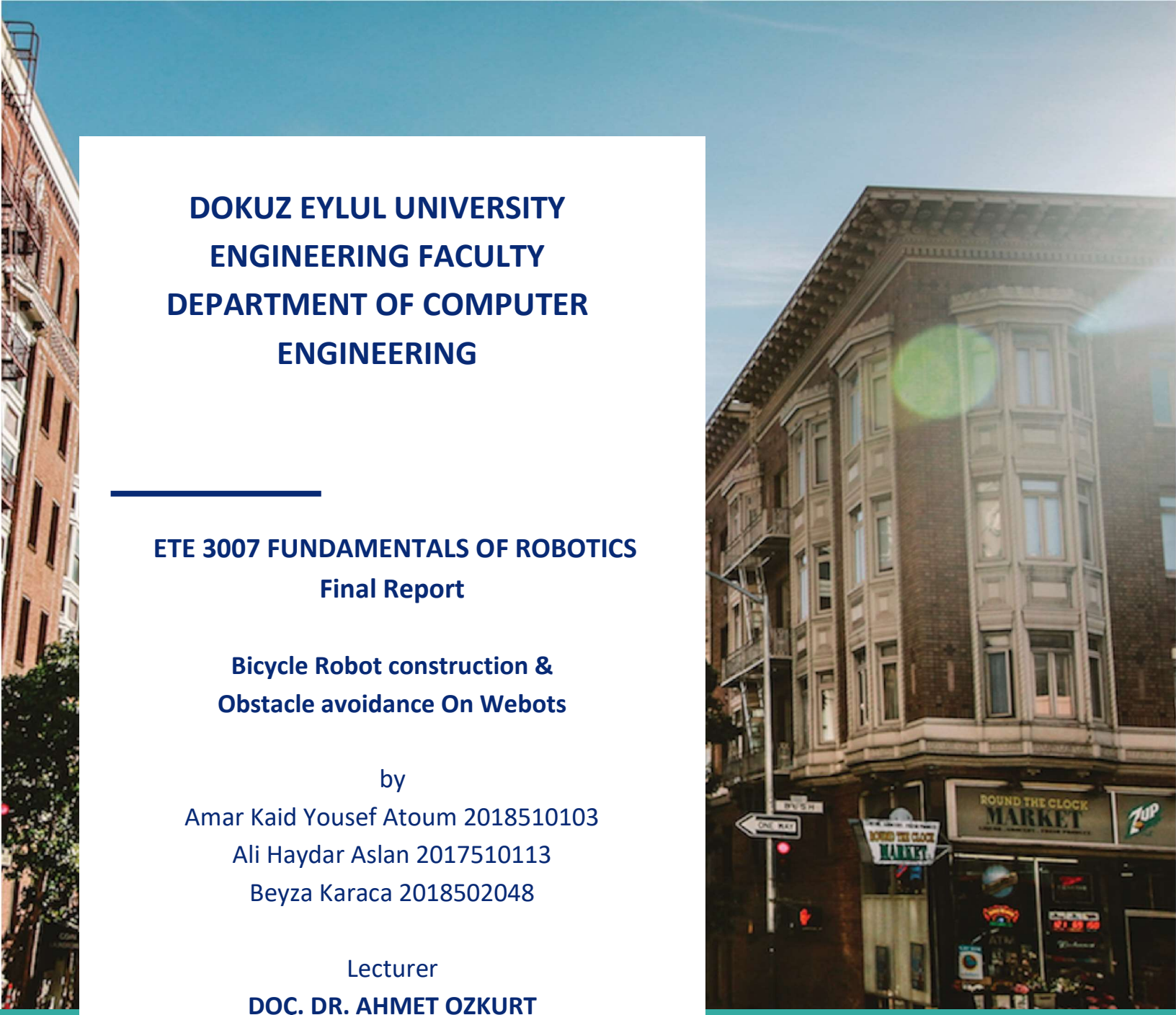


DOKUZ EYLUL UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

ETE 3007 FUNDAMENTALS OF ROBOTICS
Final Report

Bicycle Robot construction &
Obstacle avoidance On Webots

by
Amar Kaid Yousef Atoum 2018510103
Ali Haydar Aslan 2017510113
Beyza Karaca 2018502048

Lecturer
DOÇ. DR. AHMET OZKURT

26 MAY 2022

1. Introduction

Webots is an application used to simulate robots and models with different movements and designs. The application was used to construct an auto mobile bicycle that avoids obstacles. Keep in mind that physics of the robots and its geometry had to be as close as it is in real life. The controller of the robot was written in Python 3.7.

Aim of the Project

This project seeks to create a world as well as a bicycle robot in webots, that can autopilot its way around the world it's made in. The robot should also be able to avoid obstacles like cars, walls, humans whatever is solid and not moving.

Used technologies and references:

- **Webots.**
- **TinkerCad.**
- **Python 3.7.**
- **Discord Webots Community server.**

2. Progress Summary

2.1 Work done

The team was able to achieve a lot through the time given, due to its teamwork and motivated teammates. However, there were some hardships along the way, that will be discussed later in the upcoming chapters. Tasks were divided among the three colleagues as follows.

Amar Atoum:

- Modelling the bicycle.
- Physics and Hinge point allocation.
- Motors allocation and coordination with the wheel's coordinates.
- Environment and world creation.
- Creating the controller for the bicycle.
- Taking inputs from the user to control the bike. (To be removed later)
- Adding a camera to the robot that will later be used for the obstacle avoidance.
- Adding sensors to the robot.
- Modelling the obstacles.
- Obstacle avoidance code addition to the controller

Ali Haydar Aslan:

- Modelling the bicycle.
- Environment and world creation.
- Creating the controller for the bicycle.

Beyza Karaca:

- Modelling the bicycle.
- Environment and world creation.

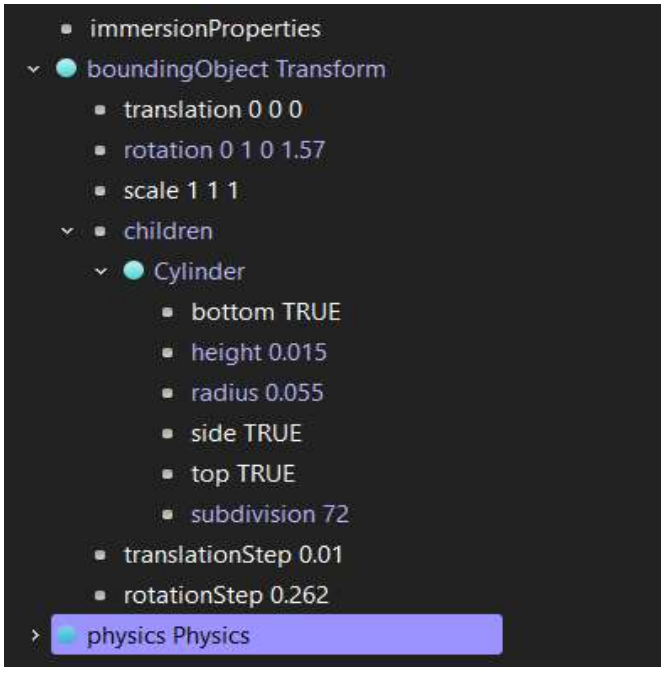
3. Algorithms & Solutions

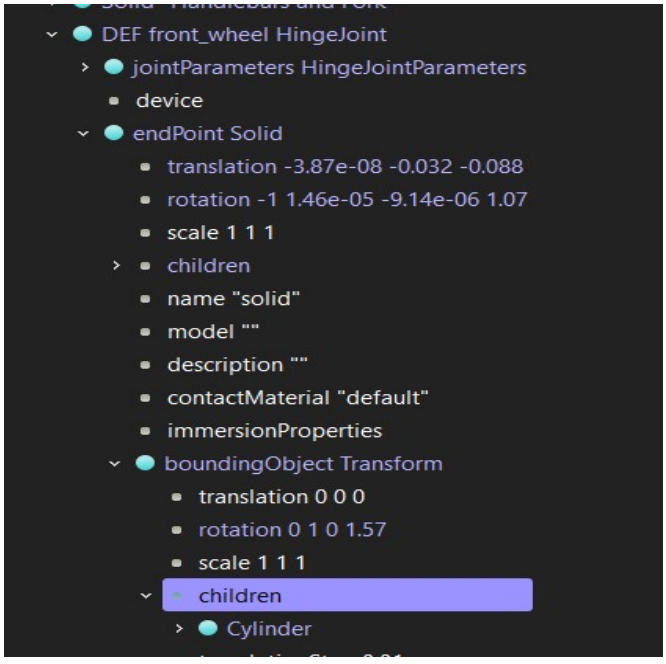
3.1. Bicycle Balancing:

The problem of balancing was something that the group was struggling with as finding a reference for it was nearly impossible but by questioning and inquiring from the Webots discord community, a solution emerged from the helpful admins. The bicycle was able to self-balance due to its design and the ability for Webots to add physics to certain objects, note that when designing the bicycle, we were being as close as possible to the real-life model. However, we didn't add any weight to certain objects the only weight added objects are the following and their symmetry were changed to achieve balance:

- The two wheels.
- The two sensors.
- The handlebar and fork.
- The whole robot is an object.

Other than that, the wheels were designed in a cylindrical form and they are a bit wider than usual, meaning they have enough surface area to not fall on either side.



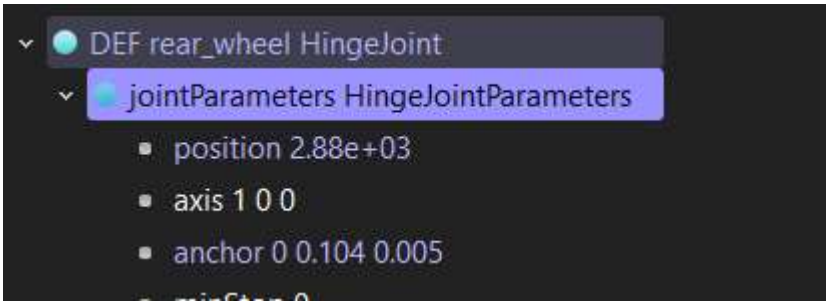


In conclusion, Due to the successful modelling of the bike and the correct weight distribution. The Bicycle was constructed successfully and was able to achieve self-balance. To clear any confusion, the bicycle will fall if its motors were activated in high velocities and had a slight collision with any object (same as real-life).

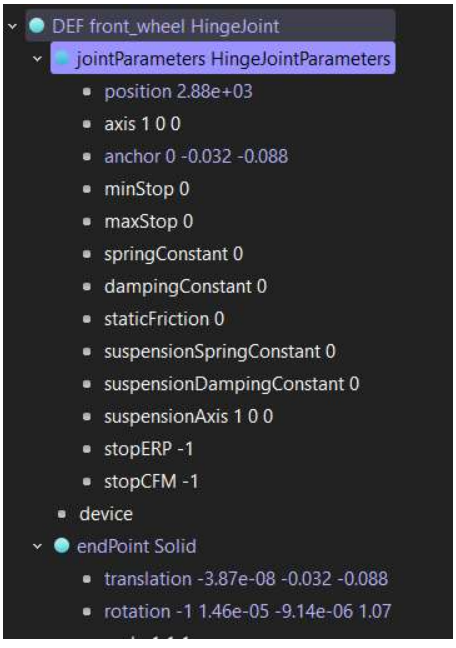
3.2. Hinge points and motors:

Allocating hinge points and setting up motor position was extremely hard to do since any mistake with the anchor position will not allow our robot to move or move correctly. However, we came up with the solution to this problem. This will be shown in the following screenshots…

This hinge point refers to the rear wheel and will be connect to its motor:

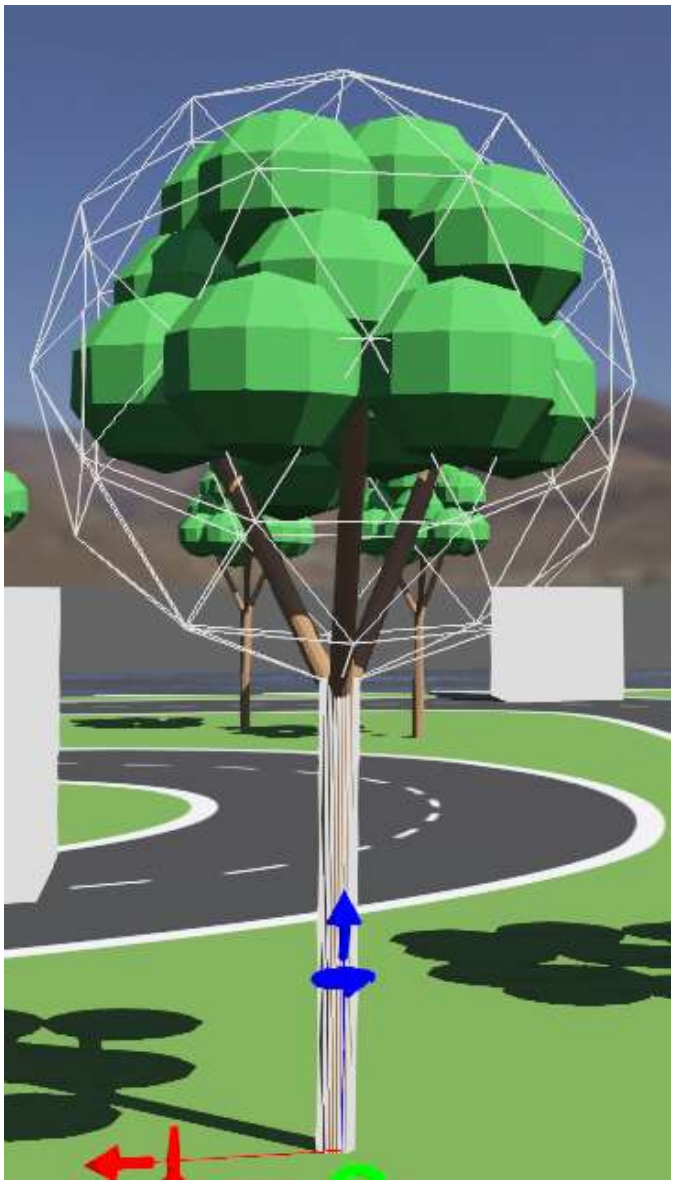


This hinge point refers to the front wheel and will be connect to the motor of the handle bar motor:



3.3. Environment creation and models importing:

The environment floor was made using a track image file and is located in the texture file, with the name “track001.jpg”. The image file allowed us to create a racing track like environment for the bicycle to move on. The team members also decided to decorate the track with trees which were modelled individually and then transformed and imported using the file menu and then selecting the import 3d model option to import 3d object files with the extension “stl”. The same procedure was implemented to the components of the bicycle and each part was transformed to an object or shape of certain geometry.



3.4. Motor movement and Code:

The bicycle operates in the follow notion, the back wheel has a rotational motor that keeps the back wheel rotating for infinity however, there is a certain limit for the rotational velocity of the motor. The motor cannot and shouldn't exceed. This part will be explained further after we explain our code. There exists two important libraries in webots and these are Robot and DistanceSensor and they are crucial for us to start up the robot and make it move. Thus, we have to import them and we initiate a robot object. Other than that, we initialize the time step which is the time of the virtual world we create. Usually, you can set that as 64 but here we try to be more realistic. Anyways, we get our Distance sensor objects that indicate for our sensor created and placed on the robot and we enable them according to the time.

Then we get the motors for the back wheel and the handle bars and we set their positions and velocity to zero.

Before we made the obstacle avoidance, we had the ability to control the bike with the keyboard and we wanted to stay there just in case we wanted to experiment with something. We also have a camera that would have taken part in the obstacle avoidance algorithm.

The handlebar's max position is declared by the variable, "hMax" where the positive value goes to the right and the negative value goes to the left.

Then we create a loop for the robot and we set the velcoity as 6 which leads to less falls and handbar position to zero which will result to no turns. We check for Obstacles by getting the 2 sensors values. After that it will check wether they are far enough (less than 700)which is a value for less than 5 m. Take in mind that , if its more than 5 m, then no obstacle will be detected. These values are stored in the look up table of both sensors. When there is an bostacle deteceted the counter will become a hundred and the counter will decrement with time and turn to the left and go backwards as to give distance to avoid the obstacle.

```
"""little_bicycle_P controller."""

from controller import Robot #Supervisor
from controller import DistanceSensor
# create the Robot instance.
robot = Robot()

# get the time step of the current world.
timestep = int(robot.getBasicTimeStep())
ds = robot.getDevice('ds_left')
ds.enable(timestep)

ds1 = robot.getDevice('ds_left1')
ds1.enable(timestep)



# get wheel motor
whemotor = robot.getDevice('wheel motor')
whemotor.setPosition(float('inf'))
whemotor.setVelocity(0)

# get handlebars motor
hndmotor = robot.getDevice('handlebars motor')
hndmotor.setPosition(0)

# keyboard enable
robot.keyboard.enable(timestep)
robot.keyboard = robot.getKeyboard()

# Initialize camera
camera = robot.getDevice('camera')
camera.enable(timestep*4)

# Max bicycle velocity
maxS = 11

# Handlebar angle
hMax = 0.1920 # rads (11°) Max
```

```
# Ride mode (1 autoatic; 0 manual)
autoR = 0
avoidObstacleCounter = 0
# Main loop:
key = 0
while robot.step(timestep) != -1:

    # Set velocity rear wheel
    whemotor.setVelocity(6)
    hndmotor.setPosition(0)
    #turn right
    if avoidObstacleCounter > 0:
            avoidObstacleCounter -= 1
            hndmotor.setPosition(hMax)
            autoR = 0
            whemotor.setVelocity(-3)
    else:  # read sensors
        if ds.getValue() <700 or ds1.getValue()<700:
           avoidObstacleCounter = 100


    # if (key == 314): # left key
        # hndB = hMax
        # autoR = 0
    # elif (key == 316): # right key
        # hndB = -hMax
        # autoR == 0
    # elif (autoR == 0):
        # hndB = 0 # center
    # if (key == 315):
        # hndB = -hMax
        # autoR = 1 # automatic on

    pass
```

3.5. Sensors:

As you can see the through the picture the sensors are the little black squares. Note that the sensors have to be rotated in a way where their y axis is the on facing and collecting information for the obstacles. Otherwise, they won't work.

4. Screenshots

little_bicycle_P.py

```
19
20 # keyboard enable
21 robot.keyboard.enable(timestep)
22 robot.keyboard = robot.getKeyboard()
23
24 # Initialize camera
25 camera = robot.getDevice('camera')
26 camera.enable(timestep*4)
27
28 # Max bicycle velocity
29 maxS = 11
30
31 # Handlebar angle
32 hMax = 0.1920 # rads (11°) Max
33
34 hndB = 0 #-hMax # default: right
35
36 # Ride mode (1 automatic; 0 manual)
37 autoR = 0
38
39 # Main loop:
40 while robot.step(timestep) != -1:
41
42     # Set velocity rear wheel
43     whemotor.setVelocity(maxS)
44
45     # Set position handlebar
46     hndmotor.setPosition(hndB)
47
48     # Get pressed key
49     key=robot.keyboard.getKey()
50
51     if (key == 314): # left key
52         hndB = hMax
53         autoR = 0
54     elif (key == 316): # right key
55         hndB = -hMax
56         autoR == 0
57     elif (autoR == 0):
58         hndB = 0 # center
59     if (key == 315):
60         hndB = -hMax
61         autoR = 1 # automatic on
```

5. PROBLEMS ENCOUNTERED

- Learning the software (Webots) was difficult at first as the learning curve for the application is hard and lacks tutorials and guidance videos on YouTube.
- Modelling the bicycle was difficult as the team wasn't aware of the fact that you can import Models from tinker cad.
- Difficulties while importing the model were persistent as it took a long time to collect each part and assign them to their right place in webots. In addition, when importing the model, the whole bicycle came in as one object where each part of it can't be manipulated. Each part had to be modelled and exported individually.
- Getting the model to move was difficult as the hinge points, the physics and the geometry of each part had to be perfect and in coordination with the wheels.

6. Conclusion

The project was successfully created and simulated by the group and the projects seems to be doable. The bike can be controlled with the left and right arrows of the keyboard, however it lacks obstacle avoidance and sensors.