Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Scientific Experimentation and Evaluation

Project Report

Team Members

Syed Musharraf Ali (sali2s)
Muhammad Talha (mtalha2s)
Hamza Ullah Khan (hkhan12s)

Supervised by

Prof. Dr. Paul G. Plöger
Santosh Thoduka

April 2021

Contents

List of Figures

1

Motion Analysis of LEGO Differential Drive Robot

1.1 EV3 Lego Robot Design

1.1.1 Robot Design

- Robot is designed using the LEGO Mindstorms kit as per the instructions given in the manual.

Figure 1.1: Basic Robot Design

- In differential drive configuration, the robot consist of 2 wheels which are motorized and move independently from each other. A third wheel which is called the spherical wheel is also attached to the robot.

↳ check terminology?

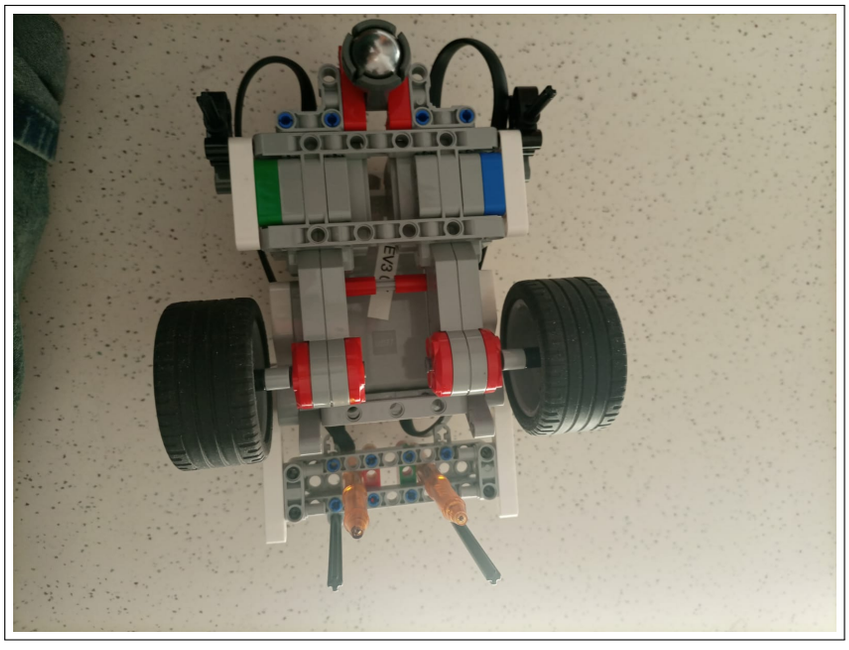

Figure 1.2: Bottom view of the robot

- Due to the 3 wheels configuration, the robot is stable and the centre of gravity is within the triangle formed by the wheel configuration.
- A controller is attached at the top of the robot, this sends movement commands to the motor which allow the robot to move forward, right and left.

Figure 1.3: Controller attached to the top of the robot

- For the measurement purpose, a mechanism is designed which hold the pens in a fixed position and it is situated at the front of the robot as can be seen in Figure 1.4.

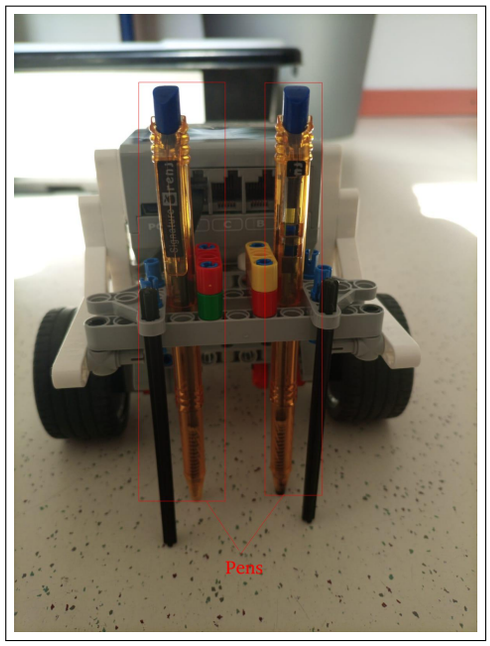

Figure 1.4: Front view of the robot where the pen is situated

- To ensure that the starting position of the robot for each trial run remains the same, 4 markers are added with 2 at the front (Figure 1.5) and 2 at the back (Figure 1.6).

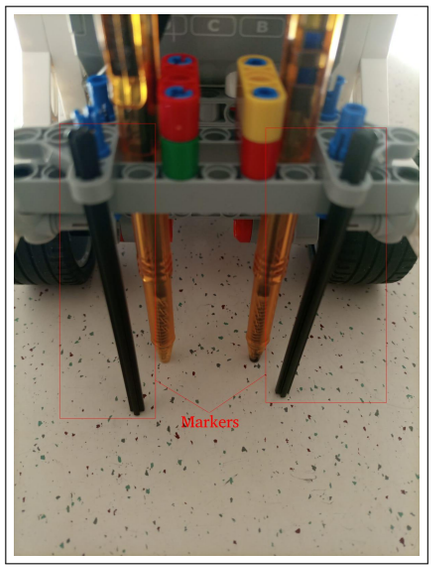

Figure 1.5: Front view of the robot with placed markers

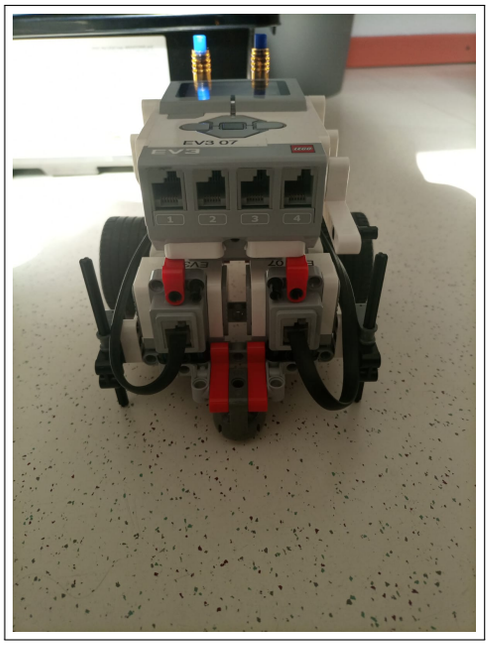

Figure 1.6: Rear view of the robot with placed markers

- When the robot stops, the final position of the robot is marked using pens which are attached at the front.

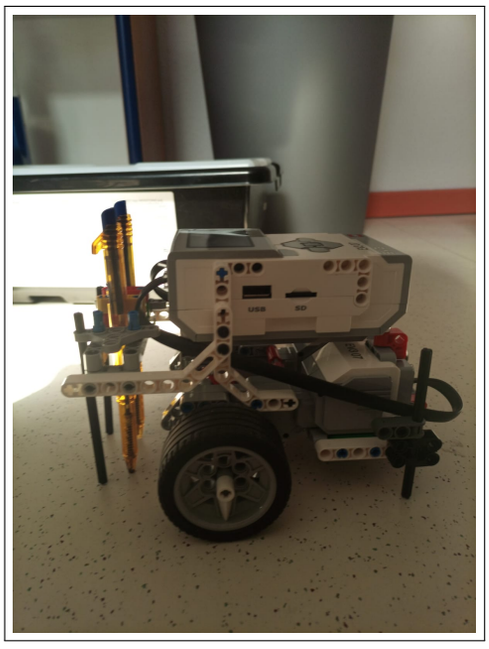

Figure 1.7: Side view of the robot

Pretty good job on defining all terms related to the measurement process :)

1.1.2 Measurement Facility

The following components are used for the purpose of the measurement and hence referred to as the measuring facility:

- LEGO Robot
- Two Pens
- Four Markers
- Measuring Tape with 0.1cm accuracy
- A1 sheet with grid length of 2.5cm

1.2 Measurement Methodology

1.2.1 Approach

- Initially the robot which is our device under test (DUT) is placed on the A1 sheet.
- The position of marking pens will be considered the initial position of robot.
- The markers are also marked on the grid sheet to ensure that the initial position of the robot remains same.
- The initial and final positions of the robot will be marked by descending the two pens after which these will be lifted again.
- The robot will then be controlled to move forward, to the left or to the right and then the final position of the robot is marked on the sheet using two pens. This process will be repeated at least 60 times with 20 times for each different movement.
- Distance between the starting and the final position is the measurand. In our measuring method we will used measuring tape to calculate the distance.
- The orientation of the robot with respect to its initial position is also a measurand and it will be determined with the help of the starting and the ending position vectors as can be seen in the following equation:

$$\theta = cos^{-1}(\frac{\mathbf{a}.\mathbf{b}}{||\mathbf{a}||x||\mathbf{b}||}) \quad (1.1)$$

where $\mathbf{a}$ is the starting position vector of the robot and $\mathbf{b}$ is the final position vector of the robot.

- The pose of the robot is denoted by $[x, y, \theta]^T$ where $[x, y]^T$ is measured in cm where as the orientation θ will be measured in degrees.

it is a measurement result

Refer slide no. 21 of "measurement as a process"

translation

Maybe use a diagram to represent global origin & the dimensions of the robot including its centre.

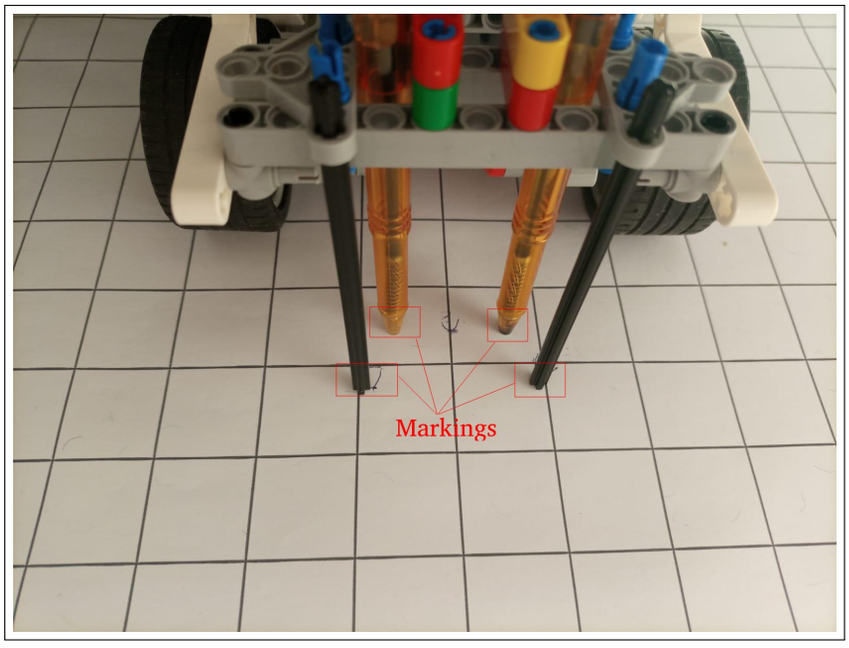

Figure 1.8: Markings on the front of the robot

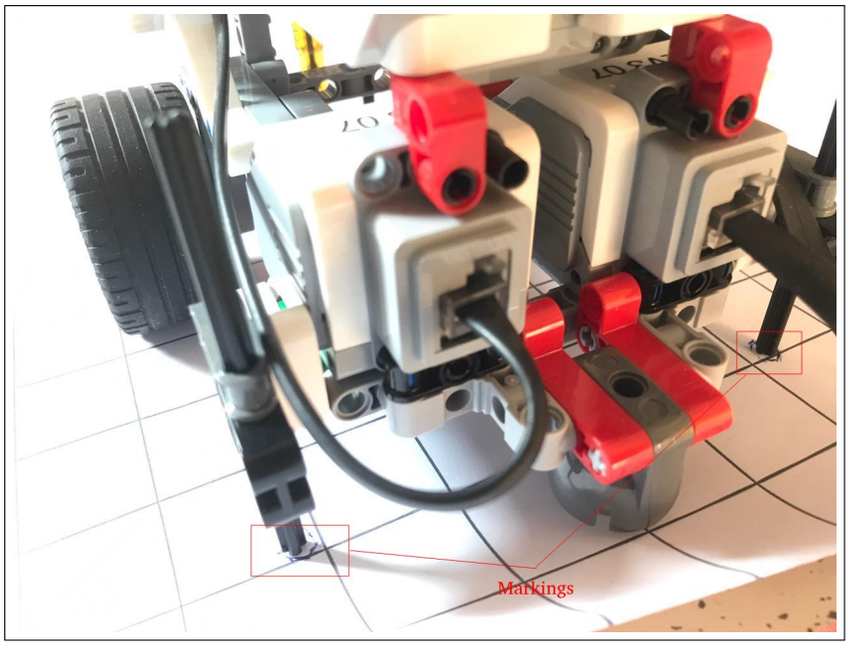

Figure 1.9: Markings on the rear of the robot

1.2.2 Expected Problems

- Parallax error can occur causing an uncertainty of ±0.2 cm.

↳ when taking what measurement?

- Measurement error since the measuring tape can measure with an accuracy of 0.1 cm hence there is an uncertainty of ± 0.1 cm for each individual measurement.
- Wheel slippage can cause an error with an uncertainty of ± 0.5 cm.
- Ascending and descending of the robot may cause the markers to be dropped at different positions each time resulting in the variations of the marker positions hence causing an error of about ±0.3 cm.

1.2.3 Expected Performance

- Errors will be handled with care to ensure that they are avoided or at least minimised. This will be accomplished by doing the experiments with utmost care.
- Readings will be taken by looking directly perpendicular to the measuring scale in order to avoid parallax errors.
- Wires will be checked before each experiment is performed to ensure that there are no loose connections.
- It will be made sure that markers are tightly held above the ground so that no marker comes in contact with the ground during robot motion.

1.3 Error Propagation using Jacobian

The measurand, orientation angle is the function of four values. first two values are from initial position (A) and they are calculated by $x_1, y_1 = ({}^{A}x_2 - {}^{A}x_1), ({}^{A}y_2 - {}^{A}y_1)$. The last two values are from the final stopping position (B) of the robot and they are calculated by $x_2, y_2 = ({}^{B}x_2 - {}^{B}x_1), ({}^{B}y_2 - {}^{B}y_1)$. So our θ will be:

$$\theta = f(x_1, y_1, x_2, x_2)$$

So error analysis using jacobian will be

$$\Delta\theta = Jac(f) \times \begin{bmatrix} dx_1 \\ dy_1 \\ dx_2 \\ dy_2 \end{bmatrix}$$

The resolution of the measuring tape is 0.1 cm and since we will be using it for two pens for initial and final position so $dx_1 = dx_2 = dy_1 = dy_2 = \pm 0.2cm$. The python code for error propagation is given below

```
import sympy as sp
import numpy as np
```

```
import itertools

def calculate_jacob_matrix(x_1,x_2,y_1,y_2):
    #defining symbols
    x1,y1,x2,y2,theta = sp.symbols('x1,y1,x2,y2,theta')
    #dot product of robot position vectors
    dot_prod = x1*x2 + y1*y2  → shouldn't it be np.dot(vector1, vector2)?
    #magnitude of initial position
    mag_init = sp.root(x1**2 + y1**2,2)
    #magnutude of final position
    mag_final = sp.root(x2**2 + y2**2,2)
    #defining angle equation
    eq1 = sp.acos(dot_prod/(mag_init*mag_final))
    #angle equation in the matrix
    theta = sp.Matrix([eq1])
    #measure values in matrix
    meas = sp.Matrix([x1,y1,x2,y2])
    #jacobian of angle function
    jac_f = theta.jacobian(meas)
    jac_subs = jac_f.subs({x1:x_1, x2:x_2, y1:y_1, y2:y_2})
    return jac_subs


#finding all possible combinations of error
error_comb = []
for comb in itertools.product([ -0.2 , 0.2] , [-0.2, 0.2],[-0.2 , 0.2] , [-0.2, 0.2]) :
    error_comb.append(list(comb))

#example_of jacobian
jacobian = calculate_jacob_matrix(x_1 =2, x_2 = 2, y_1 = 2, y_2 = 0 )


#all of the possible angle errors
list_error = []

for error_meas in error_comb:
    list_error.append(np.dot(jacobian, np.array(error_meas)))

#finding maximum and minimum change in angle
```

```
max_angle = round(np.max(list_error)*180/np.pi,2)
min_angle = round(np.min(list_error)*180/np.pi,2)
print("Maximum angle error in degrees is {} deg".format(max_angle))
print("Minimum angle error in degrees is {} deg".format(min_angle))
```

The output is :

```
Maximum angle error in degrees is 11.46 deg
Minimum angle error in degrees is -11.46 deg
```