Gebze Technical University
Computer Engineering

BİL396
Project Group 11
Report #3

Group Members
Abdulhakim Emre Artış
Cihan Can Ayyıldız
Esra Emirli
Hamza Yoğurtcuoğlu
İlkay Can
Melike Serra Kalyon
Oğuzhan Agkuş
Oğuzhan Şentürk
Sezer Demir
Selman Özleyen
Yusuf Can Kan

Consultant
Prof. Dr. Erkan Zergeroğlu

August 2020
Gebze / KOCAELİ

1. Ping Pong Ball Tracker and Bouncer

1.1. Project Definition

The aim of the group project is to construct a ping pong ball bouncer. The overall position on the plate (with respect to x and y coordinates) and an estimate of the height of the ball above the plate should be plotted in real-time both on a host computer and a remote mobile device (an android phone). The mechanism should also be able to move a ping pong ball placed on top of the glass table in a predetermined 2D trajectory.

1.1.1. Application

1.1.1.1. Stabilizing/Balancing the Ball

When the ball is placed on the plate, the system will locate the ball with the help of the camera and balance the ball while drawing the chosen shape.

1.1.1.2. Starting to Bounce the Ball

When the bouncing request arrives to the system, the system must place the ball properly for the bouncing movement. For example, the ball must be stabilized in the center of the plate.

1.1.1.3. Bouncing the Ball

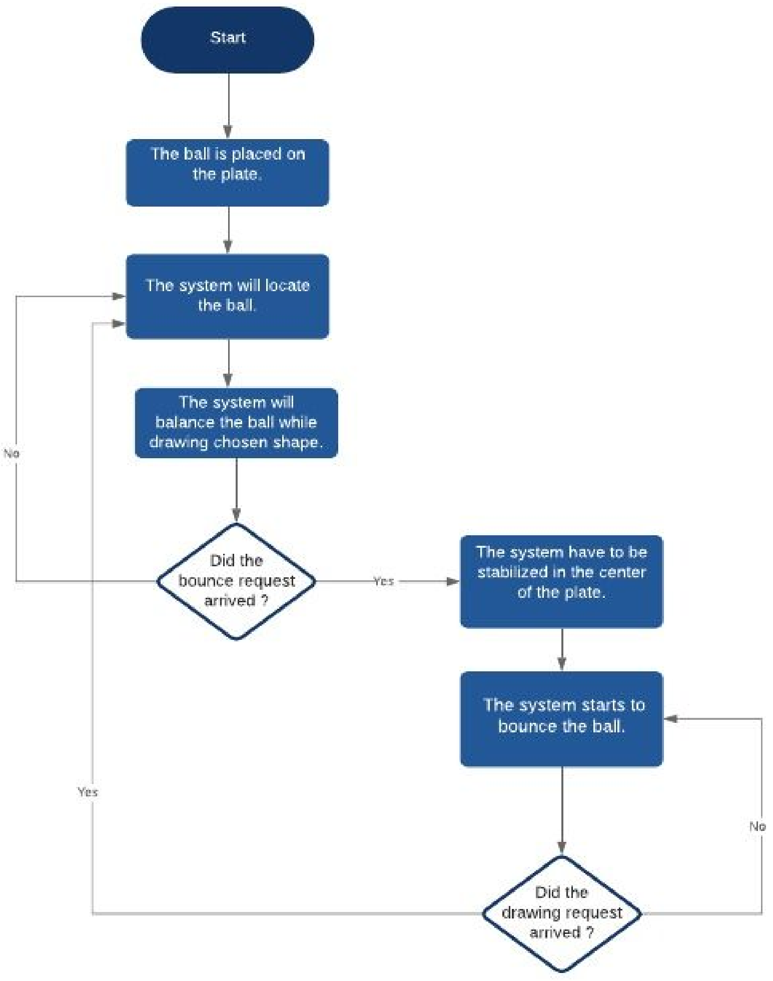
When the ball is balanced at the center of the plate, the system starts to bounce the ball until the next request.

1.1.1.4. Restabilizing/Rebalancing the Ball

When the ball stops bouncing, the system will balance it at the center again. This process repeats itself.

1.2. Example Usage Scenarios

Example usage of the program is provided with the following diagram.


Start
The ball is placed on the plate.
The system will locate the ball.
The system will balance the ball while drawing chosen shape.
No
Did the bounce request arrived ?
Yes
The system have to be stabilized in the center of the plate.
The system starts to bounce the ball.
Yes
No
Did the drawing request arrived ?

1.3. Modules of the Project

1.3.1. Definitions of Modules

1.3.1.1. Image Processing

This module uses OpenCV to process data from the camera. The data is analyzed and the radius, height, velocity, location attributes of the ball is calculated. These outputs are sent to the PID Controller Module.

1.3.1.2. Mechanical Design & Implementation

This module designs and implements the mechanical body of the project. The body consists of a transparent plate and servo-controlled four arms. The servos are controlled by a microcontroller. The microcontroller communicates with the controller device (PID Controller) via a serial port.

1.3.1.3. PID Controller

This module performs all mathematical and physical calculations. With the Image Processing Module sending the properties of the ball such as the velocity, the coordinate information calculates the movements of the servos in order to balance or bounce the ball. At the same time performs the same calculations for the Simulation Module. The communication with the mechanical body is done via a serial port.

1.3.1.4. Mobile and Desktop Applications

This module displays the 2D picture of the ball in real-time according to the coordinates and radius value received from the server computer. These data are received from the server via wifi. Real-time plotting will be shown both in mobile (Android) and computer (Linux and Windows). “

1.3.1.5. Simulation Module

This module will simulate the “Ping Pong Tracker and Bouncer”. The platform Webots simulator will be used. The simulation will include the objects (servo motors, ball, etc.). These simulations will be controlled with the C++ code.

In the previous report, simulation platform was specified differently, due to the complexity of the program, progress was highly slow. Since it is a threat for the project's deadline, OpenGL platform used for animating movements of servo motor.

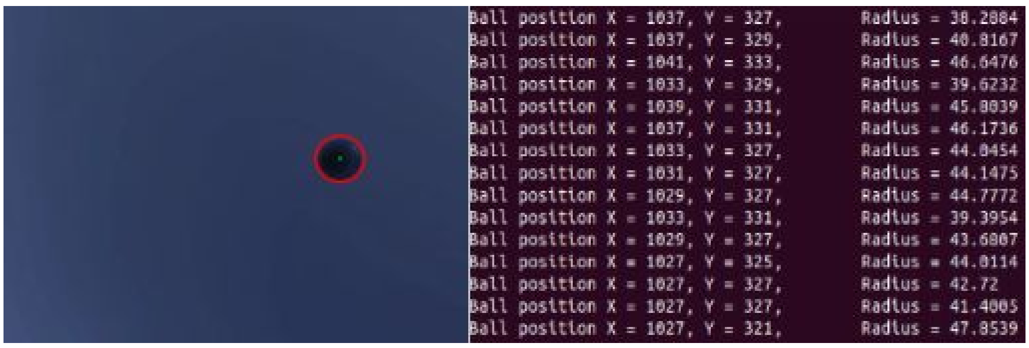
1.3.2 Implementation of Modules

1.3.2.1. Image Processing

In this module, the Hough Circle Transform algorithm is used. This algorithm is written in C++. In our algorithm, first the original image is converted to a hsv image. Then basic thresholding operations using OpenCV cv::inRange function is performed. An object based on the range of pixel values in the HSV colorspace is detected.

HSV (hue, saturation, value) colorspace is a model to represent the colorspace similar to the RGB color model. Since the hue channel models the color type, it is very useful in image processing tasks that need to segment objects based on its color. Variation of the saturation goes from unsaturated to represent shades of gray and fully saturated (no white component). Value channel describes the brightness or the intensity of the color.

We find the exact range of HUE values according to the color of the object. The saturation and value depends on the lighting condition of the environment as well as the surface of the object. Using this algorithm, we are able to reach the x and y coordinates and radius of the ball on the plate.



```
Ball position X = 1037, Y = 327,        Radius = 38.2884
Ball position X = 1037, Y = 329,        Radius = 40.8167
Ball position X = 1041, Y = 333,        Radius = 46.6476
Ball position X = 1033, Y = 329,        Radius = 39.6232
Ball position X = 1039, Y = 331,        Radius = 45.8039
Ball position X = 1037, Y = 331,        Radius = 46.1736
Ball position X = 1033, Y = 327,        Radius = 44.0454
Ball position X = 1031, Y = 327,        Radius = 44.1475
Ball position X = 1029, Y = 327,        Radius = 44.7772
Ball position X = 1033, Y = 331,        Radius = 39.3954
Ball position X = 1029, Y = 327,        Radius = 43.6807
Ball position X = 1027, Y = 325,        Radius = 44.0114
Ball position X = 1027, Y = 327,        Radius = 42.72
Ball position X = 1027, Y = 327,        Radius = 41.4005
Ball position X = 1027, Y = 321,        Radius = 47.8539
```

Image 1. Detection Location , Radius of Ball with Camera Output Sample in Simulation

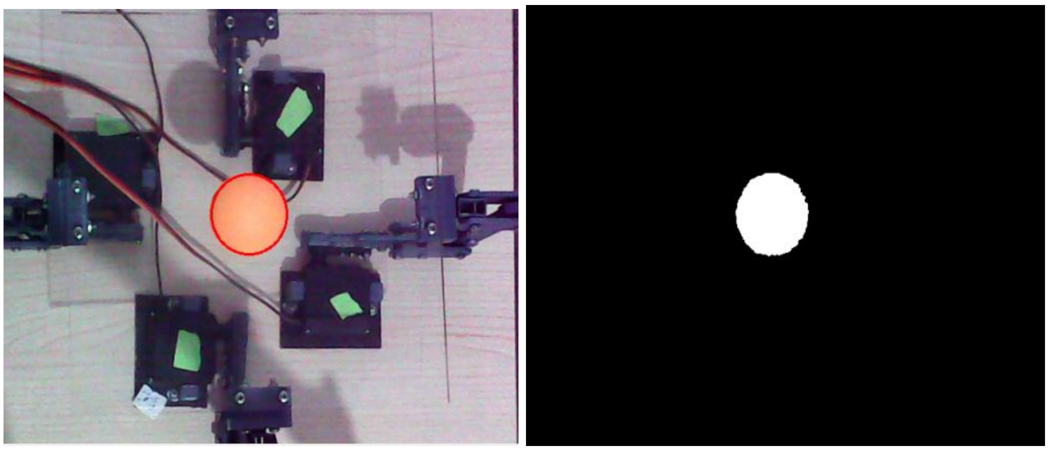


Image 2. Detection Location , Radius of Ball with Camera Output Sample in Embedded System

Finally, since we could not place our camera under our embedded system, we decided to take images from the top of the platform.

1.3.2.2. Mechanical Design & Implementation

In the first stage, the test of the servo operation was performed. We decided the counter period to be 20000. It means at 20000 the pwm duty cycle with period 20 milliseconds or 50 Hz frequency.

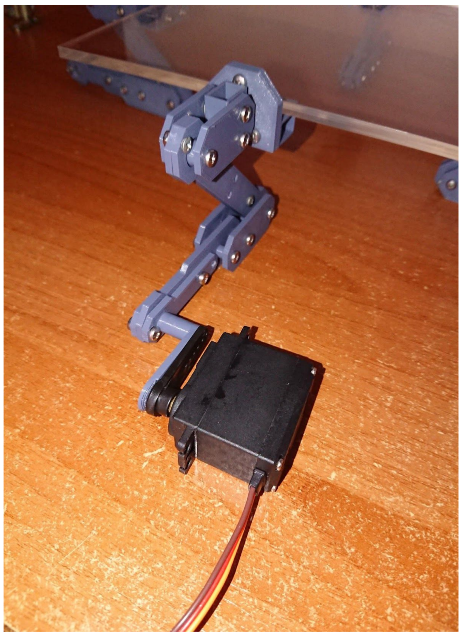


Image 3. Servo Motor and 3D Arm Connection

The skeleton of the platform is provided with a 3D printer. Its movement is provided by 4 servo motors placed on this device. There are holes on the frame for fixing the arm.

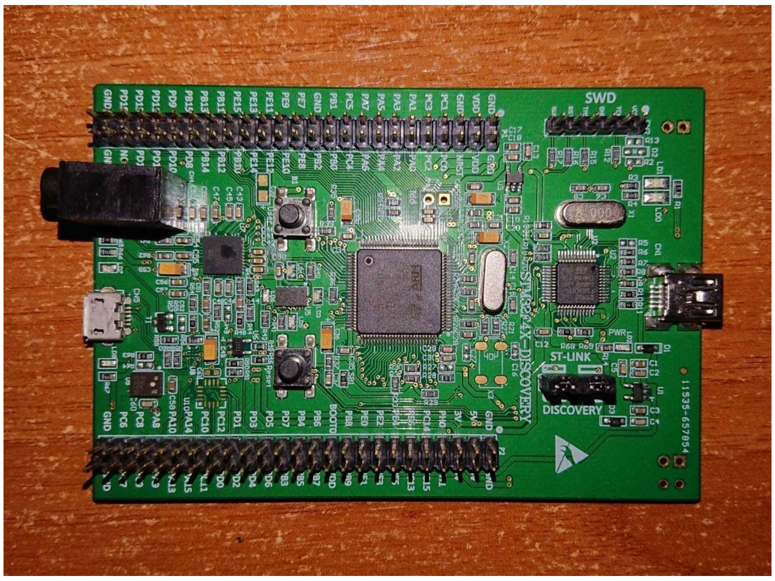


Image 4. STM32F4 Discovery

STM32 board takes the angle values for 4 servos from the server via serial port. It transmits these angle values to servo motors.

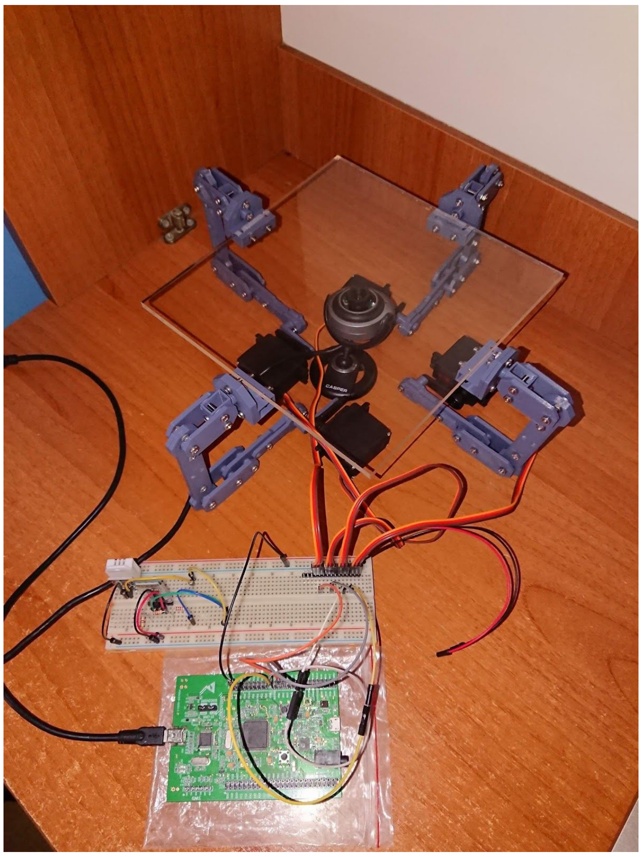


Image 5. First Hardware Prototype

As a result of the completion of the necessary works of all modules, the hardware was combined and its final form was achieved. All hardware products such as STM32 Discovery, servo motors, camera, 3D arms are placed and tested on the platform in the figure.

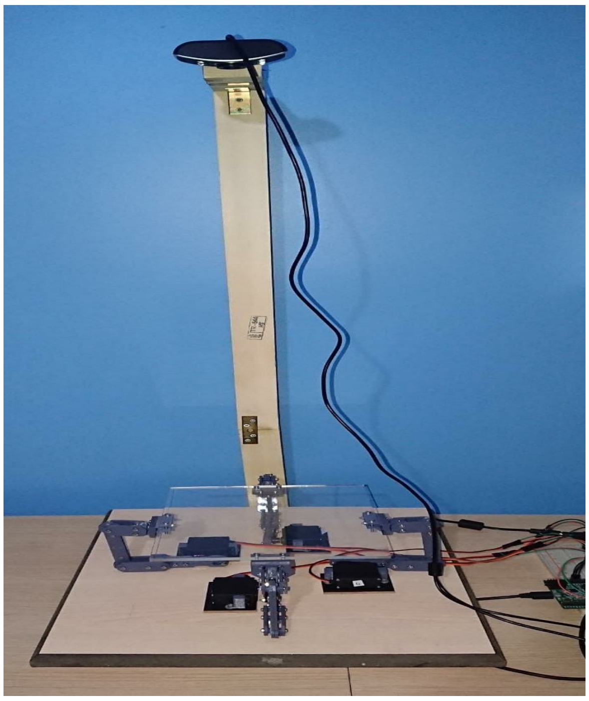


Image 6. Final Hardware

Because the camera does not have wide viewing angle and it does not cover all the plate from below the plate, we placed the camera above the mechanical part.

Required Components

Name of the Product	Price	Purpose of Usage
STM32F4 Discovery Board	125.00 TL	To transmit the movements of servo motors
Plexiglass	30.00 TL	Platform where the ball is bounced and balanced
4 x MG996 Servo Motors	100.00 TL	Providing movement to the 3D arms according to the incoming angle signal
Camera	150.00 TL	To be used for taking images
4 x 3D Printer Arms	50.00 TL	Providing movement to the platform according to the servo angle

1.3.2.3. PID Controller

A PID controller compares the signal coming from the output with the feedback (reference) signal and an error occurs. According to this error, the PID controller tries to minimize the error and makes an effect and sends it to the output. In this way, errors are determined by continuous feedback from output to input until the error is minimized and the error is reduced by sending the controller effect to the output.

$$u(t) = \underset{\text{Proportional Control.}}{K_P e(t)} + \overset{\text{Integrating Control}}{K_I \int_0^t e(\tau)d\tau} + \overset{\text{Derivative Control}}{K_D \frac{de(t)}{dt}}$$

Proportional Control

Proportional control is the easiest feedback control system. In proportional control, the signal of the controller is multiplied by the error signal. The ideal Kp value varies from system to system.

- *error = setpoint – current_position_ball*
- *P_contr = kp * error*

The proportional control system was not sufficient to reach the desired value.

Integral and Derivative control was added to stabilize the system.

Integrating Control

The accumulated offset caused previously in the system is corrected by the sum of errors over the finite time. The signal is created with the same field on the input signal to zero the error.

- *error_sum += error*
- *i_contr = ki * error_sum*

Derivative Control

The derivative control calculates the time variation of the error in the system. Future changes are predicted from past changes in the system. The effect of derivative control varies with the change of error over time.

- *d_contr = kd * error_diff*

PID outputs are computed for 4 servos. The stability of the platform is checked according to these outputs. The signals obtained are collected in an array 4 and the angles of the servo motors are changed separately by taking the average of the previous signals.

Balance

After setting the preferred points, our algorithm will try to adjust the angles of the servo motors to move the ball to the preferred position. There are some config files to adjust the P, I, and D weights for an axis.

Circle Drawing

In our system, we have an array of points which we can traject the ball upon. In our case, we used an array with 4 points which are the target points for the ball to balance on after a specific period of time the new target point will be set. At a time the system will have only one target point which will be changed after the determined period of time.

1.3.2.4. Mobile and Desktop Applications

The current x, y, and z coordinates of the ping pong ball are sent from the Host Desktop Application via UDP broadcasting to the Desktop and Mobile Application. The module receives the values and displays the data in the desktop and mobile applications.

The application displays the ping pong ball's current location. The interface includes 3 graphs:

- 2D graph where the x and y coordinates of the ball are represented. The yellow indicator shows the coordinates of where the ball hit the ground last. The green indicator shows the current coordinates of the ball.
- 2D graph where x and z coordinates of the ball are represented. The green indicator shows the current coordinates of the ball.
- 3D graph where x, y, and z coordinates of the ball are displayed. The previous 10 locations are kept and displayed in order to see the movements clearly. This feature is available as a desktop application.

The ping pong ball can be tracked from the user via these graphs.

Sample interface screenshots of ping pong ball’s movements are provided below.

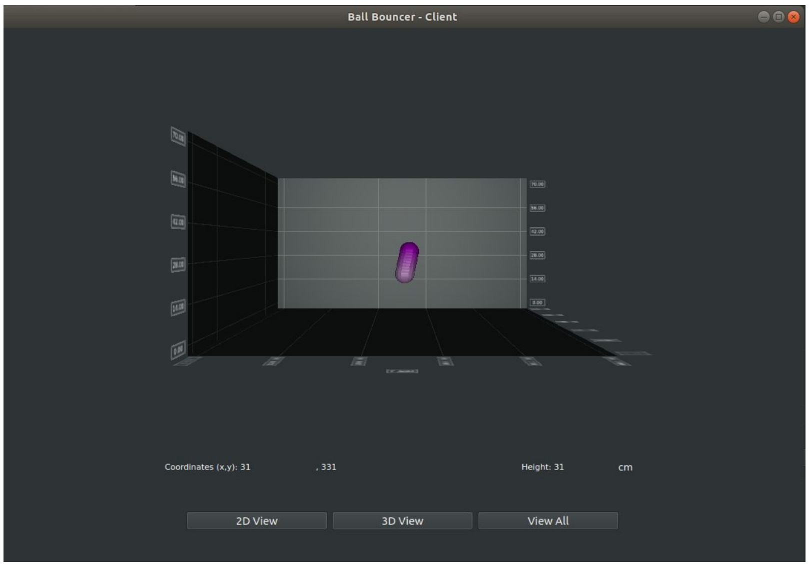


Image 7. The 3D Graphics of Ball in the Desktop Application Screen.

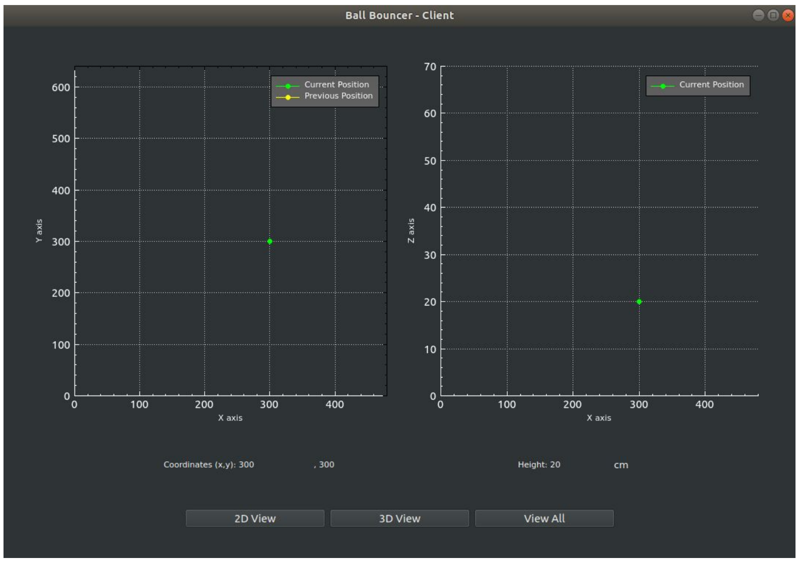


Image 8. The 2D Graphics of Ball in the Desktop Application Screen.

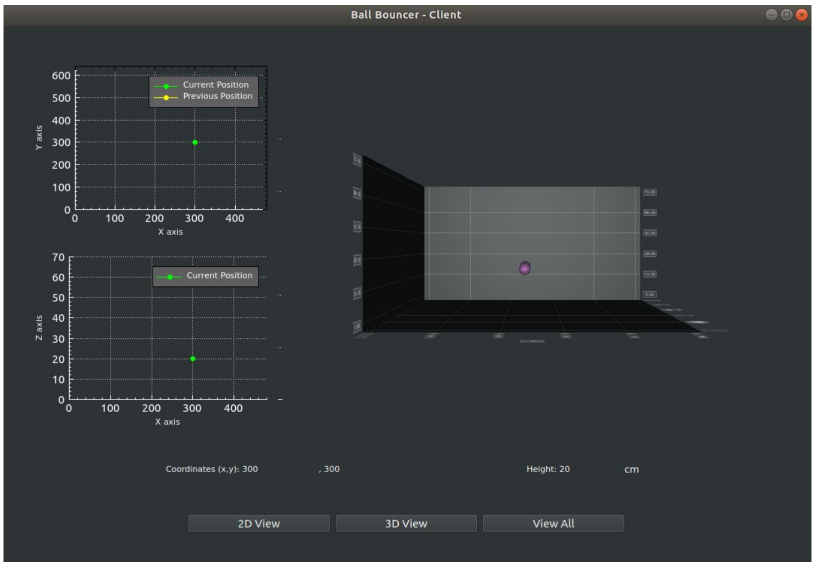


Image 9. The Graphics of Ball in the Desktop Application Screen.

The Desktop Application is written in C++ using the QT 5.15.0 development platform.

The Android Application is written in Java using the Android Studio development platform.

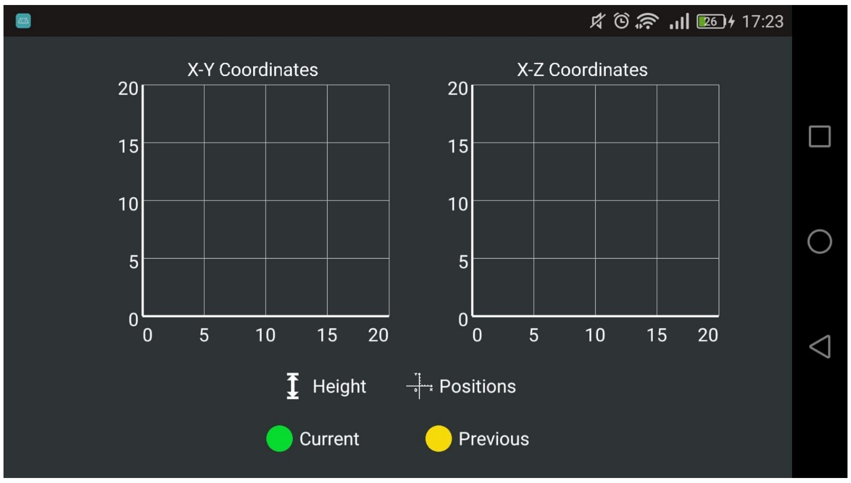


Image 10. Mobile Application Screen of Before Server Connection

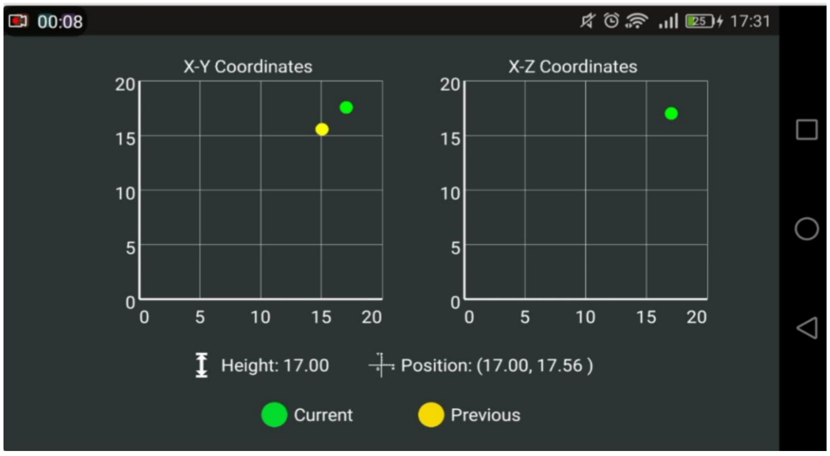


Image 11. Mobile Application Screen of After Server Connection

Server Desktop Application

All modules are integrated to the Qt Server Application. Since Qt supports OpenCV and OpenGL, we have successfully integrated the simulation, the image processing code, the PID code, UDP connection, Serial Port connection to the same platform.

The Server Desktop Application does the following steps in the background:

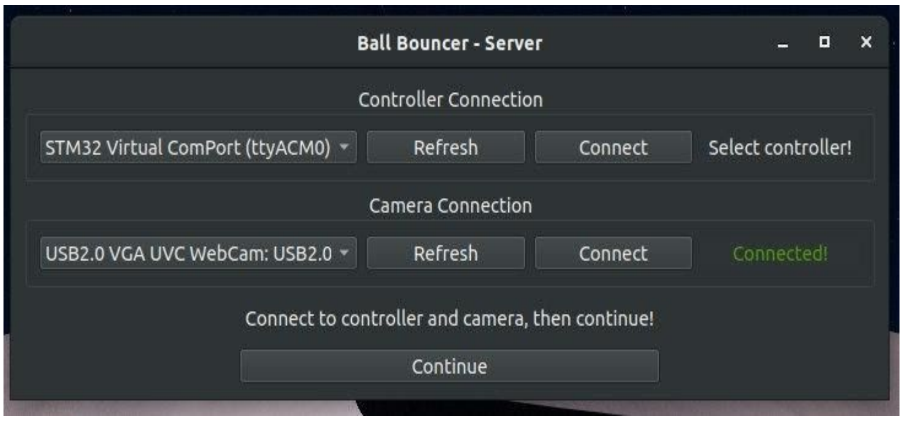
- Processes the image received by the camera, finds the x, y, z coordinates of the ball. The image processing is done by an individual thread. When the coordinates are calculated a signal is sent to the main process.
- When the coordinates are received from image processing, it sends the values to the graphs in the interface. The values are then sent via UDP broadcast to Mobile and Desktop Applications. When the values are then received, the position of the ping pong ball is displayed in the applications.
- The position of the ball also is sent to the PID module, and the corresponding arm angles are calculated.
- The calculated arm angles are then sent to the STM card using Serial Port connection and also is sent to the Simulation.
- The STM card adjusts the motors according to the angle values.

These steps are executed as a cycle that make up the system.

The Qt Server Desktop Application Interface

The Qt Server Interface start-up window includes the following:

- A drop down menu to select the STM card.
- A drop down menu to select the Camera.

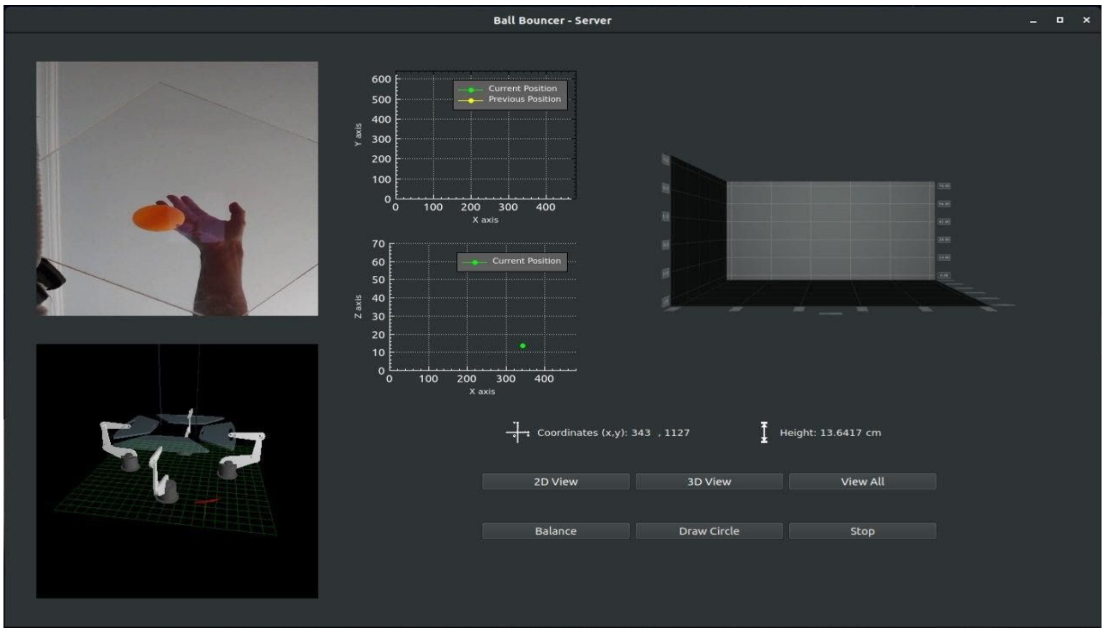


STM32 Virtual ComPort (ttyACM0)
Select controller!
USB2.0 VGA UVC WebCam: USB2.0
Connected!
Connect to controller and camera, then continue!

Image 12.Devices Connection Page - Server Application

If the connection is successfully done, user can continue to the next window. The next window includes the same graphics of the Desktop Application and additionally includes the following attributes:

- The Simulation of the motor, arms, and the glass plate.
- The camera image received from the platform.



Coordinates (x,y): 343 , 1127
Height: 13.6417 cm

Image 13. Embedded System Control Panel

1.3.2.5. Simulation Module

In this module, the main goal is simulating the real mechanism. For this, the main need is designing the hardware and combining this equipment properly. For the hardware part, the system needs 4 motors, a plate, one camera, and arms for controlling the plate angle.

The hinge joint structure in Webots was used for the servo motor. This is done by adding the rotational motor feature.

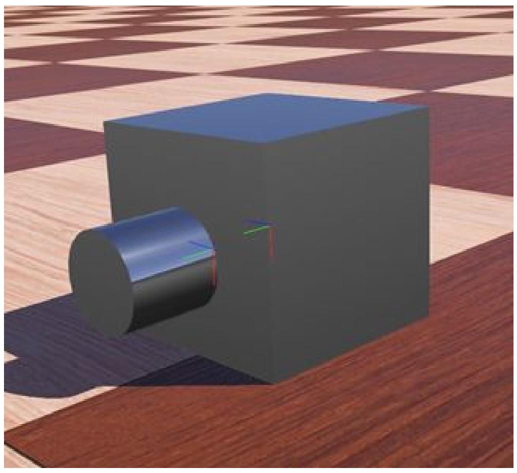


Image 14. Simulated motor

The controls of the motors are done with the C++ code. The engines can be moved as desired with the velocity parameter given to the cylinder on the engine. These movements can be limited over the assigned value. In this way, the system sets the angle for the plate with these controls.

For linking the platform and motors with each other, one arm is linked with the motor cylinder of the motor, and the other arm is linked with the first arm and main plate for the controlling platform. The two arms linked with each other with one hinge point, in this way physical movement is transmitted between arms and motor plate.

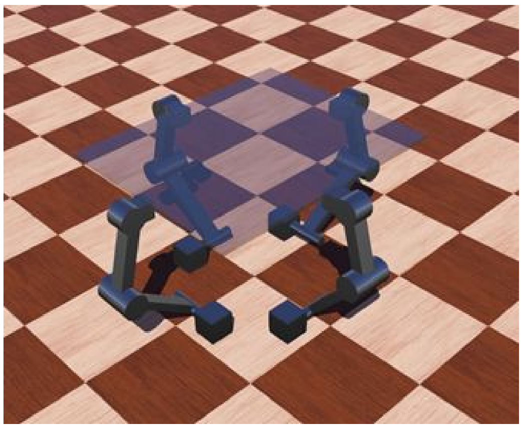


Image 15. Ball bouncer

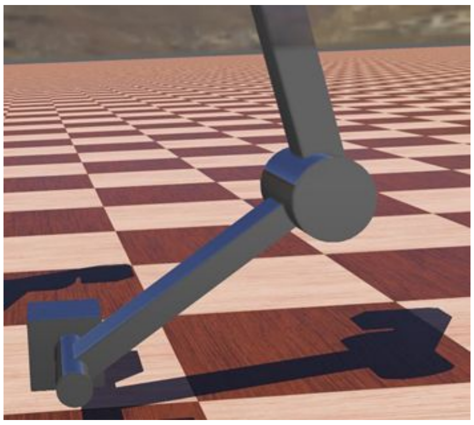


Image 16. Arm and motor connection

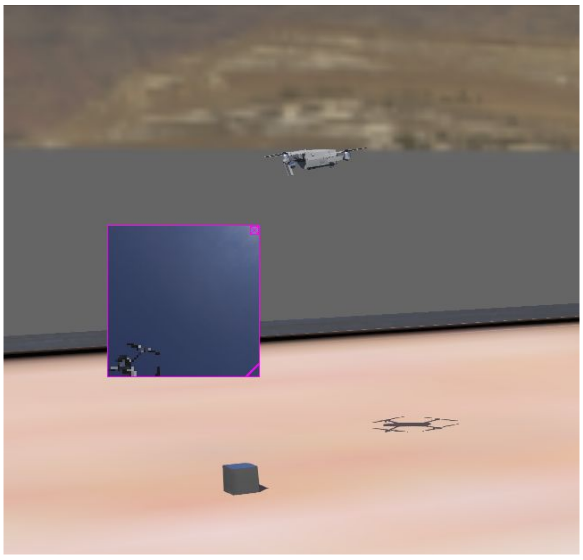


Image 17. Camera

To find the position of the ball, Webots takes the images of the plate just like the real hardware and transmits the images to OpenCV. Webots has its own camera object that can be customized by the user. There is a frame in the simulation which shows the camera view. After enabling the camera in the controller, camera views can be saved using the wb_camera_save_image(..) method. The image is saved in a file in either PNG or JPEG format. The image format is specified by the filename parameter. If filename is terminated by .png, the image format is PNG. If filename is terminated by .jpg or .jpeg, the image format is JPEG. Other image formats are not supported. The quality parameter is useful only for JPEG images. It defines the JPEG quality of the saved image. The quality parameter should be in the range 1 (worst quality) to 100 (best quality). Low-quality JPEG files will use less disk space. For PNG images, the quality parameter is ignored.

The return value of the wb_camera_save_image function is 0 in case of success. It is -1 in case of failure (unable to open the specified file or unrecognized image file extension).

Webots has a strange hierarchy which makes it impossible for us to simulate our system. So we tried Unreal Engine and had a very fast progress. But in the end we couldn't connect the plate with the arms and make it act as we expected.

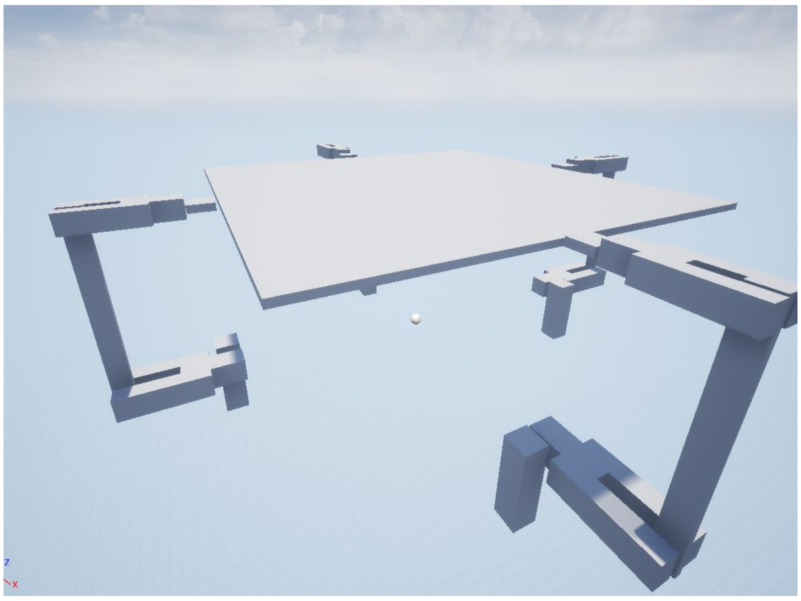


Image 18. Ball bouncer

As we continued our research, we came to a conclusion to use OpenGL for the simulation. Since Qt supports OpenGL, we have successfully integrated the simulation to the same platform.

OpenGL (Open Graphics Library is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics.

The motor, arms, and the glass plate are rendered using the Blender program. Each part is exported as an .obj and .mtl file. These are then used in OpenGL for simulating the system.

When the program calculates the angles that will be sent to the STM also sends the angles to the simulation. When these angle values are received, the program simulates the movement of the arms, thus lifts the plate.

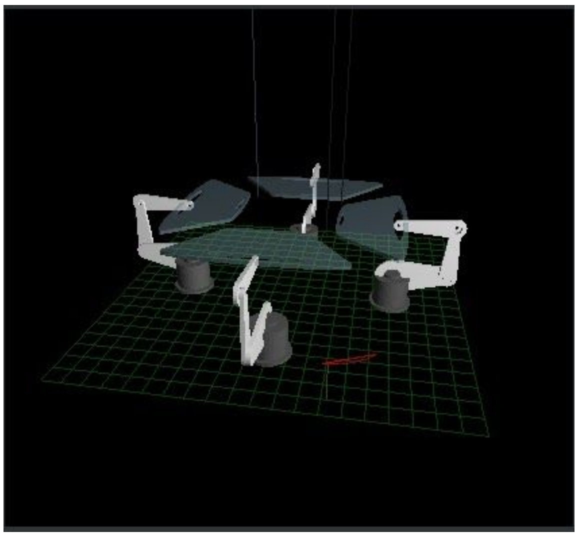
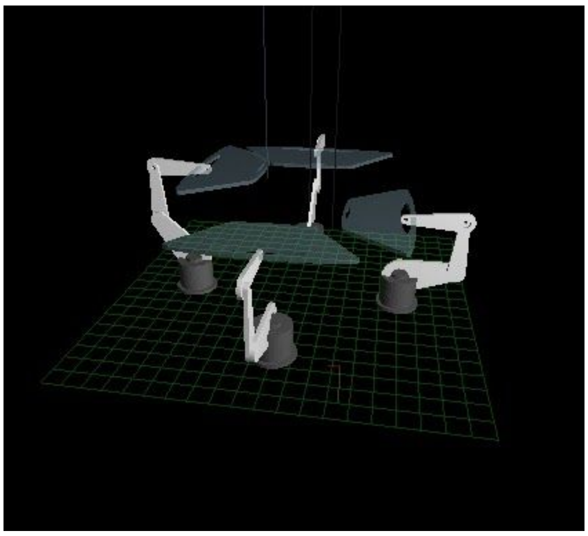



Image 19. Initial Position Image 20. Arms with angles 50, 50, 10, 10 respectively

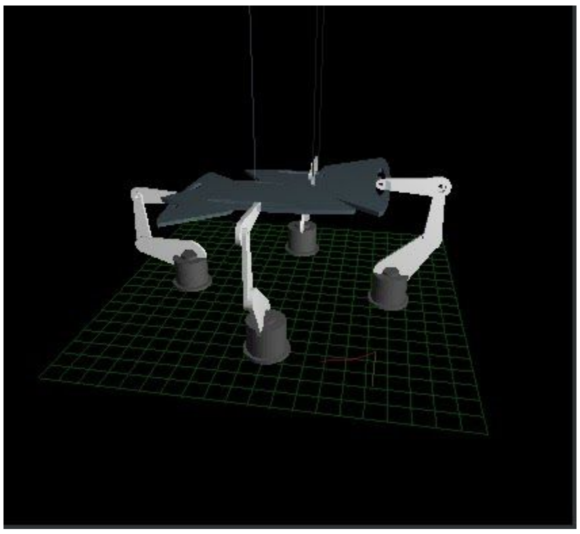
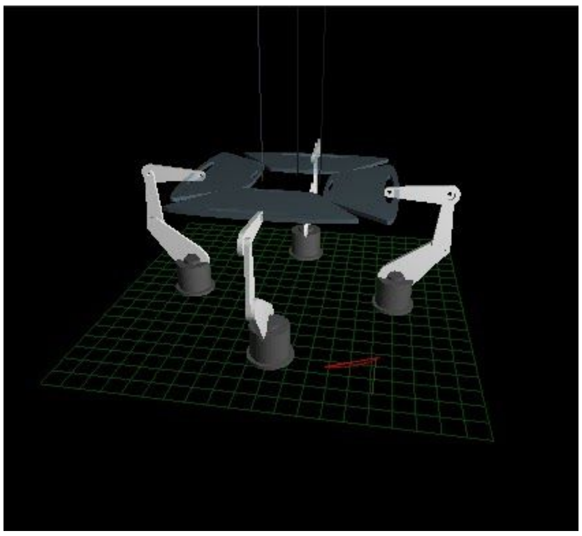
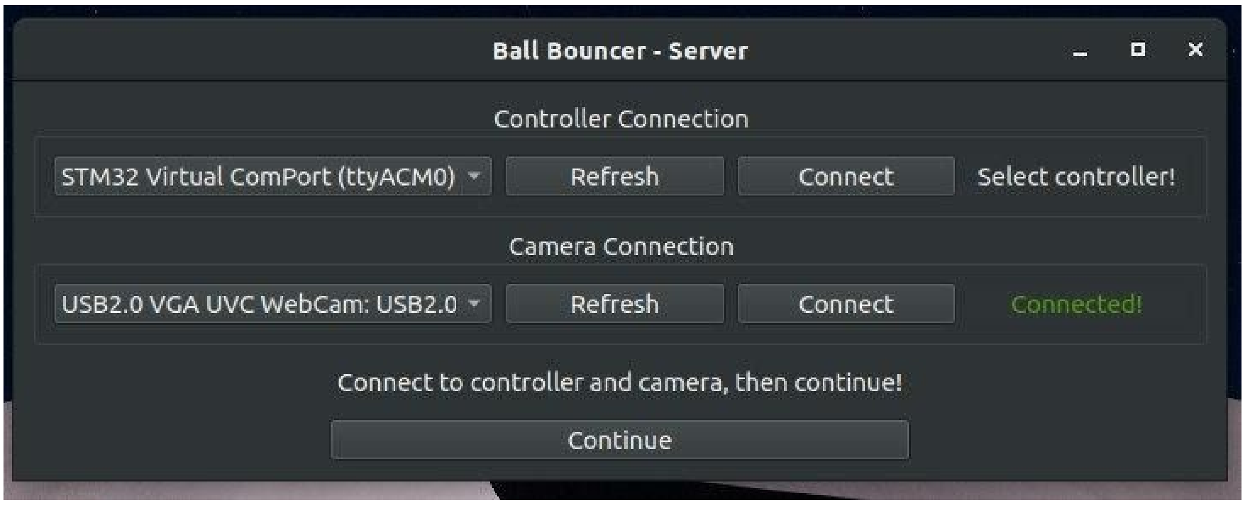



Image 21. Arms with angles 50, 50, 10, 10 respectively Image 22. Position of all arms with 50 degree angles

1.4. Setup and Run

The Server Computer

Camera and stm communication cable must be connected to the host computer. Camera and stm communication port must be selected over the server application.



Ball Bouncer - Server

Controller Connection

STM32 Virtual ComPort (ttyACM0) | Refresh | Connect | Select controller!

Camera Connection

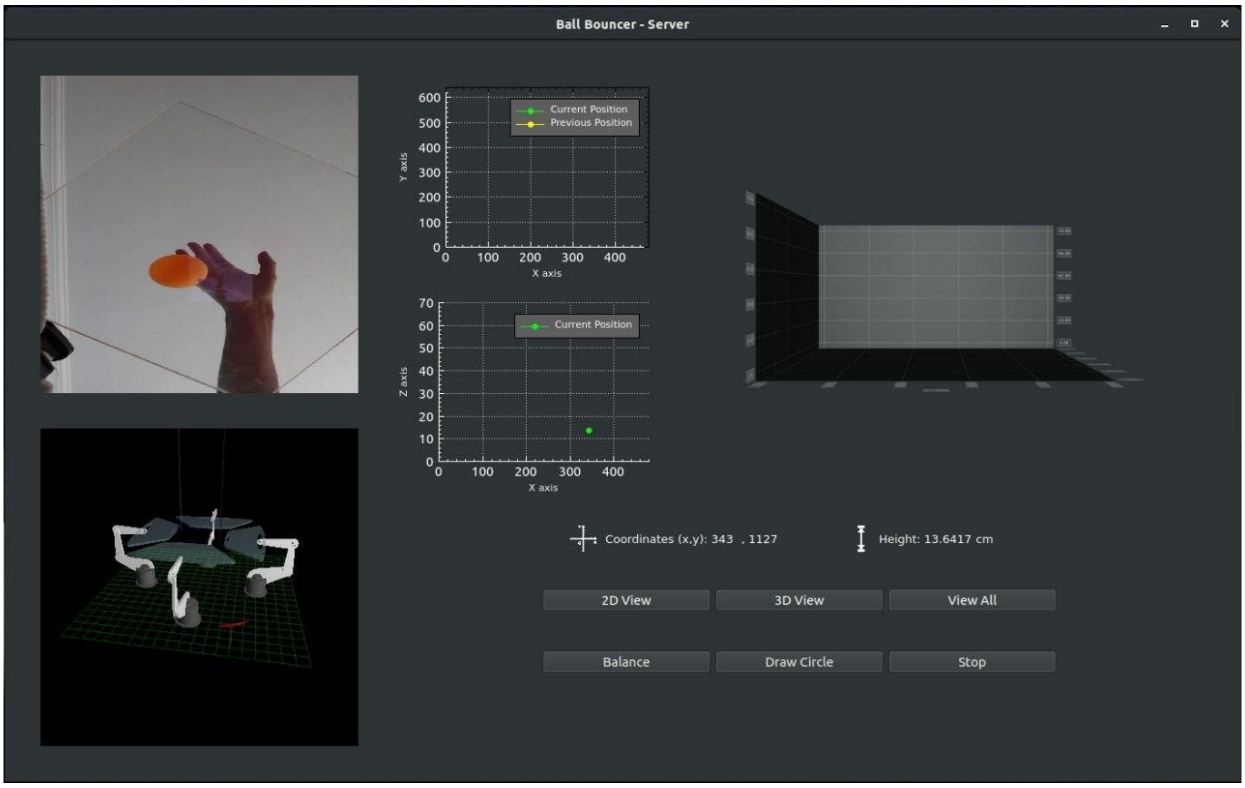
USB2.0 VGA UVC WebCam: USB2.0 | Refresh | Connect | Connected!

Connect to controller and camera, then continue!

Continue

Image 23. Devices Connection Page - Server Application

After the connections are successful, ping pong ball balance and circle commands can be given to the embedded system via the server application. If you want to stop the system, the stop button should be used. Then you can switch to the other command you want.



Coordinates (x,y): 343 , 1127

Height: 13.6417 cm

Image 24. Embedded System Control Panel

You can follow the camera and simulation live from the server application.

The Mobile Application

Install the application apk file to watch the system live from the mobile application. Then connect to the server computer's network.

The Desktop Application

Obtain the executable file. You can watch live after connecting to the network of the server computer from your Linux or Windows operating system.

The Embedded System

Just plug in the system's adapter and follow the steps in the server application.

1.5. Conclusion

The main purpose of our project was to balance a ping-pong ball, to draw a shape with this ball and to bounce this ball on a plate. In addition to producing hardware with this capability, it was also aimed to make user applications in order to observe the movements of this hardware and the ping-pong ball by the user.

All modules are integrated to the Qt Server Application. We have successfully integrated the simulation, the image processing code, the PID code, UDP connection, Serial Port connection to the same platform.

1.6. Members of Modules

Image Processing	Mechanical Design & Implementation	PID Controller	Mobile and Desktop Application	Simulation
Sezer Demir	Oğuzhan Agkuş	Sezer Demir	Oğuzhan Agkuş	Yusuf Can Kan
İlkay Can	Emre Artış	Emre Artış	Cihan Can Ayyıldız	İlkay Can
Cihan Can Ayyıldız	Yusuf Can Kan	Yusuf Can Kan	Esra Emirli	Sezer Demir
Esra Emirli	Hamza Yoğurtcuoğlu	Selman Özleyen	Melike Serra Kalyon	Emre Artış
Selman Özleyen	Cihan Can Ayyıldız	Oğuzhan Agkuş	Oğuzhan Şentürk	Selman Özleyen
Hamza Yoğurtcuoğlu		İlkay Can		Esra Emirli
Melike Serra Kalyon				Cihan Can Ayyıldız
Oğuzhan Şentürk				Melike Serra Kalyon

1.7. System Components

- 1x STM32 Discover Microcontroller
- 4x Servo Motors
- 1x Power Supply
- 1x Camera
- 1x Linux Distributed Computer with OpenCV installed on it
- This Custom Simulation Application (made with OpenGL)
- 1x Plexiglass
- Plastic and metal components for body

1.8. Risk Analysis

The main risks we may encounter in this project is to turn the simulated system into hardware. Since we have a limited time to make the mechanical design, we may experience difficulties while synchronizing hardware with the simulation calculations. Making small mistakes on our calculations may cause this.

1.9. Improvable Things

- In the simulation part, we couldn’t assemble the plate and 4 servo motor arms. We tried it in 3 different simulation software (Webots, Unreal Engine, Gazebo) but it didn’t work. When we did it, the plate behaved absurdly.

- We use a 30 fps camera in the project. With 60 or 120 fps camera simulation could be more smooth.

- Servo motors those we used are a little bit weak on tork. So the arms of the machine move slower than we expected.

2.0. Additional Notes

Fatih Dural, who is a member of our team, informed us that he could not attend the project work any longer due to personal problems and left the group.