Multiplayer Soccer Game Professor Kaliappa Ravindran

(Robo soccer simulation using Webots)

Members: John Badji, Fherdinand Corcuera

I. Introduction.

1.1 Background.

The field of robotics has developed into a technological area that can automate numerous tasks in the manufacturing sector. Additionally, robots are presently utilized to mimic human life in a variety of situations, including but not limited to the cleaning of industrial plants and structures, monitoring of airports and buildings, and hospice care. Most robots being used at the moment are extremely simplistic, with limited computing power, memory, and intelligence. As a result, work has begun on a new generation of autonomous humanoid robots to enhance both the intelligence and dimension of robotic gaming.

The goal of this project is to design a football-soccer game simulator that can be played by humanoid robots. Its goal is to incorporate this simulator's Artificial Intelligence techniques into each player of the game, giving them the ability to play intelligently and compete to win in their game. Within the scope of the project, there will also be an implementation of the development of programs in such a way that these robots should be able to detect their surroundings and compete against an opponent while working together with other robots on their team.

1.2 Motivation.

The study of fully autonomous humanoid robots is one of the most exciting areas in robotics research. All the electromechanical issues associated with the performance of the mechanisms that allow movement and its autonomy must be solved, as well as the creation of a system that allows the robot to walk upright and make actions similar to those of a person. Like that issue, this one calls for modeling, simulation, and intelligent control of a complicated robotic mechanism with more than 20 degrees of freedom.

1.3 Problem Statement

Overall, this project aims to accomplish the following through the programming of a robotic game simulator:

1. To enhance mobility, expedite travel, and locate the quickest routes between two points.
2. Increase perceptual and sensory acuity, or the capacity to accurately identify one's surroundings in order to respond appropriately in the present.
3. Being able to model the various states the player goes through during the game more efficiently.
4. Improve each robot's capacity to transition from one state to another in a way that meets the efficiency and speed objectives.
5. Create smart systems that will let robots devise their own tactics when playing soccer.

The major goal for this semester is to create object-oriented models for controlling and executing the robot's actions. In this report, the following tasks were implemented in simulation software:

1. Get up from the ground
2. Locate its own position on the court
3. Locate the ball’s position
4. Walk toward the ball
5. Position itself close to the ball
6. Locate the goalpost
7. Kick the ball toward the post

II. Basic Concepts of Robots

Typically, a standard machine is constructed with a degree of flexibility that permits all preprogrammed motions to be carried out by a single actuator system (be it a motor). Nevertheless, a robot is constructed with multiple degrees of freedom in order to achieve flexible automation and facilitate the programming of the desired movement in response to the industry's ever-changing demands. A degree of freedom is the ability of a manipulator to perform horizontal, vertical, or rotational movements.

2.1. Robots that are being used in the project

- **Humanoid and Bipedal Robots**

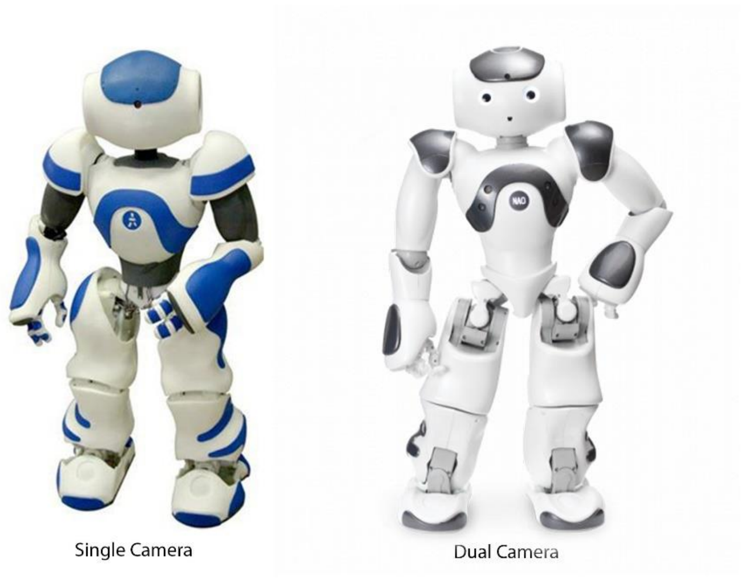
  A bipedal robot is a walking robot that moves by using only two extremities known as legs. With this description, the human being is definitely the model of bipedal movement. A human is not, however, the only living being that uses bipedal walking for locomotion. Kangaroos and other bipedal marsupials jump using their legs, which are stabilized by their tails. Birds, like humans, are bipedal animals that utilize their legs to jump and walk. In the case of penguins, however, they adopt another mode of mobility, such as flying or swimming.

Atlas by Boston Dynamics (An example of a bipedal robot)

- **Nao**

  Nao is a humanoid robot that is a self-aware, programmable and moderately sized robot. The French company Aldebaran Robotics designed and built it [1]. The Nao robot has been utilized in the RobotStadium competition since 2007 when it was selected to take the position of the AIBO as the official robot of the RoboCup [2]. This particular robot initially just had a single camera, but it has since been upgraded to include two, which has significantly improved its capacity to locate the ball.

  The fact that Nao can be fully programmed using tools like Choregraphe, a graphical application program from Aldebaran Robotics, or the Webots framework, which permits working with simulation and downloading the code straight to the robot helps a lot in creating this project [3].



Different Types of Nao: Nao V1(2006), and Nao V6(2018)

III. Robotic Simulation.

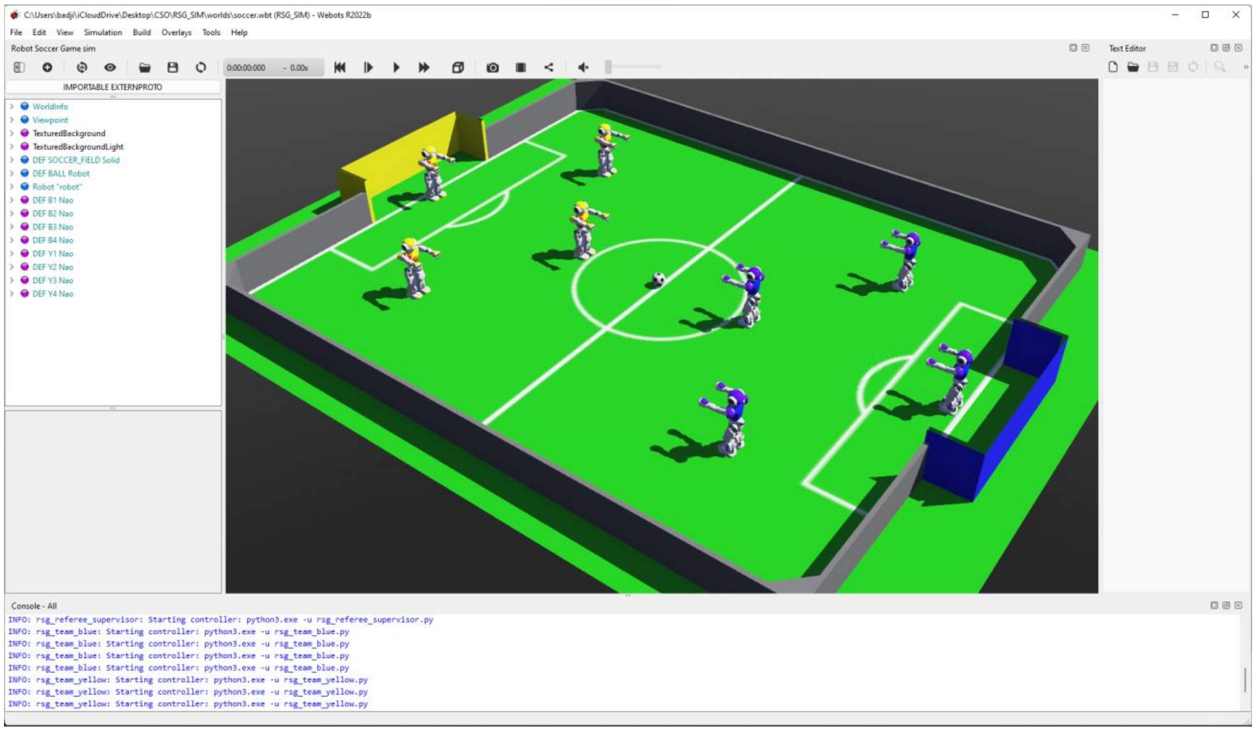
3.1 Webots Simulator

Webots program lets the user build 3D virtual worlds, complete with physical attributes like mass, distribution, friction, inertia, etc., to simulate robots and design their environments. The simulated robots can feature a variety of sensor and actuator devices, including infrared (IR) distance sensors, wheeled motors, cameras, servos, touch sensors, gyroscopes, emitters, and receivers, and can move in a variety of ways (wheeled, legged, swimming, or flying). With Webots, you may model and customize a wide variety of commercially available robots. Atlas may be trained using a variety of languages, including C, C++, Java, Python, and MATLAB.

Webots' simulation engine is capable of both kinematics simulation and dynamics simulation. However, you may customize the simulation's underlying physical parameters to your liking (such as friction, wind, etc.). Since the physical characteristics of the simulated elements (such as mass, inertia, or friction) are ignored in kinematic simulations, they also operate more quickly. Webots makes use of the ODE (Open Dynamics Engine) package to simulate physics-based interactions. This package has a lot of functionality for simulating physical systems. Webots allows for very nuanced regulation of robots' kinetic interactions with their surroundings. In addition, Webots lets you represent forces like wind and water currents, which aren't present in a "normal" simulation, by incorporating other libraries into the ODE forces.

3.2 Controllers

In the multiplayer-soccer game, there are eight robots in total: four on each team, with three attackers and one goalkeeper for each side. It is important to understand, from a programming standpoint, that during a match, two distinct types of controllers are being run simultaneously, each with its own unique set of capabilities. The two types of controllers in the multiplayer-soccer game are Soccer Player and Supervisor. The Soccer Player controller will control all the players, and the Supervisor controller will be the match referee. The strategies and tactics that we are going to use are implemented in the controller programs. We wrote our controllers using Python 3 as the programming language.



```
INFO: rsg_referee_supervisor: Starting controller: python3.exe -u rsg_referee_supervisor.py
INFO: rsg_team_blue: Starting controller: python3.exe -u rsg_team_blue.py
INFO: rsg_team_blue: Starting controller: python3.exe -u rsg_team_blue.py
INFO: rsg_team_blue: Starting controller: python3.exe -u rsg_team_blue.py
INFO: rsg_team_blue: Starting controller: python3.exe -u rsg_team_blue.py
INFO: rsg_team_yellow: Starting controller: python3.exe -u rsg_team_yellow.py
INFO: rsg_team_yellow: Starting controller: python3.exe -u rsg_team_yellow.py
INFO: rsg_team_yellow: Starting controller: python3.exe -u rsg_team_yellow.py
```

Figure 1: simulation world before kick-off

3.2.1 Supervisor

Unlike Soccer Player Controller, everyone uses the same version of this controller. In each game, only one supervisor will oversee the game. From the kickoff of the match, the referee is responsible for setting up the robots in the appropriate positions, keeping score, remaining time, and other match-related statistics. When the ball leaves the playing field, it is in charge of replacing the ball. As the game progresses, the Supervisor updates all robots on the score, the condition of the game, the remaining time to play, and any other relevant match data.

3.2.2 Supervisor variables that were done for this semester.

- **Python Code Examples:**

  Goal Check: a function that checks if a team scored a goal.

  ```
  def ball_in_TeamBlue_goalpost (x, y):
  IF T1_lowLim_goalpost <x <T1_upLim_goalpost
      IF T1_bwall_limit < y < T1_fwall_limit
          return TRUE
  return FALSE
  ```

  Ball Outside; a function that checks if the ball went outside the playing field.

  ```
  def ball_out (x, y):
      IF x > xArea_upperLim or x < xArea_lowerLim
          return TRUE
      IF y > yArea_upperLim or y < yArea_lowerLim
          return not (Goal Check TeamBlue or Goal Check TeamRed)
  return FALSE
  ```

- **Declared constant variables:**
  - Upper and Lower Limit of the field
    - In order for the referee to maintain track of whether or not the ball has left the field and relay that information to the players, these two variables have been declared as constants.
  - Upper, Lower, Back wall, Front wall(imaginary) Limit of the goalpost

    - These variables were also declared constants so that anytime a player shoots the ball into the goal, these variables will be able to assist in determining whether the ball was kicked to the goal or if it rolled out of the playing area.

3.2.3 Soccer Players

The program that controls the robots is referred to as the Soccer Players. This is the phase at which each player should be modified in order to compete with their opponents. Each participant in the game has their own controller. Soccer player controllers might be the same for all players, yet each player has different tasks to do.

Every participant is free to choose the code they want to program for the implementation, and no restrictions will apply. This means that the Webots installation includes a standard program, and each participant updates it to create more competitive robots than the others.

3.2.4 Soccer Players variables that were done for this semester.

We wanted to develop a simple collaborative strategy, so communication among players is crucial. Every team member must be on the same page. For example, For the collaborative approach to work, other player needs to know when one of their teammates has the ball. So, they can stop chasing after the ball.

- **How do we make robots communicate?**

  Webots allow us to create communication channels through which every robot that has an emitter connected to that channel can send messages and everybody with receivers connected to the channel can receive the messages.

  Each team is assigned a different channel through which they communicate. While Webots support changing channels. This is turned off so one team will not be able to spam the opponent team with a bunch of messages.

Every time a message is sent through an **emitter**, all **receivers** listening to the channel are going to receive it.

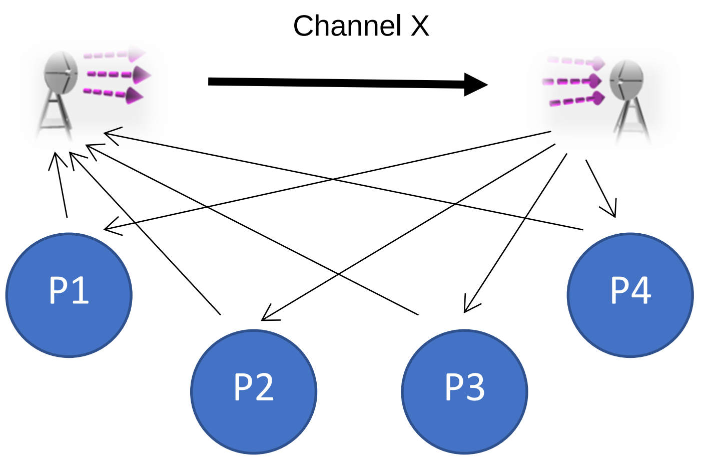


Figure 2: Drawing depicting intra-team communication trough a certain channel X

The receiver uses a ***Queue*** data structure. Therefore, when a new message is received it goes to the end of the queue. The oldest message gets processed first. It is also important to empty out the queue regularly, so we don't end up with outdated messages.

- **Simple strategy developed**

  For experimenting with the simulation world, we came up with a simple play strategy.

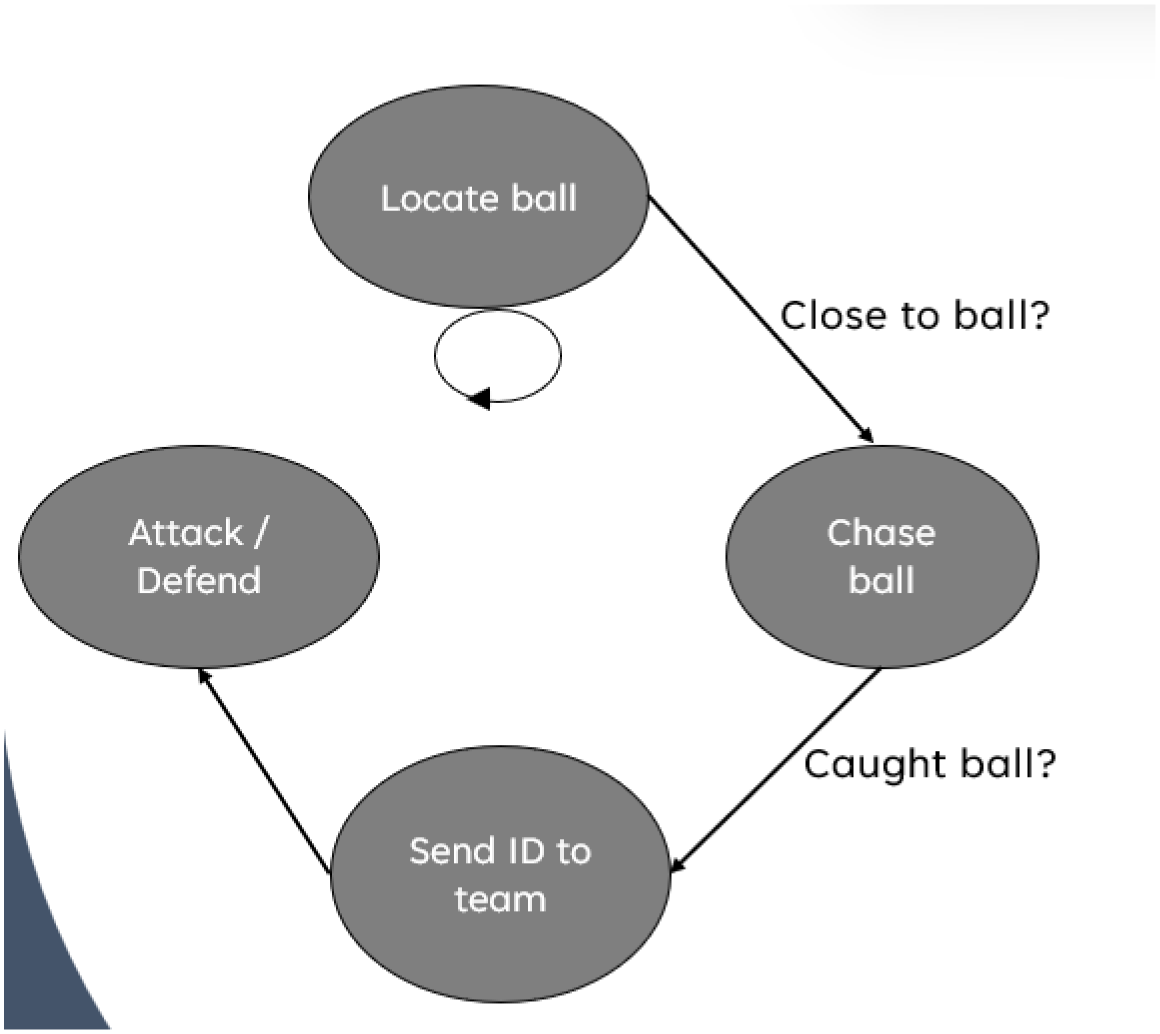


Figure 3: Play strategy as a finite state machine

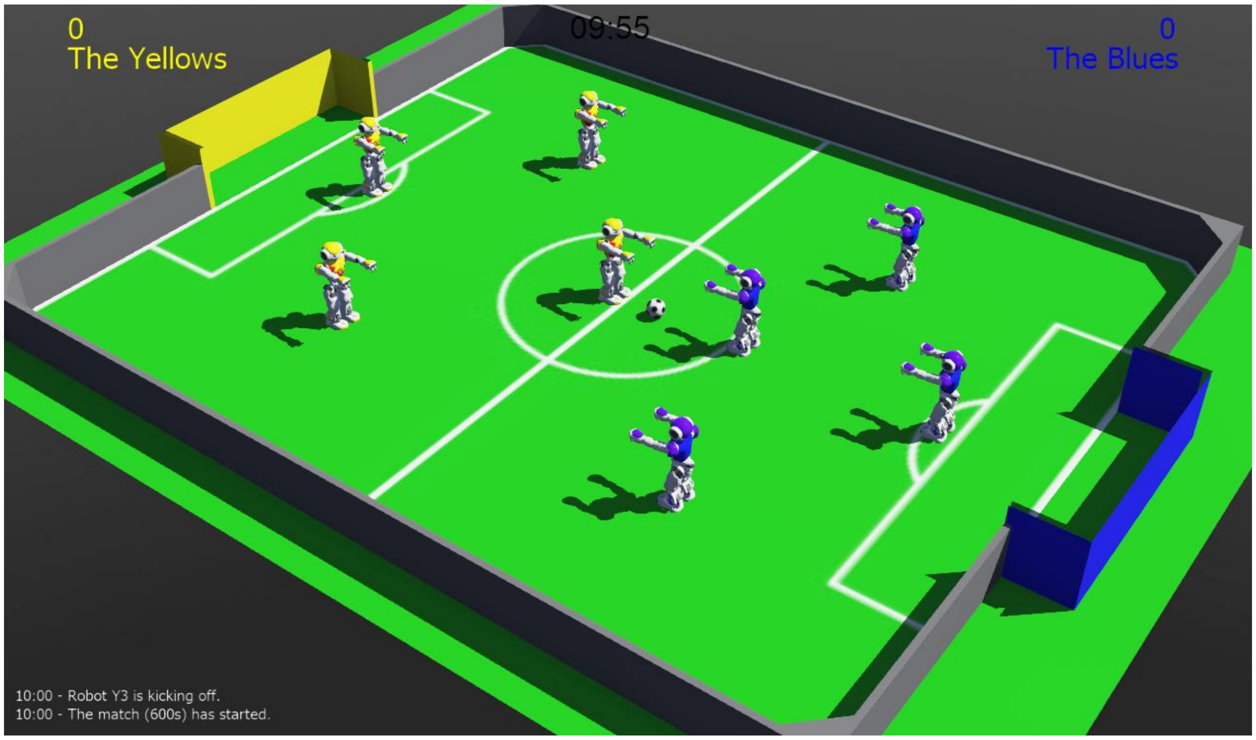
The ball is equipped with an emitter and constantly sends messages through a channel that is different from the team communication channels. All players have receivers that listen on that channel. Each player receives the direction of the ball and the strength of the signal as the message.

The direction part of the message allows players to know where the ball is headed, and the strength of the signal allows players to know how close they currently are to the ball.

IV. Expected Result and Future Tasks

4.1 Expected Result

The major goal of this report was met, which was to achieve that a single Nao robot operating in a virtual world could undertake a series of independent actions to locate a ball, direct it towards a goal, and shoot towards the goal identified. These activities serve as the foundation for carrying out the project's second phase, which consists of programming a group of robots to play soccer against an opponent.



10:00 - Robot Y3 is kicking off.
10:00 - The match (600s) has started.

Figure 4: team yellow player kicking off the match

4.2 Future Tasks

The second semester would consist of the following tasks to be done for our project:

- Fix some issues concerning the player's controller that produces undesirable results.
- Improve coordination and communication between robots to allow them to play better strategies.

    - Once robots can better exchange information, a variety of activities can be coordinated between them. This will allow us to expand the player's controller to include more complex actions. For example, passing the ball to a teammate that is open.
- A separate controller for goalkeepers
- Implement separate offensive and defensive game plans
- Auto-localization
    - This action is essential for the robot soccer game since it would help to figure out where on the pitch the players are positioned, which is necessary for planning effective moves. It could, for instance, employ a game plan on what to do in case the striker never drops below half-court, and the defender never advances over half-court.
- Host players on different machines. Set up TCP/IP channels for robots' communications.

V. Conclusion

The creation of a behavior simulator that allows a Nao robot to play a soccer game was carried out in this project. Incorporating artificial intelligence so that the robot could play independently was also implemented. While doing this, it enables the robot to make decisions about when to conduct specific activities. The objectives outlined at the beginning of the thesis were met and the creation of algorithms for vision, movement, and behavior control was completed successfully.

VI. References

1. Aldebaran Robotics, http://www.aldebaran-robotics.com
2. https://www.robocup.org/leagues/5
3. https://www.intelligentroboticslab.nl/robots/the-nao-robot/