Pi
    - LED
3. Conveyor Belt Motor
    - Stepper Motor Nema 17 Bipolar Stepper 12v 0.4A 40Ncm (56.70z.in)
    - Stepper Motor Driver L293D
4. Ice Gate
    - DC Motor
    - Motor Driver L293D
5. Infrared Sensor
    - GP2Y0D810Z0F: 2 cm a 10 cm, 2.7 V to 6.2 V, 5 mA
6. Level Sensor
    - Grove - Water Level Sensor (10cm) for Arduino 3.3V to 5V
7. Robotic Arms Electrical Design.
    - Since the electrical components and circuits are embedded within the robotic arm, there is no design to present.
    - The following information was obtained from ABB website.

Electrical Connections	
Supply voltage	200-600 V, 50/60 Hz
Rated power transformer rating	3.0 kVA
Power consumption	0.24 kW

Fig. 41. Electrical specifications for the robotic arms.

D. Requirements covered for Electrical Design

The first part of the electrical design was based upon the design requirements A.8 and B.10 which establishes the control for the liquid poured into the glasses. In order to comply with these, the system includes 6 electrovalves, one for each container. The design for dropping the ice was just a simple switch that opened or closed a door when the microcontroller sent a signal. Since the electrovalves need 12V to work, the power supplier needed for the whole system was designed this way. Furthermore the electrovalves needed to be controlled by a microcontroller, but its I/O pins only supply 5v. So to solve this problem relays were added to the circuit, this way the valves can be controlled and the Raspberry is protected from any voltage overload.

For the electrical design of the conveyor belt, the requirement B.9 stated that the preparing sub-system shall have 3 different positions for the 3 different steps of the processes. Taking this into consideration and using the 12v supplied to the system, a stepper motor was the best option to make it work. A decoder was implemented in order to have control over it with the microcontroller. With this stepper motor the process can be controlled and locate the conveyor belt at any position needed.

E. Sensors used in RobotStudio

For the sensor's implementation the PlaneSensor intelligent component was used. To use this sensor the radius of detection and the object to detect needed to be defined, in this case the radius is 10 cm and the object is the glass. The system has one sensor in each valve, a total of eight sensors, and depending on the sequence selected, the sensors that get activite to start reading. On each sequence four sensors are activated, the one for the ice pouring, for the alcohol, the mixer and to indicate when the drink is finished.

Fig. 42. Plane sensor dialog box in RobotStudio.

F. Actuators

For the actuators implementation the LinearMover intelligent component was used. To use this actuator the velocity of the real conveyor belt was measured. This component works on the glass and moves it in the y axis. There are 4 different LinearMovers in the system in order to move the glass over the different positions in the conveyor belt, one from the start position to the ice sensor, the second one from the ice sensor to the alcohol sensors, the third one from the alcohol sensors to the mixers sensors, and finally from the mixers sensors to the final sensor.

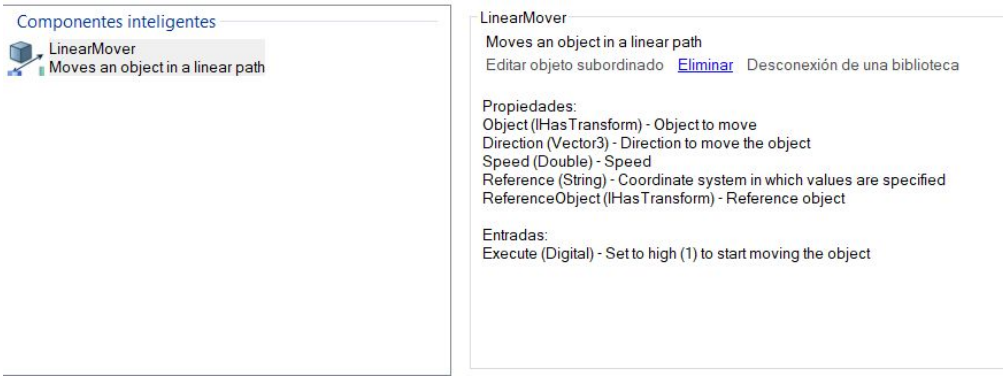


Fig. 43. Linear Mover dialog box in RobotStudio.

VI. Software

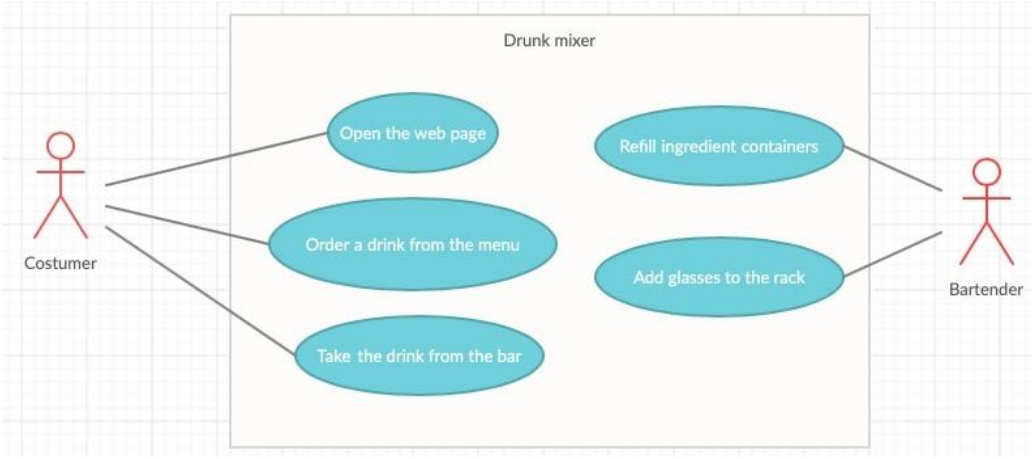


Fig. 44. System Use Cases Diagram.

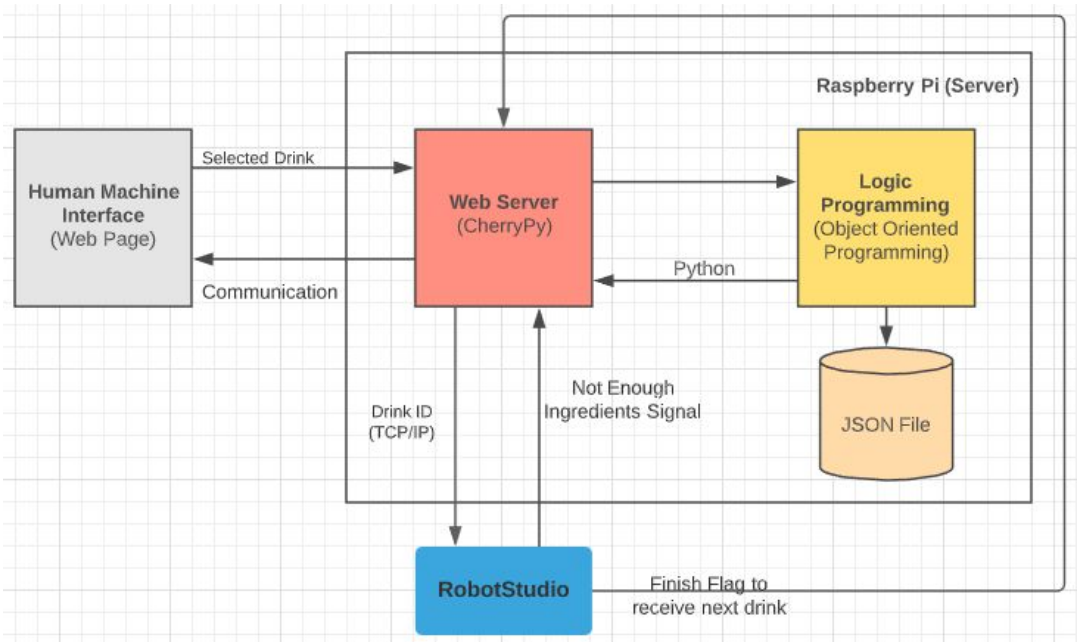


Fig. 45. Top Level Software Design Diagram.

In the diagram shown above it can be seen the interaction between the 3 main interfaces. In order to begin the production of a beverage the web page will send to the server the selected drink with its ID. The server will then accept the ID, add it to a list, and check the attributes of the drink in order to know which ingredients are needed. After knowing this information it will send the drink ID to Robotstudio. Then RobotStudio will prepare the drink and send a completion signal that the server will be waiting in order to send the next drink.

The server will be created with the CherryPy framework using the Python programming language and will be hosted locally in the Raspberry Pi. It will provide the web application to order the drinks, which will be accessed via https on a browser inside the local network. The server will also integrate websockets that will act as the interface to communicate with RobotStudio. It will manage all the incoming requests from the web page and organize them to serve them in a FIFO queue. This queue will have a limit of 60, if it reaches the limit it will stop receiving requests until the drinks are served out of the queue so new requests can come in, fulfilling the requirement A.3. The web application will serve as our interface with the persons to select the drink desired as the requirement A.2 specifies.

At the beginning of the process the robot will connect to the server via websocket, the server will use this connection to send messages that contain the ID of the requested drink and receive messages of signals from the robot. In this way this communication fulfills the requirement A.1 which states that the system shall be able to access our web server and prepare the drinks requested.

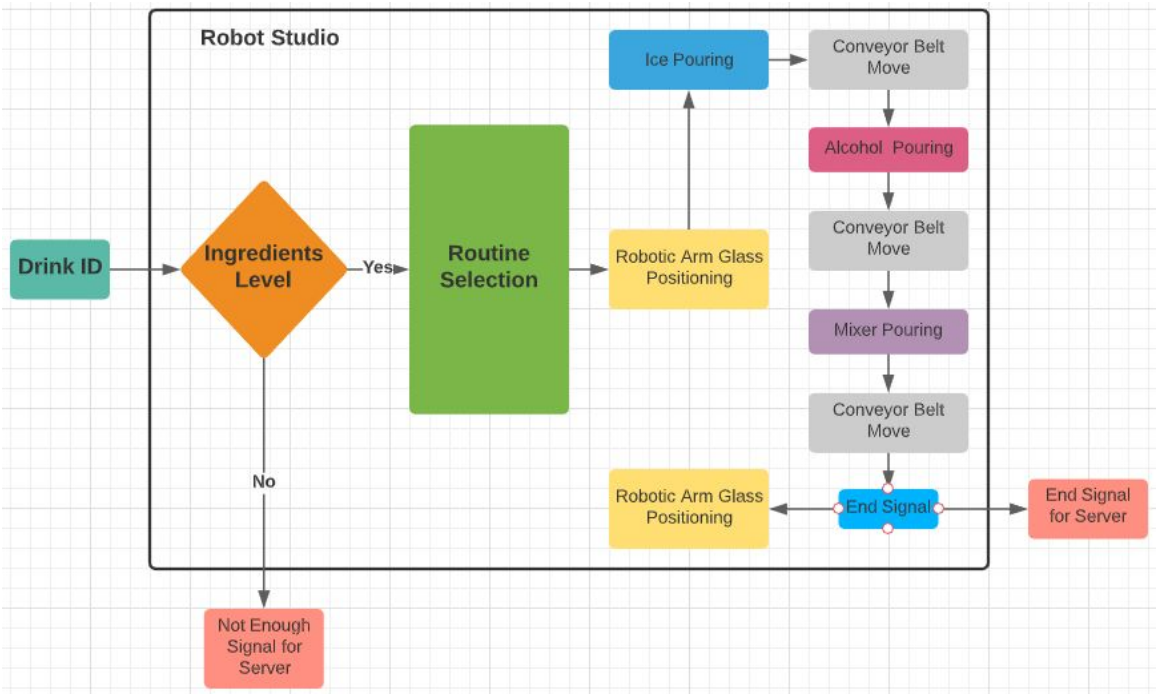


Fig. 46. Detailed Robot Studio Software Design Diagram.

The diagram shown above represents the signals that will control the production of each beverage. At first Robostudio will receive the drink ID from the web server, then it will check if there are enough ingredients to prepare it, if not it will send a signal to the server reporting it. The next step is to select the routine, Robotsutdio will have 9 different routines according to each drink that can be prepared, each routine will be identified by the ID of the drink. After knowing the routine a signal will be sent to the first robotic arm so it can place the glass in position. Then it will pour the ice into the glass, move the conveyor belt to the position of the selected alcohol container, pour the liquid, move again the conveyor belt to the position of the selected mixer, pour the mixer and finally move the glass to the final position. In this position the second robotic arm will move the prepared drink to the bar.

As the requirements A.9 and A.7 state, the preparing sub-system shall have different positions for the different steps to prepare the drink and the system shall be able to stop in the right position according to the liquid needed. The system has already predefined routines for all the combinations available, this way each presence sensor is activated when required to add the specific quantities of ice, alcohol, and mixer needed. It has one infrared sensor on every position of the system. It will detect when the glass is under the corresponding valve and start pouring the liquid. It has a close loop controller that will receive the sensor's signal in order to stop the stepper motor and open the valve to pour the specified quantity. Fulfilling both of the requirements mentioned above and also, the requirement A.10 which says that the system shall pour 100 ml of soda and 40 ml of alcohol.

The first step of the process mentioned above will check for all the level sensors inside the containers and if they do not have the minimum amount of ingredients to prepare the drink it will send a signal back to the server and exit the preparation process. This function fulfills the requirement A.4 that prevents the preparation process from starting if it does not have enough ingredients and sends a signal to the server so it can be refilled.

A. UML Diagram

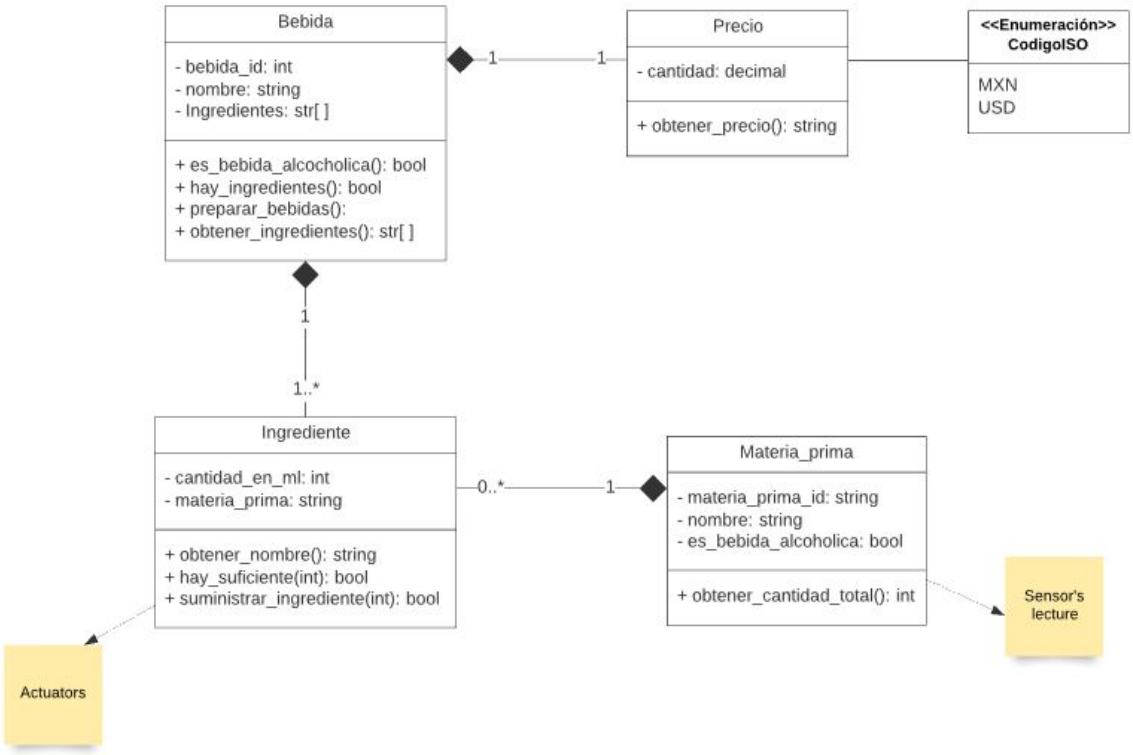


Fig. 47. UML Diagram.

B. Web Page

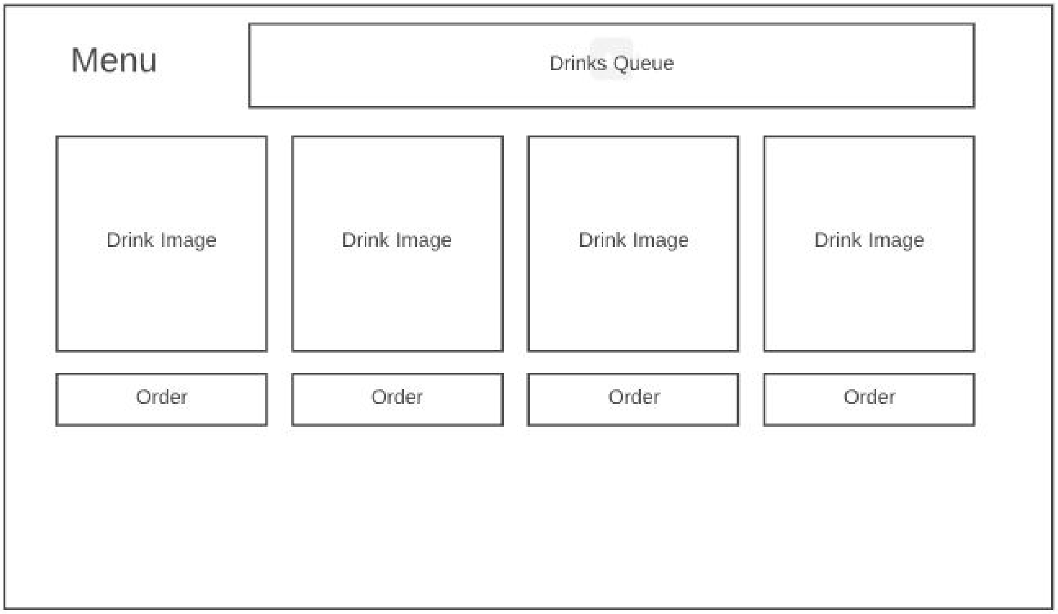


Fig. 48. Web Page Layout.

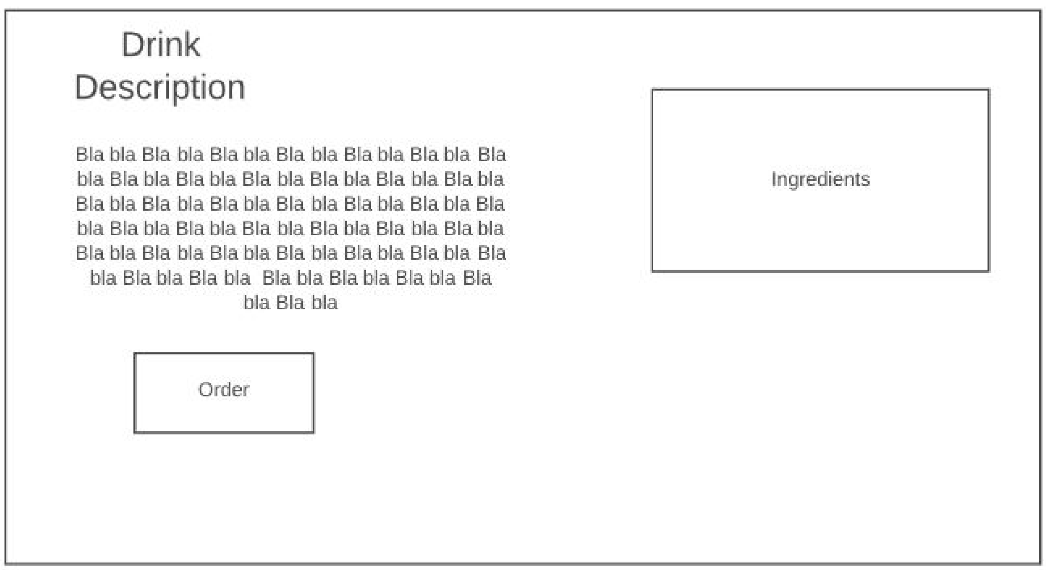


Fig. 49. Web Page Layout Drink Description.

C. Server

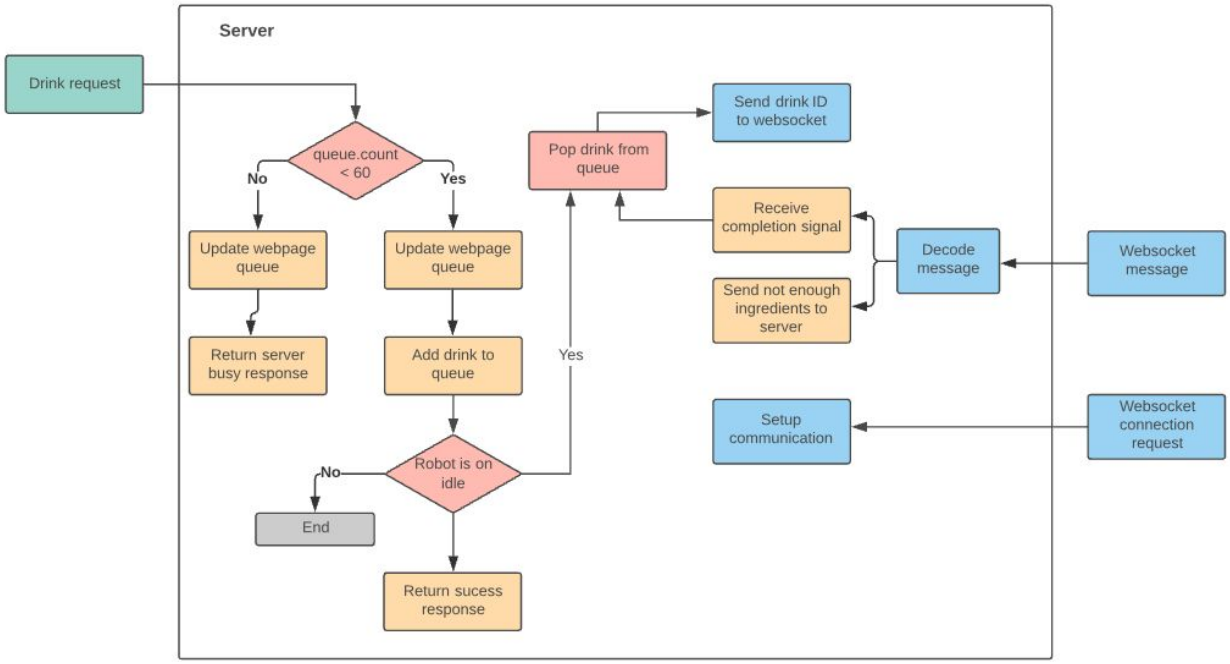


Fig. 50. Server's Diagram.

VII. Control Design

The final system was designed as a closed-loop control system. Where it received the signal from a sensor and it kept moving according to the step of the process it was in. There are 8 different sensors distributed along the process, one for the ice, three for alcohol, three for the mixers and one to stop the glass. Each sensor is independent from one another, they do not need for the previous sensor to send back a signal in order to be active. Nevertheless, for the process to be continuous the glass needs to go through each one. The logic for this process is explained further along in the flowchart.

For the implementation of the state machine on RobotStudio, a *test case* was used. This test case gets the ID of the drink, and identifies the selected drink to activate the corresponding sequence of preparation and decrement the corresponding quantity on the ingredients used.

A. Flow Chart

In order to implement the control system correctly, a flowchart was designed using Raptor. The idea was to create the process in the same way it would be implemented in RobotStudio. The first step was to declare all the variables required in order to keep track of the

sensors. As it was stated before, there are eight different sensors which stop the glass at different stages of the process, but each process uses only four sensors (ice, alcohol, mixer, stop). The first feedback needed for the system to begin is the signal from the Server when it receives an order from the customer. In the flowchart this is represented by the input chart “Selecciona tu bebida”. Once the customer has selected the drink it enters into a loop where depending on the beverage selected the alcohol and the mixer change, although every process goes through the same amount of sensors.

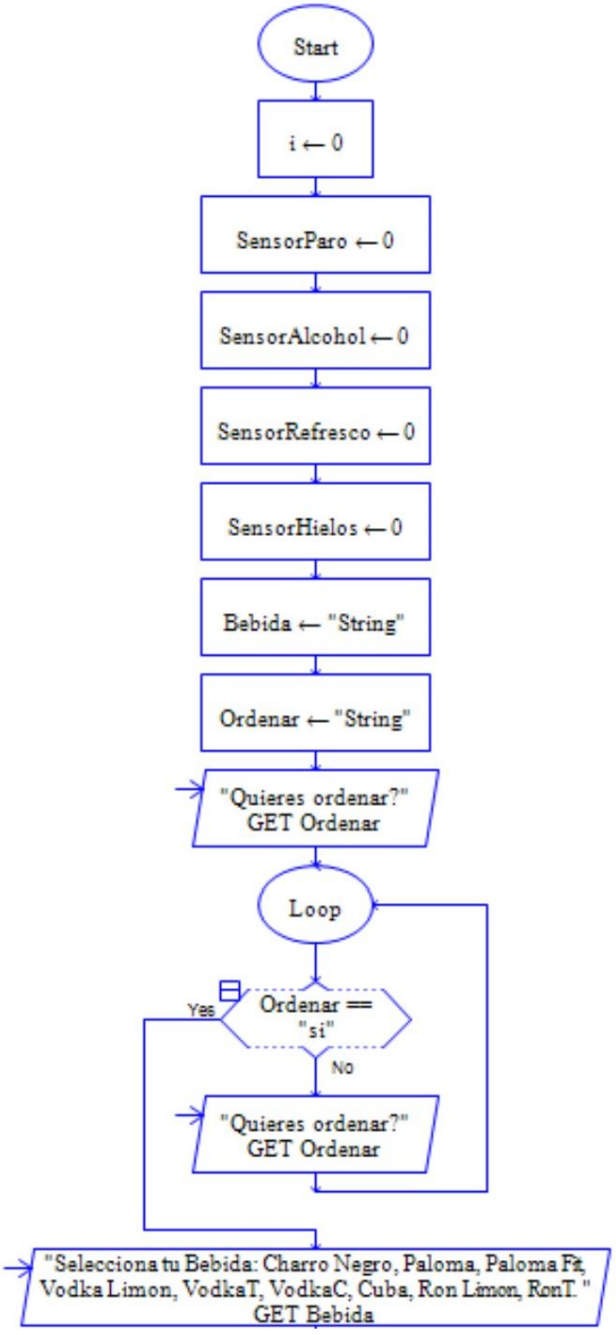


Fig. 51. Flowchart Variable Declaration (Raptor)

In this example the beverage selected was “Charro Negro” and the first step of the process if for the first Robotic Arm to pick up the glass from the rack of glasses and place it on the conveyor belt. Once the glass is on the conveyor belt, it starts moving until the ice sensor detects the glass and stops it for 3 seconds while it serves the ice. If the sensor does not detect the glass the simulation ends sending an ERROR message. Then the glass starts moving again until the second sensor detects it.

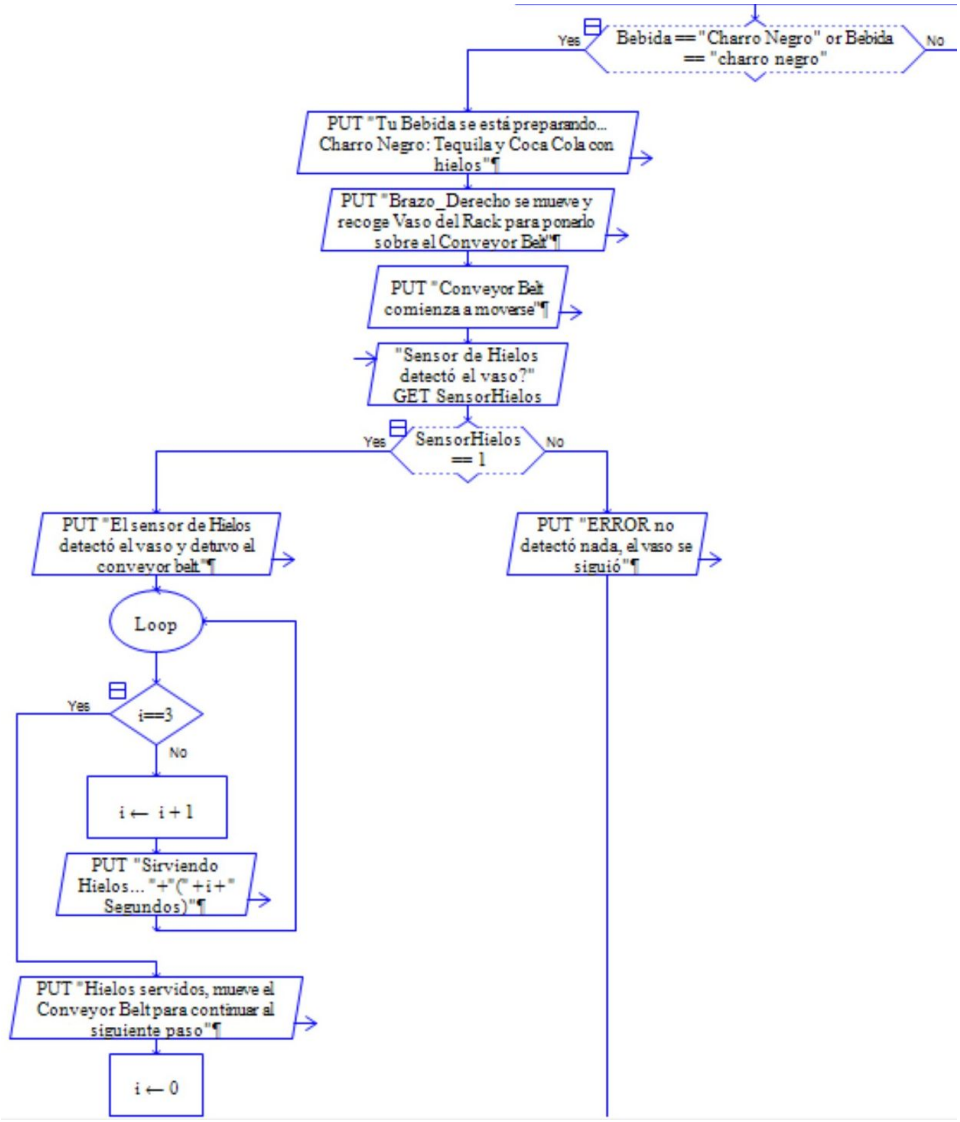


Fig. 52. Flowchart Ice Loop (Raptor)

If the alcohol sensor detects the glass it stops the process and pours the corresponding alcohol into the glass, which in this case is “Tequila”. It waits for 4 seconds and then begins its movement again. Again, if the sensor does not detect the glass the simulation ends sending an ERROR message.

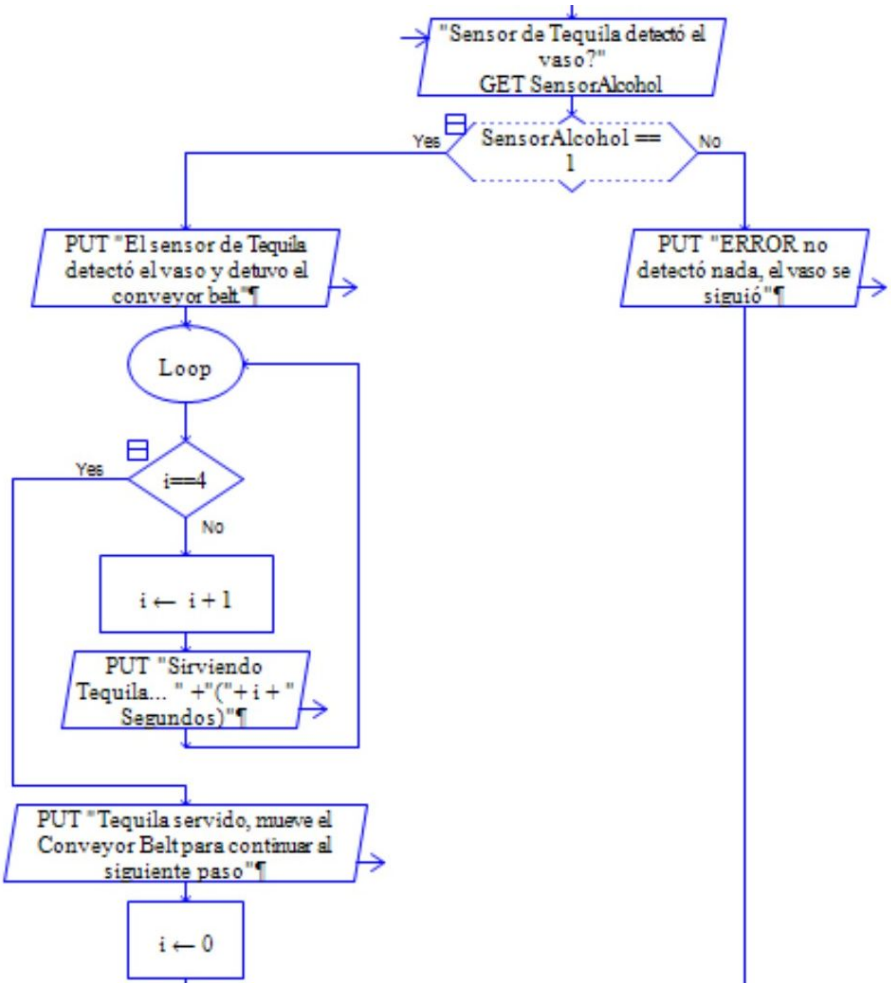


Fig. 53. Flowchart Alcohol Loop (Raptor)

The next sensor of the process is the mixer sensor, which implements the same logic as the previous two. If the sensor detects the glass it stops it and pours the mixer for 7 seconds, if not it sends an ERROR message. Finally, the glass keeps moving to the end of the conveyor belt until the stop sensor detects the glass and stops it. Then the second Robotic Arm picks up the glass from the conveyor belt and places it on the bar.

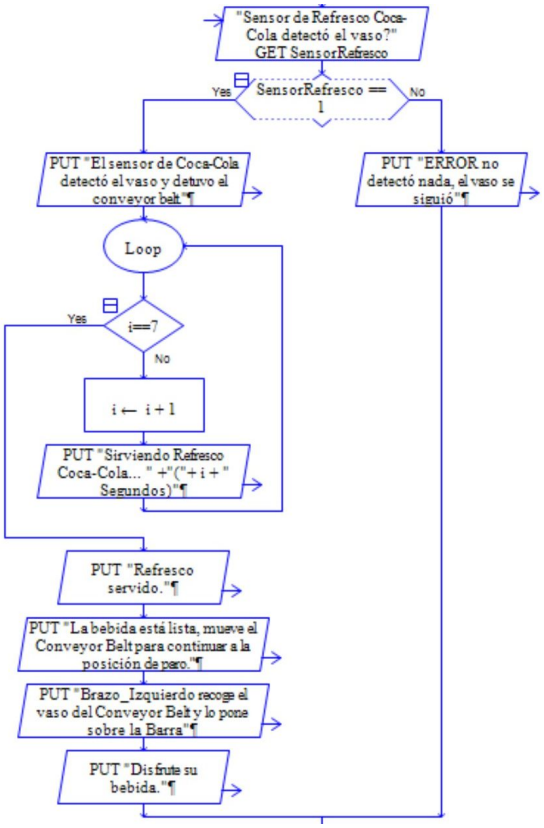


Fig. 54. Flowchart Mixer Loop (Raptor)

B. Rapid (Robot Studio)

RAPID is a high level textual programming language developed by ABB. A RAPID application consists of the program and the system modules.

- **Program:** Is an instruction sequence that controls the robot and consists of three parts:
  - **Main Routine (main):** Where the executions start.
  - **Subroutines:** Divides de program in smaller parts to obtain a modular program.
  - **Program Data:** Defines Positions, numeric values, coordinate systems, etc.

C. Station Logic

Has some of the characteristics of a Smart Component. It can be used to work with these characteristics on the station level.

The Station Logic Editor consists of the following tabs similar to that of a Smart Component editor:

- Compose
- Properties

- Signals and Connections
- View

For this project the next Smart Components we used:

- Positioner: This component sets the glass used in the simulation at the given position and orientation.

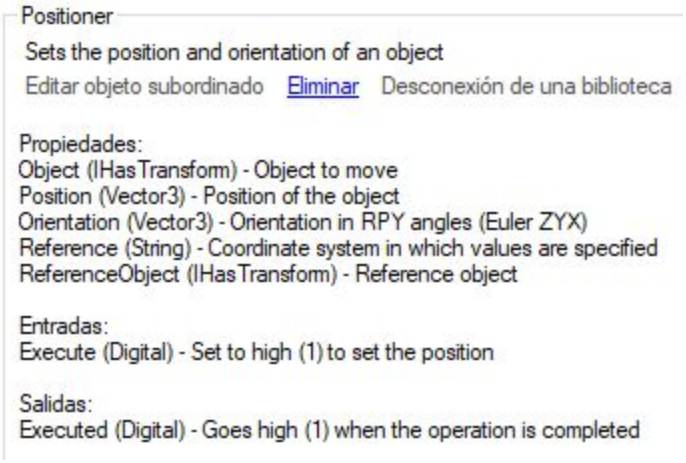


Fig. 55. Positioner description.

- Plane Sensor: This component was implemented to substitute the real infrared sensor, whenever it detects the glass sends a signal to the controller and the controller executes the programmed action.

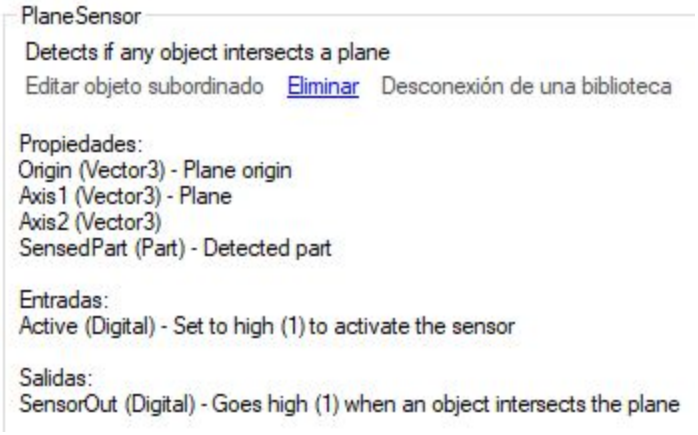


Fig. 56. Plane Sensor description.

- LogicSRLatch: This component sets and resets the linear movers of the glass whenever it's needed.

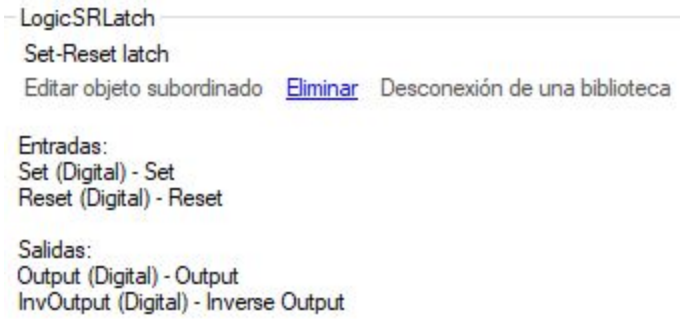


Fig. 57. LogicSRLatch description.

- LinearMover: This component simulates the movement of the conveyor belt, but instead moves the glass.

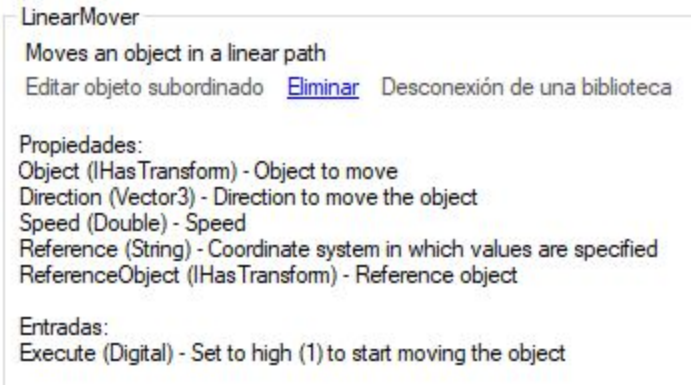


Fig. 58. LinearMover description.

- Attacher: This component is used to simulate when the robotic arm takes the glass and moves it to another position. It attaches the origin of the glass coordinate system to the origin of the robotic arm gripper coordinate system.

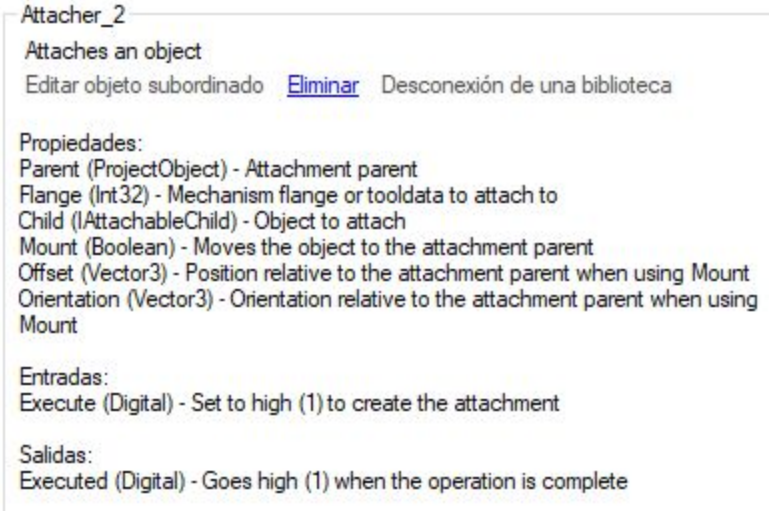


Fig. 59. Attacher description.

- Detacher: This component is used to simulate when the robotic arm takes the glass, moves it to another position and leaves it there. It detaches the origin of the glass coordinate system to the origin of the robotic arm gripper coordinate system.

Detacher

Detaches an attached object

Editar objeto subordinado Eliminar Desconexión de una biblioteca

Propiedades:
Child (IAttachableChild) - The attached object
KeepPosition (Boolean) - If false, the attached object is returned to the original position.

Entradas:
Execute (Digital) - Set to high (1) to remove the attachment

Salidas:
Executed (Digital) - Goes high (1) when the operation is complete

Fig. 60. Detacher description.

VIII. Performance Tests

A. Importing CAD's designs to Robot Studio

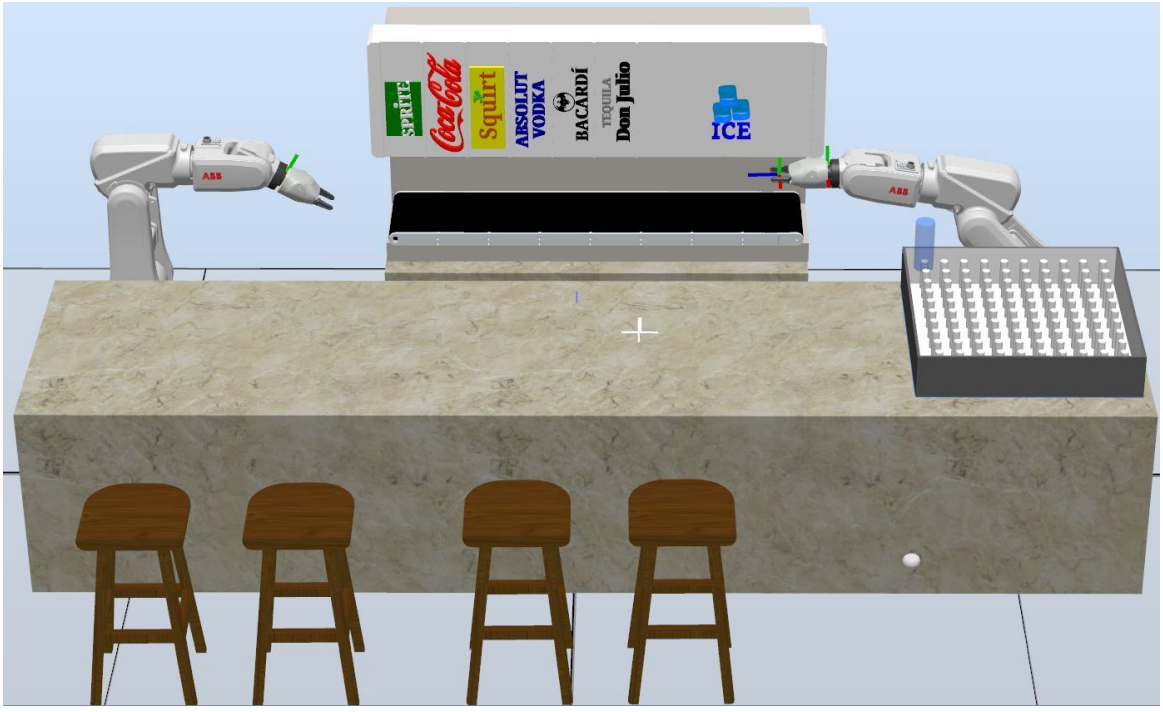


Fig. 61. RobotStudio Station View 1.

Fig. 62. RobotStudio Station Front View.

B. Simulating electrical components on Proteus

As it can not be shown the video of the simulation in proteus, here are some pictures of the components working during the simulation.

- **Power Supplier**

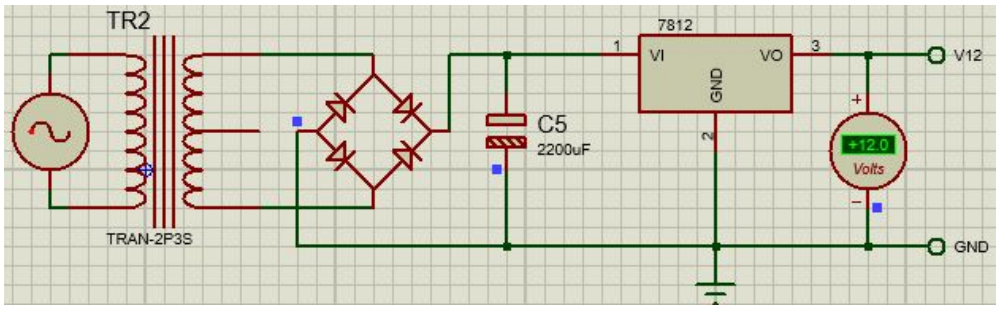


Fig. 63. Power Supplier.

In this image is shown the voltage provided by the power supplier, as needed, the supplier gives 12 volts.

- **Conveyor Belt Stepper Motor**

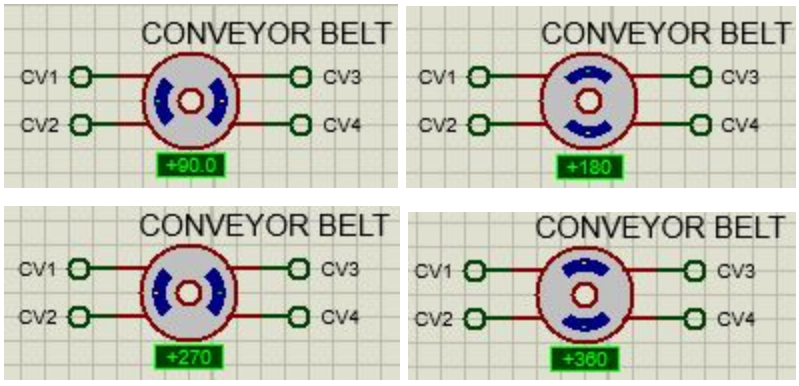


Fig. 64. Stepper Motor.

In the images shown above is shown the complete turn of the stepper motor, as it can be seen, each step is of 90 degrees.

- **Ice Gate**

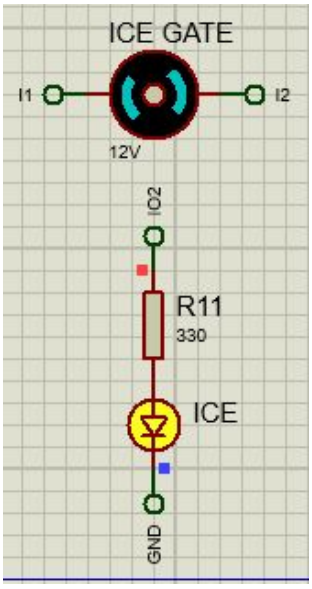


Fig. 65. Ice Gate.

In this image it is simulated the motor that opens the ice gate, the led simulates the pouring of the ice.

- **Alcohol Pouring**

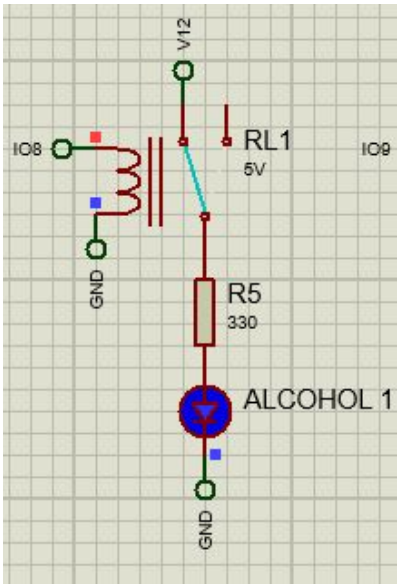


Fig. 66. Alcohol Pouring.

In the image shown above it is simulated the electrovalve that permits the pouring of the alcohol, the led shown simulates the pouring of the liquid.

- **Mixer Pouring**

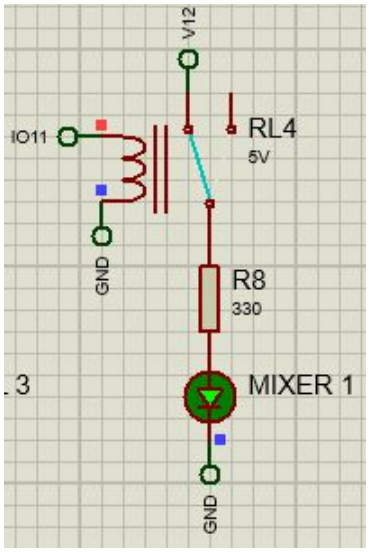


Fig. 67. Mixer Pouring.

In the image shown above it is simulated the electrovalve that permits the pouring of the mixer, the led shown simulates the pouring of the liquid.

C. Web application and the Server

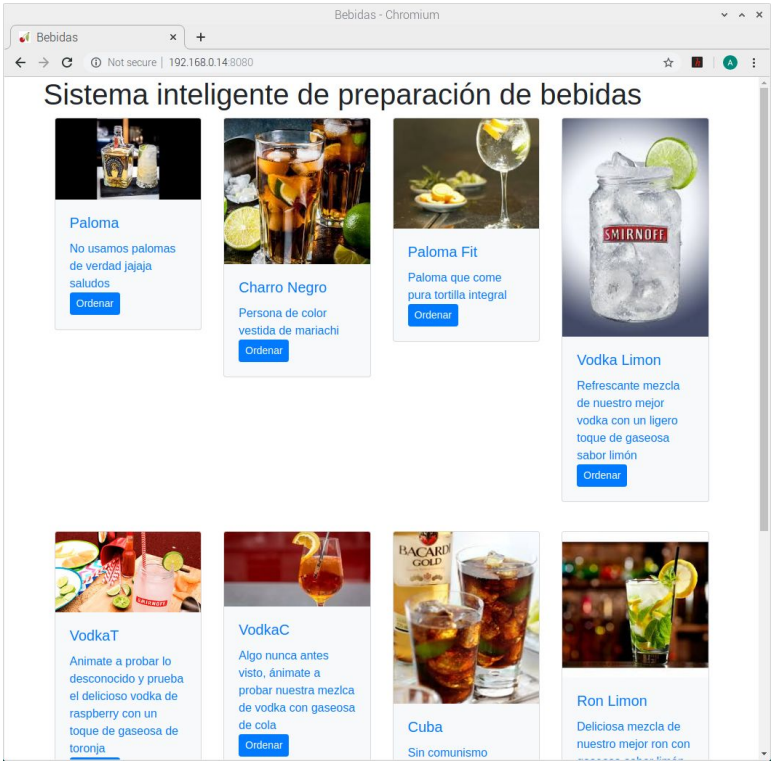


Fig. 68. Web page drink list.

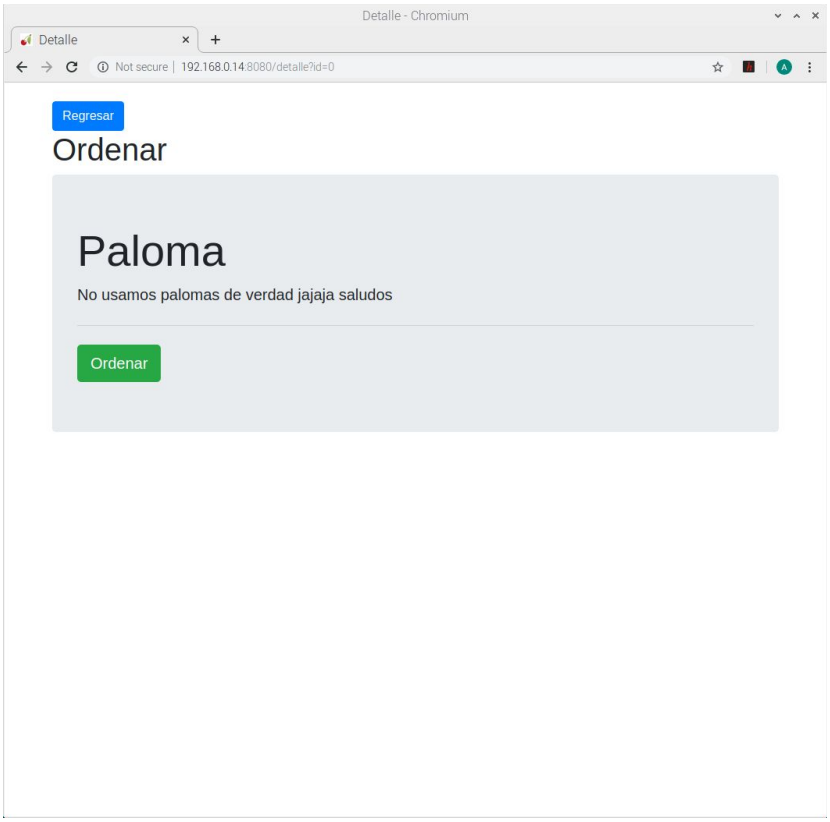


Fig. 69. Web page drink detail.

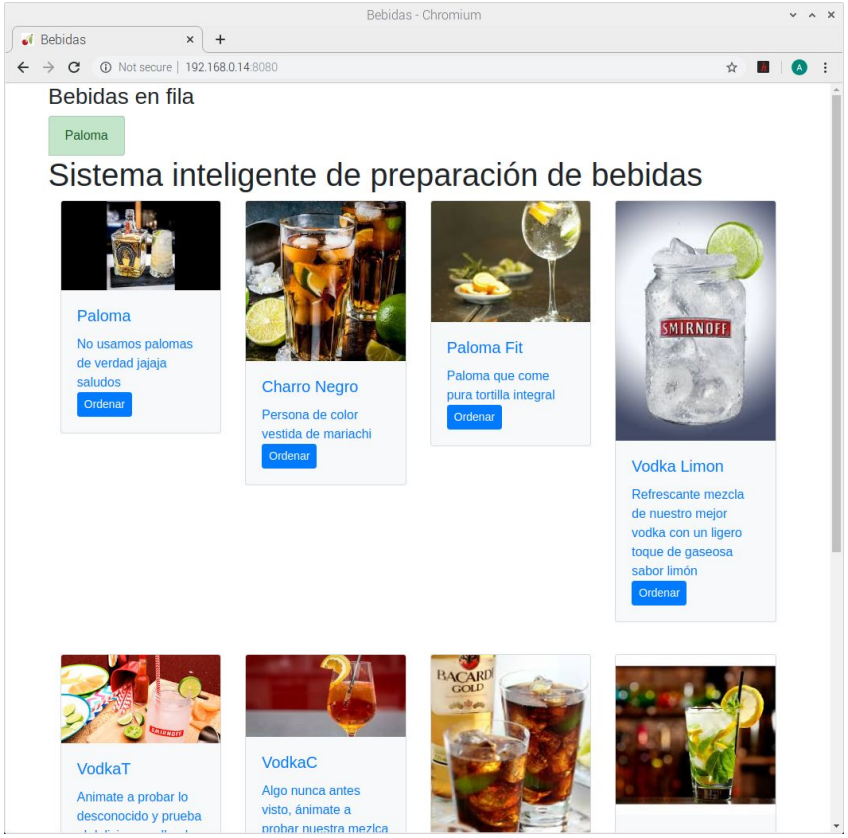


Fig. 70. Web page drink queue.

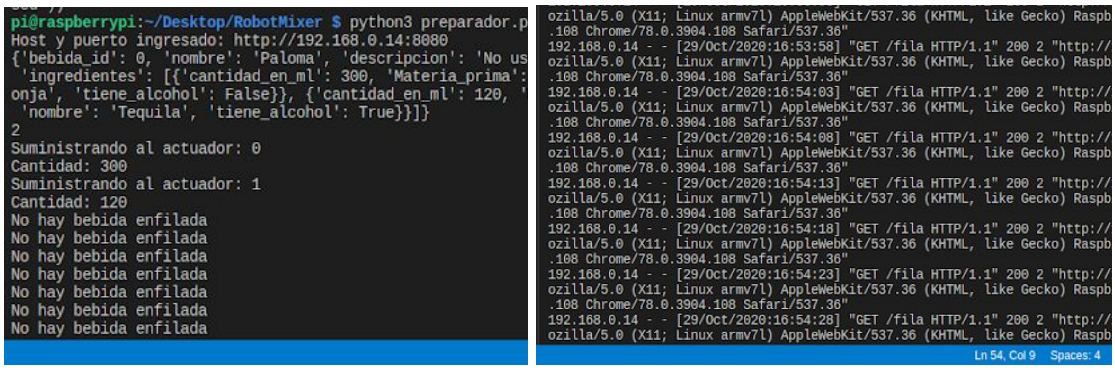


Fig. 71. Program to simulate robot studio drink preparation. Python script console of test program (left), Server console (right).

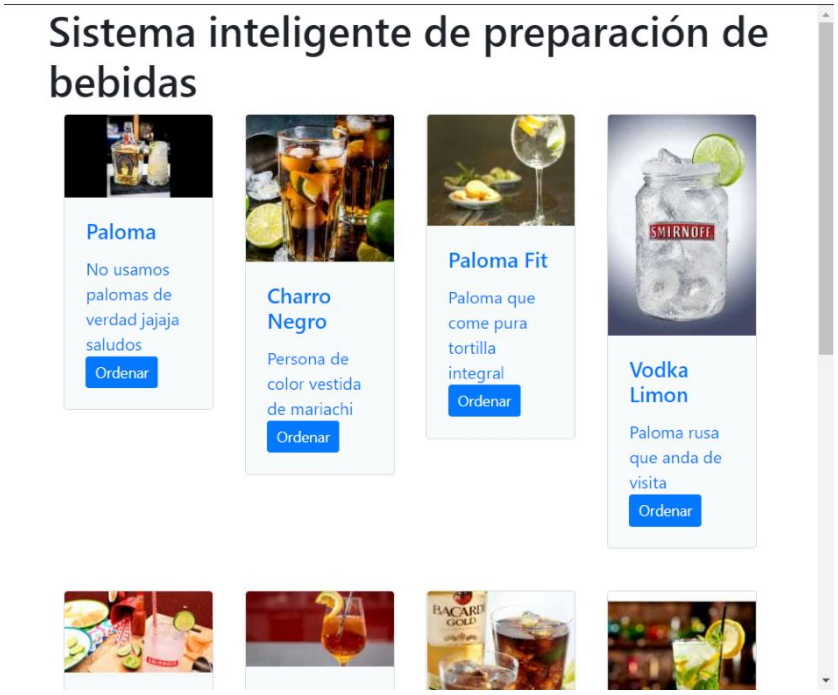


Fig. 72. Initial State of the Web Page.

Fig. 73 Drink added to the queue.

Fig. 74 Multiple beverages added to the queue.

```
192.168.0.20 - - [19/Nov/2020:22:45:58] "GET /fila HTTP/1.1" 200 2 "http://192.168
.0.20:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/86.0.4240.198 Safari/537.36"
192.168.0.20 - - [19/Nov/2020:22:46:03] "GET /fila HTTP/1.1" 200 2 "http://192.168
.0.20:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/86.0.4240.198 Safari/537.36"
192.168.0.20 - - [19/Nov/2020:22:46:08] "GET /fila HTTP/1.1" 200 2 "http://192.168
.0.20:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/86.0.4240.198 Safari/537.36"
192.168.0.20 - - [19/Nov/2020:22:46:13] "GET /fila HTTP/1.1" 200 2 "http://192.168
.0.20:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/86.0.4240.198 Safari/537.36"
192.168.0.20 - - [19/Nov/2020:22:46:14] "GET /detalle?id=0 HTTP/1.1" 200 1356 "htt
p://192.168.0.20:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
```

```
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
Bebida:  Paloma
Respuesta:  ok
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
No hay bebida enfilada
Traceback (most recent call last):
```

Fig. 75 Server's Terminal (left) and Drink's Queue (right).

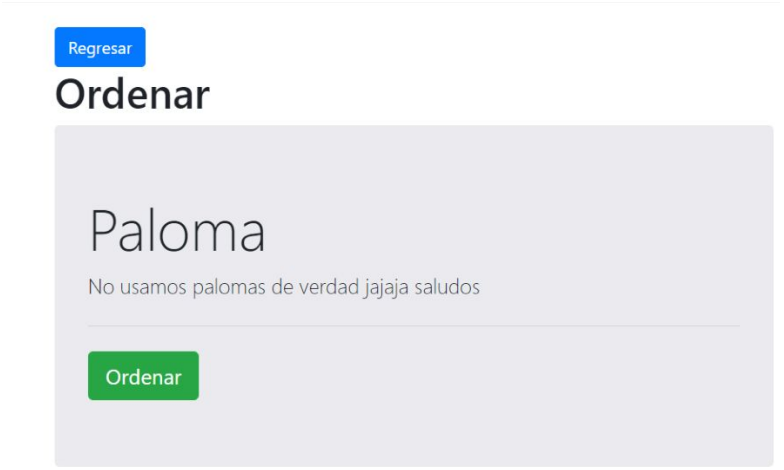


Fig. 76 Web Page with Drink's Details.

Fig. 77 Add a Drink to the Queue.

D. Simulation RobotStudio

Fig. 78. Home state of the system.

Fig. 79. The glass is taken off the rack by the right arm.

Fig. 80. Right arm puts the glass on the conveyor belt.

Fig. 81 The conveyor stops the glass on the first position to pour ice.

Fig. 82 The conveyor belt stops the glass on the second position to pour alcohol.

Fig. 83 The conveyor belt stops the glass on the third position to pour mixer.

Fig. 84 The conveyor belt stops the glass on the edge

Fig. 85 The left arm takes the glass.

Fig. 86. The second arm puts the glass on the bar.

E. Time Performance.

# Beverages	Expected Time	Simulation Time
1	<50s	44s
30	<25 min.	22 min.

According to performance requirement B.10, each beverage contains 40ml of alcohol and 100ml of mixer. Taking this into consideration, the total amount of beverages that can be made before refilling either one of the mixer containers is 30 drinks. For the alcohol containers the maximum amount of beverages that can be made before refilling is 75.

IX. References

- 3D CAD Model. (2020). Retrieved 18 November 2020, from https://grabcad.com/library/bar-stool-dwg-1
- Abhijeet. Solenoid Valve. (2011). Retrieved 2 November 2020, from: https://grabcad.com/library/solenoid-valve--1
- IRB 120 CAD models - Industrial Robots (Robotics) - IRB 120 - Industrial Robots (Robotics) (Industrial Robots from ABB Robotics). (2020). Retrieved 20 October 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-cad
- IRB 14000 YuMi Data - ABB's Collaborative Robot -YuMi (Industrial Robots from ABB Robotics). (2020). Retrieved 1 November 2020, from https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data
- Valli, Ricardo. Conveyor Belt with Stepper Motor. (2020). Retrieved 3 November 2020, from: https://grabcad.com/library/conveyor-belt-with-stepper-motor-1