CSE396 Project Report

BALL BOUNCER

GROUP 1

Gökçe Erer
İhsan Ovalı
Murat Yalçın
Sefa Nadir Yıldız
Zeynep Nazire Yüksel
Emirhan Karagözoğlu
Mohammad Ashraf Yawar
Ömer Bozaba
Samet Yurt

Supervisor: Prof. Dr. Erkan ZERGEROĞLU

Department of Computer Science and Engineering

Gebze Technical University

August 2020

Contents

1 Introduction **iii**

2 General Scheme and Modules **iv**
- 2.1 Mechanical Subsystem v
- 2.2 Desktop Application vi
- 2.3 Mobile Application vii
- 2.4 Communication Handling viii
- 2.5 Sensor Subsystem Design ix
- 2.6 Image Processing x
- 2.7 Simulation Design xi

3 Demo **xiii**

List of Figures

1 Top View of System iv
2 Side View of System iv
3 Desktop Before Connect vi
4 Desktop Connect to URL vi
5 Desktop Output Test vi
6 Mobile App Opening Page vii
7 Mobile App Connect Page vii
8 Communication Handling Files viii
9 Communication Output Test ix
10 Image Processing Test x
11 Simulation Demonstration xi
12 Simulation Connect Server xii
13 Simulation Front and Back Camera xii

List of Tables

1 Crew List iii
2 Bought Products List v

1 Introduction

This project provides a mechanism which lets the user to bounce the ping pong ball and move the ball in the desired trajectory while keeping it balanced.

The project consists of the main hardware, a mobile application, a desktop application, and several submodules that keep them organized. One of the purposes of this project is to show the position of the ball and its predicted height information through mobile and desktop applications in real-time. Also, mobile and desktop applications are going to let the user control the hardware which influences the ping pong ball.

In the purpose of developing and testing the project in the absence of hardware, a simulation environment provided.

Mobile Application	**Desktop Application**	**Communication Handling**
Mohammad Ashraf Yawar	İhsan Ovalı	Mohammad Ashraf Yawar
Emirhan Karagözoğlu	Murat Yalçın	Emirhan Karagözoğlu
Ömer Bozaba	Zeynep Nazire Yüksel	Sefa Nadir Yıldız
Zeynep Nazire Yüksel	Mohammad Ashraf Yawar	Gökçe Nur Erer
Samet Yurt	Samet Yurt	

Sensor Subsystem Design	**Image Processing**	**Simulation Design**
İhsan Ovalı	Murat Yalçın	Ömer Bozaba
Murat Yalçın	İhsan Ovalı	Gökçe Nur Erer
Gökçe Nur Erer	Sefa Nadir Yıldız	Sefa Nadir Yıldız
	Zeynep Nazire Yüksel	
	Mohammad Ashraf Yawar	

Table 1: Crew List

PS: *All crew included in the Mechanical Subsystem Design, because of that it is not included in the table.*

2 General Scheme and Modules

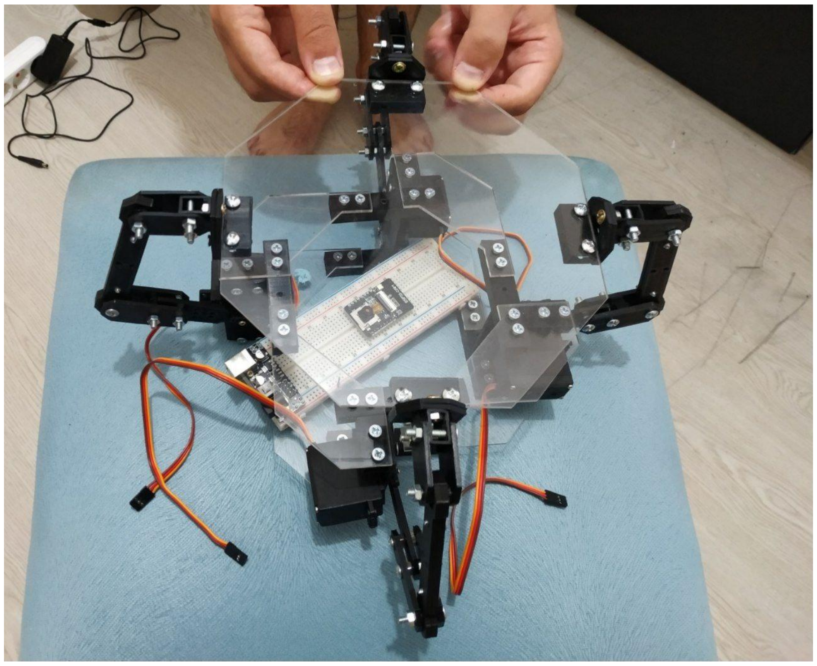


Figure 1: Top View of System

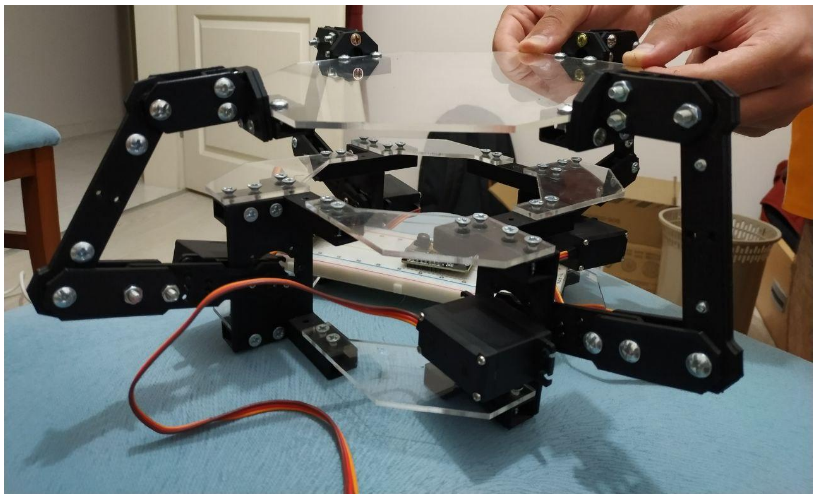


Figure 2: Side View of System

2.1 Mechanical Subsystem

The design of the mechanical system will be done in this module. The materials in the table below are bought:

Product	**Price**
ESP32 CAM	122₺
FT232BL (USB - UART) I.C.	42₺
4x MG995 Servo	124₺
LM2596 DC-DC Buck Converter Step-Down Power Module	10₺
12V 3.3A AC/DC Power Adapter	55₺
3.3V/5V Breadboard Power Supply Module	6₺
Total	359₺

Table 2: Bought Products List

The mechanical legs printed by a 3D printer were strengthened with various screws and bolts. A 3mm thick Plexiglass was used for the bounce area of the ball.

The operation of the mechanism could not be tested at this time, as the power required to lift the arms to which 4 servo motors are connected at the same time could not be supplied. Accordingly, the PID Control parameters to keep the ball in balance could not be determined, and algorithms that would provide operations such as ball jumping could not be developed. We hope that this problem will be resolved until the hardware delivery.

2.2 Desktop Application

In this module, the Qt GUI framework and C++ used during the development process. This app has a Connect section for user to enter the URL which provides communication. The position data sent by the communication handling module via the URL with using HTTP were converted into a circle plot and shown as real-time on QtWidget.

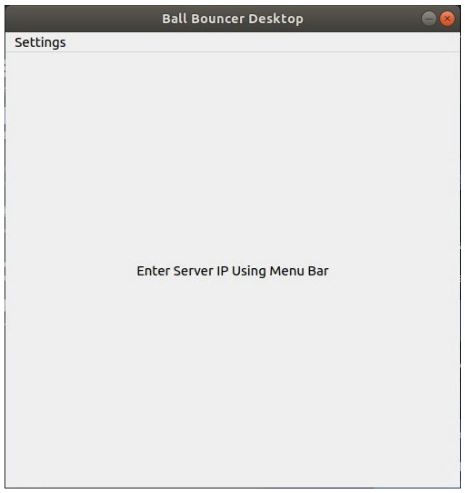
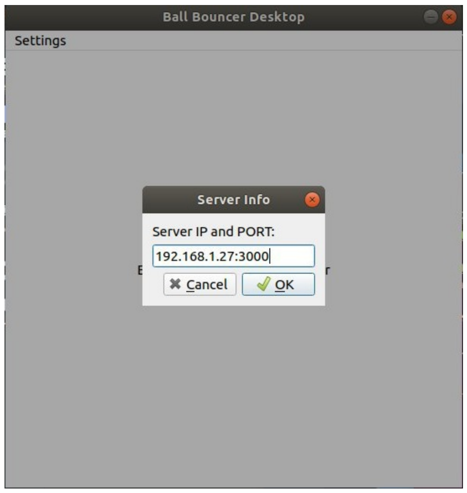


Figure 3: Desktop Before Connect



192.168.1.27:3000

Figure 4: Desktop Connect to URL

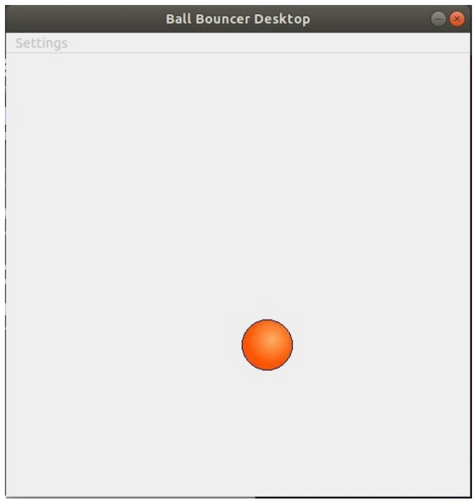


Figure 5: Desktop Output Test

2.3 Mobile Application

The purpose of the Mobile module is to provide a control mechanism to display and control the ping pong ball in real time. An interface design has been developed in this direction. There are two pages in this interface, the main screen and the login screen. On the main screen, the x, y coordinate values and radius values of the ball coming from the Communication module are taken continuously and the movement of the ball is drawn on the screen with this data.

An interface that provides login with IP and port is designed on the login screen. As the system will run over the localhost, an input system has been designed in this way to determine via which IP and port the mobile application will connect to the system

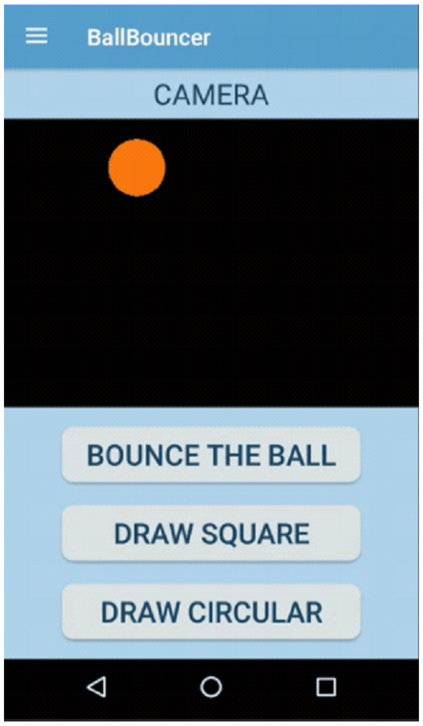


Figure 6: Mobile App Opening Page

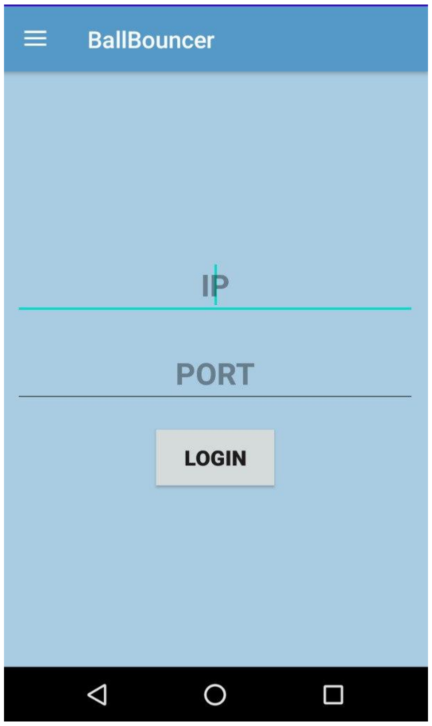
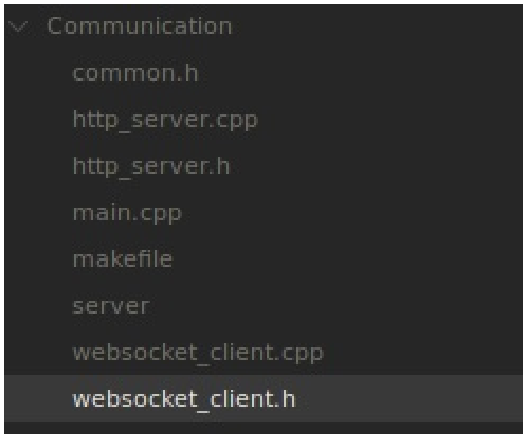


Figure 7: Mobile App Connect Page

2.4 Communication Handling

The purpose of the Communication module is to provide communication between the desktop and mobile modules and the image processing module. This communication is to transmit the data from the image processing and the image to the desktop and mobile modules, and to transmit the data that control the movement of the ball from the mobile and desktop modules. Accordingly, a server structure was created with C++ .



```
Communication
    common.h
    http_server.cpp
    http_server.h
    main.cpp
    makefile
    server
    websocket_client.cpp
    websocket_client.h
```

Figure 8: Communication Handling Files

This module consists of an HTTP server and a WebSocket Client.

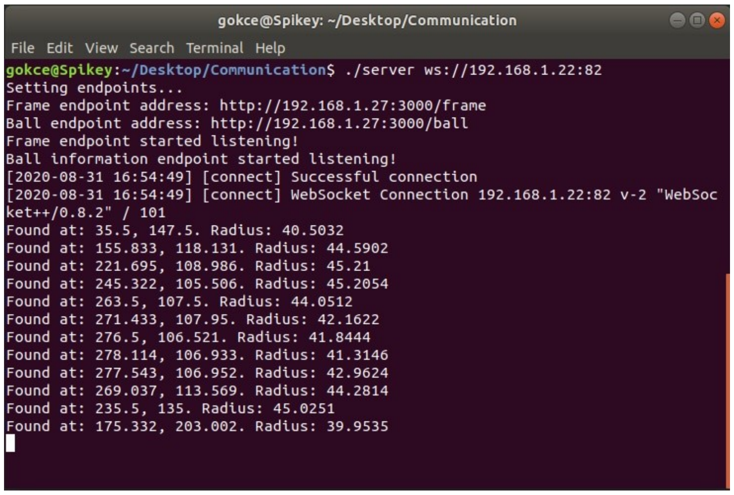
WebSocket Client receives latest image data from ESP32. It finds circle with using image processing module and records the results.

The HTTP server consists of 2 endpoints. Requests can be sent to these endpoints with the HTTP GET method. These endpoints are:

- /frame: Sends raw JPEG data. This data is the last frame taken from the camera.
- /ball : It sends the most recent ball coordinates detected by the image processing module as x and y. It also sends the detected radius of the ball. With using these 3 values by other modules, the 3-dimensional position of the ball can be determined.

Command for running program from terminal:

```
./server [ESP IP:PORT]
```



gokce@Spikey: ~/Desktop/Communication
File Edit View Search Terminal Help

```
gokce@Spikey:~/Desktop/Communication$ ./server ws://192.168.1.22:82
Setting endpoints...
Frame endpoint address: http://192.168.1.27:3000/frame
Ball endpoint address: http://192.168.1.27:3000/ball
Frame endpoint started listening!
Ball information endpoint started listening!
[2020-08-31 16:54:49] [connect] Successful connection
[2020-08-31 16:54:49] [connect] WebSocket Connection 192.168.1.22:82 v-2 "WebSoc
ket++/0.8.2" / 101
Found at: 35.5, 147.5. Radius: 40.5032
Found at: 155.833, 118.131. Radius: 44.5902
Found at: 221.695, 108.986. Radius: 45.21
Found at: 245.322, 105.506. Radius: 45.2054
Found at: 263.5, 107.5. Radius: 44.0512
Found at: 271.433, 107.95. Radius: 42.1622
Found at: 276.5, 106.521. Radius: 41.8444
Found at: 278.114, 106.933. Radius: 41.3146
Found at: 277.543, 106.952. Radius: 42.9624
Found at: 269.037, 113.569. Radius: 44.2814
Found at: 235.5, 135. Radius: 45.0251
Found at: 175.332, 203.002. Radius: 39.9535
```

Figure 9: Communication Output Test

When the program is first started, addresses of HTTP endpoints are printed on the screen and the connection status of the WebSocket client is shown. Later, as the ball enters the area of the camera, HTTP response messages are also printed on the console area.

2.5 Sensor Subsystem Design

This module provides that the camera and step/servo engines work with the ESP32CAM module collaboratively in the manner of software. We started with the ESP32 card and the OV7670 camera module, but later on we noticed that the camera module was already burned.

In order to speed up the project, the ESP32CAM module was purchased (it has an OV2640 camera) and continued to project with this card. The image captured from the camera broadcasted to a WebSocket as M-JPEG, at the same time it can connect or receive a request from an async web server to control the servo movement. PID control algorithm has been selected to balance the ball.

2.6 Image Processing

Highly accurate and dynamic masking was performed with tolerance values at certain intervals in hue, saturation, and value(HSV). Decreases the sizes of objects and removes small anomalies by subtracting objects with a radius smaller than the structuring element. Increases the sizes of objects, filling in holes and broken areas, and connecting areas that are separated by spaces smaller than the size of the structuring element. Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity. A circle with a radius within the acceptable range is determined through the minimum enclosing circle algorithm from these contours.

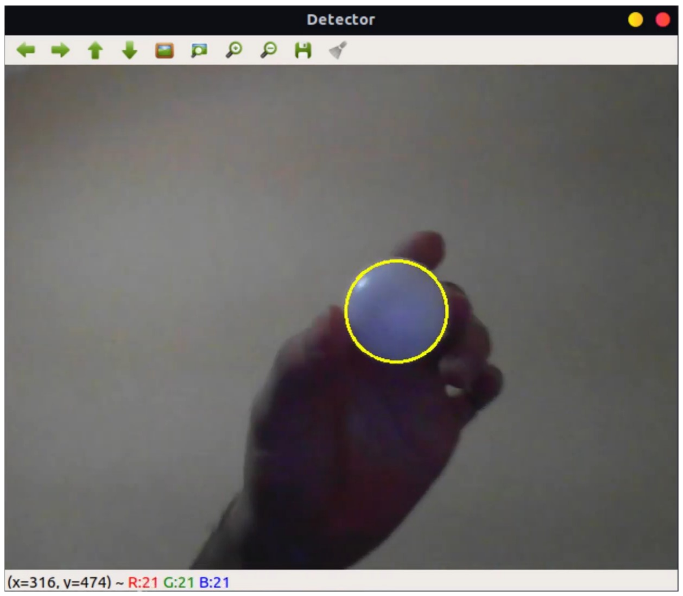


Figure 10: Image Processing Test

2.7 Simulation Design

The simulation design of the project was written and designed in Godot application. It can run in Godot, Fedora, Linux and Windows environments. Therefore, application has been improved on Godot.

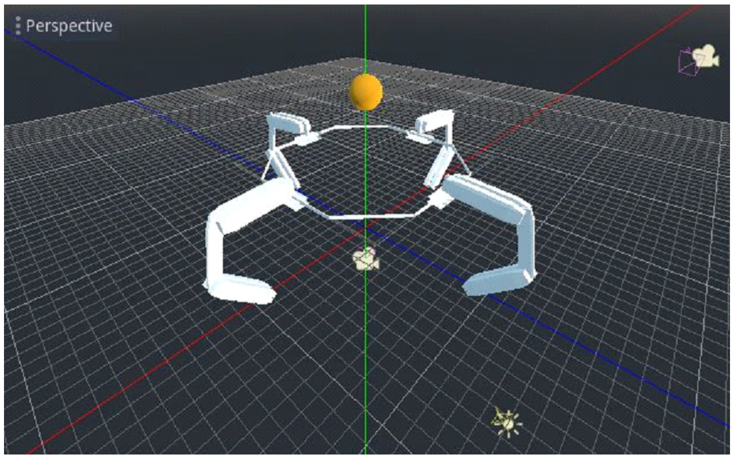


Figure 11: Simulation Demonstration

The board, on which the ping-pong ball is balanced and bounced, is designed as an octagon. In the simulation, ping-pong ball, plate and 4 pieces of leg parts are all available as separate objects and separate classes have been created for each object type. There are two cameras to watch live. There is a separate light source for both cameras.

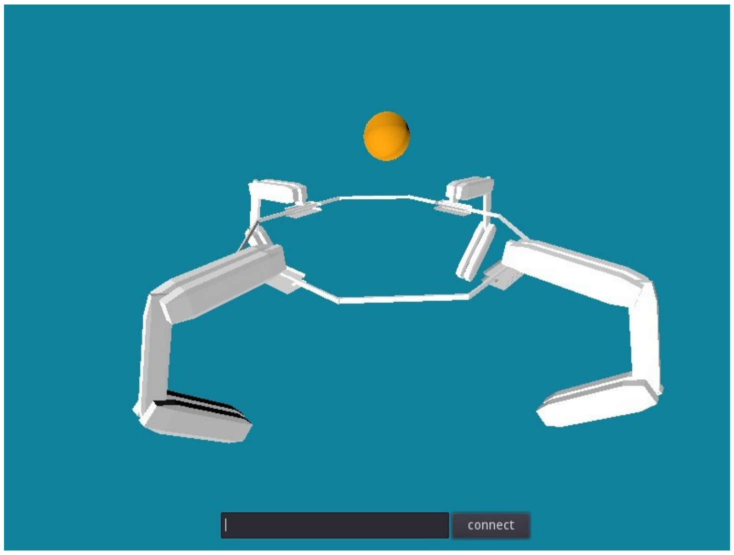


Figure 12: Simulation Connect Server

The data coming from the hardware is taken from the local server and transferred to the simulation. For this, whichever server port was created, this port information should be written on the login page and the connect button should be clicked. Example, `http://localhost:8080/`

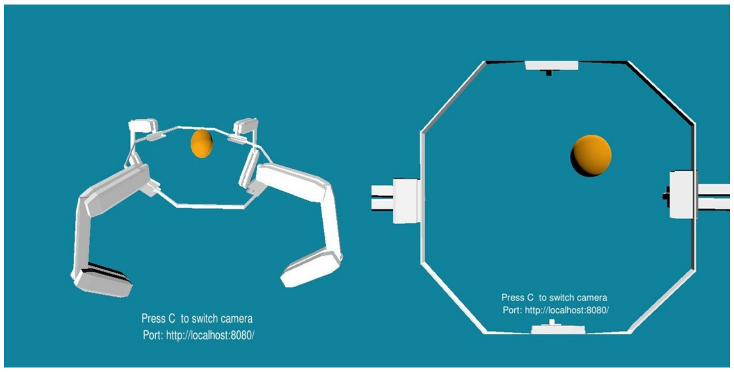


Figure 13: Simulation Front and Back Camera

To see the simulation from different angles, you can switch between camera views with the C key.

In order to receive the data from the server, CanvasLayer was created in Godot and HTTPRequest was added as a node. Connection to the server is provided as follows.

```
HTTPRequest.connect("request_completed", self, "_on_request_completed")
```

When the simulation is started, the port information entered from the interface is turned into a link makes a request to the local server with HTTPRequest.request (port). When the request is sent, the following function has been written to read the data.

```
_on_HTTPRequest_request_completed(result, response_code, headers, body)
```

Data Layout: `<x>,<y>,<z>,<rightArm>,<LeftArm>, <upArm>,<downArm>`
The x, y, z coordinates indicate the position of the ping pong ball. Other data indicate that the legs will remain up, down or motionless. Corresponding values for legs are 1, 0 or -1:

$$1 : \texttt{up direction}$$
$$0 : \texttt{no movement}$$
$$-1 : \texttt{down direction}$$

The data received from the server is in a structure separated by commas. It is first formatted with UTF-8 and parsed with comma parser. At the end of this parsing process, ”key” and ”value” values are created. This data is sent to all simulation objects in runner function as follows:

```
runner(p["x"], p["y"], p["z"],p["rightArm"],
        p["leftArm"],p["upArm"],p["downArm"])
```

The legs, floor and ping-pong ball have overrided _move function. Each object’s own _move function is called with the relevant parameters inside the runner function. Thus, each object is simulated by making its own movement according to the data coming from its own server.

3 Demo

Our demo videos are:
`https://youtu.be/hoIqqhmpntk`
`https://youtu.be/H717wQNtBfE`