Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN2532 - Robot Design and Competition

Dc Motors for Your Mobile Robot

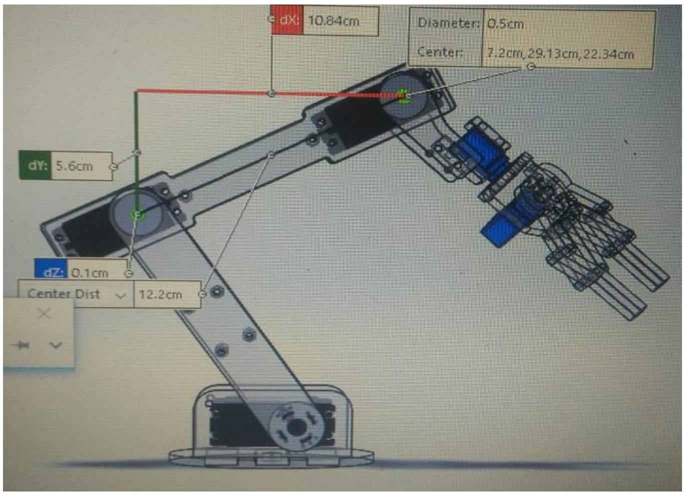
Submitted by

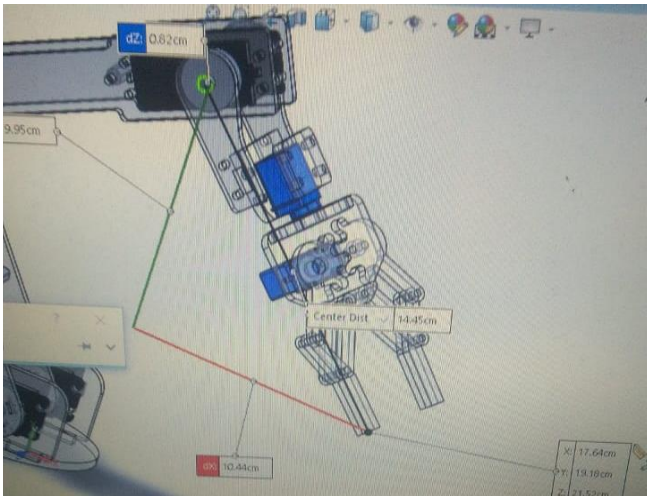
KANNANGARA D.N.	180301A
SIRITHUNGA M.R.A.	180609B
ARIYARATHNE H.D.M.P.	180045P
JAYAWEERA D.S.B.C.L.	180288L
HIROSHAN H.H.R.	180245E

1. usage of servo motors in your mobile robot

We must use servo motors in our Arm Mechanism. As shown in below photo our robot arm consists of three main joints which are used to move the arm point to the desired place and we have to use three servo motors in those three joints to achieve that task cause by giving needed movements to those servos we can move arm end point to required positions.

Other than that we have used another three servo motors in the end point of the Arm which we have to grab the box which we need and those three servos are used to grab the box tightly and move the box sideways.





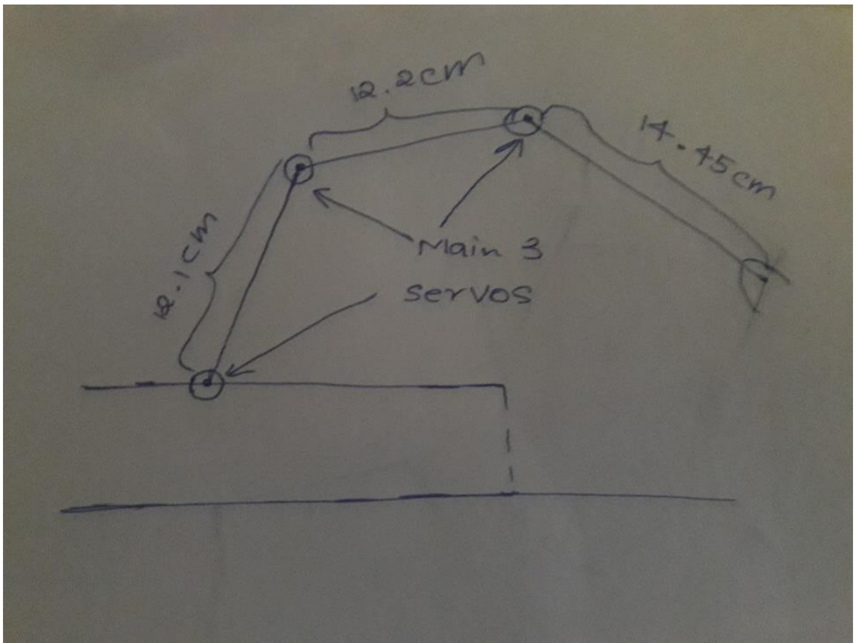
Full Arm

Gripping Part of the Arm

CALCULATIONS

We can represent this arm mathematically like below.

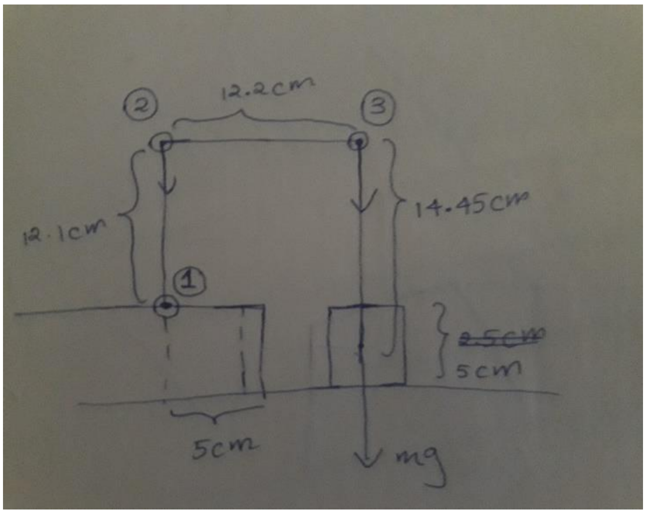
Since we must calculate the maximum torque which servos are going to experience it is enough to measure torque which three mains joint servos as obviously they must endure much more torque than gripping torques.

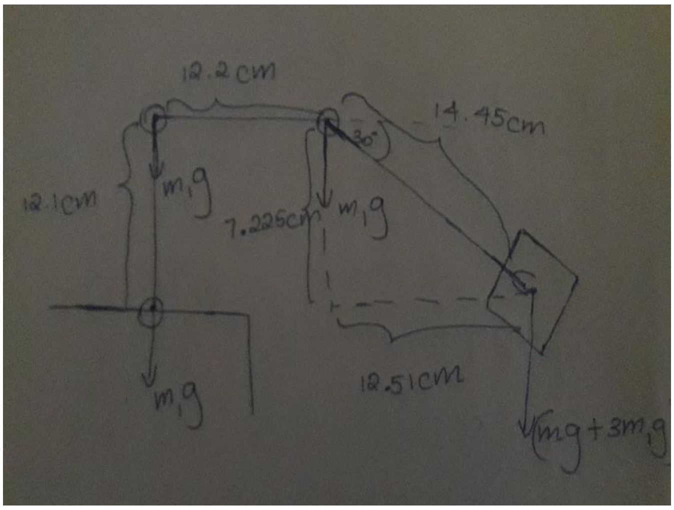


ASSUMPTIONS

- Masses of the Arm parts can be neglected.
- Average Mass of a servo motor is (m1= 15g)
- Mass of the box which should be lifted (m=80g)

The torque around any servo motor is high when the box is lifted by the arm. So, we are going to consider movements done when the box is lifted as they are the worst cases there could be.





Position 1 Position 2

Arm is going to move from position 1 to position 2 while lifting the box and as we understand **the worst-case scenario** can be seen at the position 2 and it affects to the motor 1 and motor 2. That means If we consider any other time and position torque which any servo will experience will be lower than this.

Hence, we did our calculations regarding position 2.

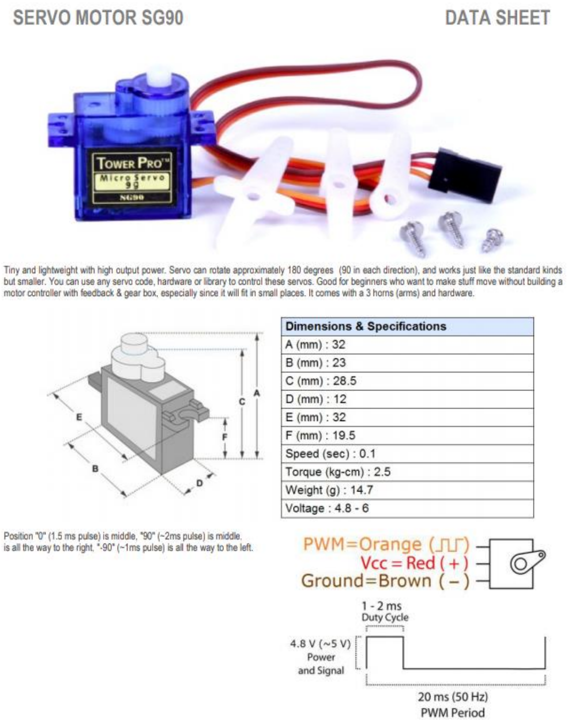
If T1 is the torque which motor 1 is going to experience at that point.

T1= m1* 12.2cm + (3m1 + m) * 12.51cm (Distances are shown in figures)

T1=1.747kg cm

Hence, needed **maximum torque is 1.747kg cm** and we must find servo motors which have this **stall torque.**

We were able to find two servos which have above stall torque in reliable operating voltages. Their specifications are in the next page.



SERVO MOTOR SG90 DATA SHEET

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

MG90S servo, Metal gear with one bearing

Tiny and lightweight with high output power, this tiny servo is perfect for RC Airplane, Helicopter, Quadcopter or Robot. This servo has *metal gears* for added strength and durability.

Servo 1

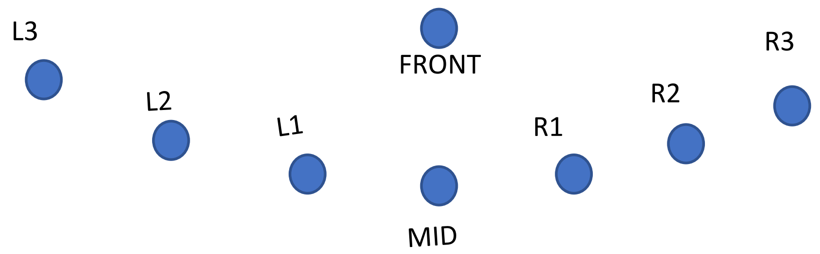
Servo 2

As we can see above both servos can endure the needed torque at manageable operating voltages. But Servo 1 is made of made of plastic while Servo 2 has metal gears. Also, Servo 1 cost Rs 250.00 while Servo 2 cost Rs 450.00.

Considering those factors, we thought to use Servo 2 to main 3 joints if we can afford them at the time as it is good to have added strength to those joints and use Servo 1 to other joints.

2. Line following algorithm with PID control: -

IR sensor array which we are going to use have eight sensors. Seven sensors are on the curve which have 75cm radius. So, it is nearly a line. Therefore, we can detect middle circle and 90 degrees turn properly. The arrangement of the sensor panel is given below.



Each sensor has been assigned a coefficient such as,

L1 = -100	R1 = 100
L2 = -200	R2 = 200
L3 = -300	R3 = 300

MID = 0

Each sensor on the IR panel give analogue reading. We can map that analogue reading to the digital number. If analogue value is converted to digital with 8-bit resolution, we can map each sensor reading to the value in between 0 - 255. So, we can get error function using these sensor value as follows

Left_error $= -100 \times \{\frac{255-l1}{255} + \frac{255-l2}{255} \times 2 + \frac{255-l3}{255} \times 3\}$

Right_error $= 100 \times \{\frac{255-r1}{255} + \frac{255-r2}{255} \times 2 + \frac{255-r3}{255} \times 3\}$

Error = Left_error + Right error

According to above equations, Left_error is always positive and Right_error is always negative. Error can be negative or positive.

Thresher hold: -

We need to find better thresher hold to identify whether a sensor is on the line or not. At the calibration stage we can determine the thresher hold by using maximum and minimum analogue read. In the pseudo code, we define thresher hold as “Th”

PID control: -

We can read the error of the line array sensor when array goes away the line. So, according to that reading using PID control method we can follow the line smoothly. When robot follows a straight line, left and right wheels have equal speed. If some error occur speed of the both wheels should be different to correct that error. So, robot can drive smoothly without oscillation by using PID control. Adjustment for the both wheels can be calculated as follows,

Current_time = Get_cuurent_time

Elapsed_time = Current_time – Previous_time

Rate_error = (Error - Last_error)/ Elapsed_time

Cumulative_error += $Error \times \ Elapsed\_time$

Adjustment = $Kp \ \times Error + Kd \ \times Rate_{error} + Ki \ \times \ Elapsed\_time$

Last_error = $Error$

Previous_time = Current_time

Kp and Kd values are constant. We can change these constant while tuning

Dashed line following: -

When we follow dashed line, we need to pass line regions and full black region without white lines. We know we met whit line after every 5cm. so we need the implement the code for the region which line is not available. We can force the robot go 5cm forward if no line is detected. Pseudo code for that part is given below.

If (Error < 90) AND (wall_detect == False) AND (FRONT > Th) {

Dashed_follow (Drive_speed)

}

Define Dashed_follow (speed) {

Drive Both motors 5 cm forward with the given speed.

}

Full Pseudo code for line following: -

Begin

Define Drive (speed); {

#get the position error

Left_error = $-100 \times \{\frac{255-l1}{255} + \frac{255-l2}{255} \times 2 + \frac{255-l3}{255} \times 3\}$

Right error = $100 \times \{\frac{255-r1}{255} + \frac{255-r2}{255} \times 2 + \frac{255-r3}{255} \times 3\}$

Error = Left_error + Right error

```
If (MID < Th) {
If (Error < 90) AND (wall_detect == False) AND (FRONT > Th) {
Dashed_follow (Drive_speed)
}
Else {
 Wall_follow (Drive_speed) //call the wall following function
}
}
Else {
#PID tuning the adjustmen
Current_time = Get_cuurent_time
Elapsed_time = Current_time – Previous_time
Rate_error = (Error - Last_error)/ Elapsed_time
```

Cumulative_error += $Error \times \ Elapsed\_time$

Adjustment = $Kp \ \times Error + Kd \ \times Rate_{error} + Ki \ \times \ Elapsed\_time$

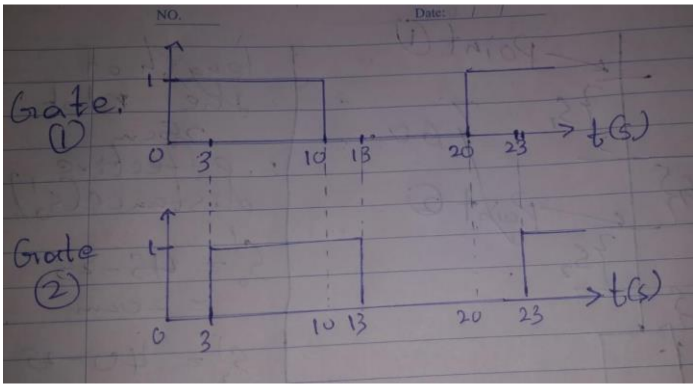
Last_error = $Error$

```
Previous_time = Current_time

left_motor.drive(Drive_speed+ Adjustment)  // call the motor  module function by giving the speed
right_motor.drive(Drive_speed - Adjustment)
}}
Define Dashed_follow (speed) {
Drive Both motors 5 cm forward with the given speed.
}

Drive(x)
End
```

3.Overcome the passage of synchronized gates



Length of the robot = 25 cm

Maximum speed (Vmax) = 5 cm/s

Time taken to reach maximum speed (t0) = 2 s

s/t = (u+v)/2

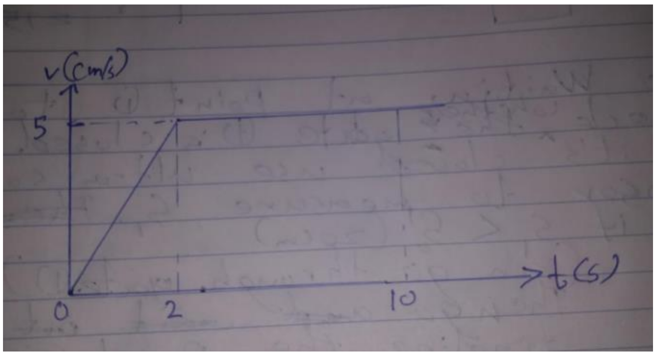
s/2 = (0+5)/2

s = 5 cm

Thus, distance travelled when coming to maximum speed from start = 5 cm

S1 = 5 + 8 * 5

S1 = 45 cm



Reducing the length of the robot,

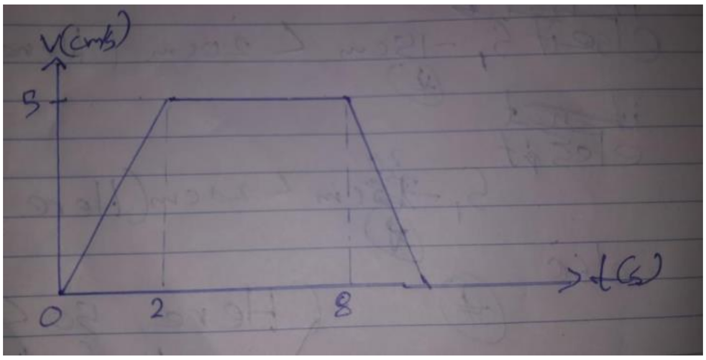
X1 = 45 – 25

X1 = 20 cm

Maximum distance that robot can travel within 10s If stopping (S2),

S2 = 5 + 6 * 5 + 5

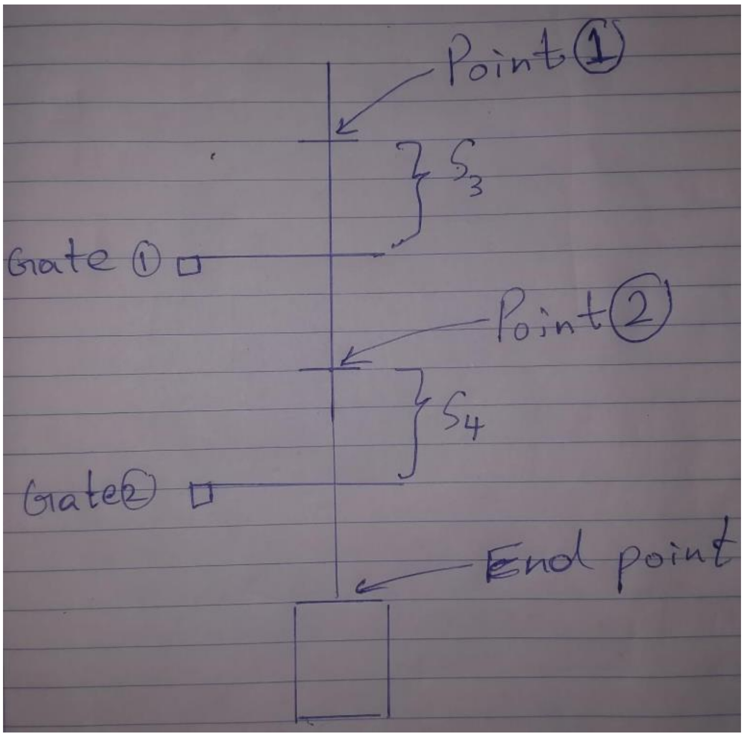
S1 = 40 cm



Reducing the length of the robot,

X1 = 40 – 25

X1 = 15 cm



The robot will be waiting at Point 1 to check whether Gate 1 is closed or not.

If closed,

  Use ultra sonic sensor to measure the distance between Gate 1 and the robot (S3),

  If S3 < 20,

    (Then the robot can go through the Gate 1)

    After the gate is opened go until finds Point 2

  Else,

    Go 20cm and stop

    If S3 – 20 < 20,

      (Then the robot can go through the Gate 1)

      After the gate is opened go until finds Point 2

    Else,

      Go 20cm and stop

      After the gate is opened go until finds Point 2

Else,

  Wait till the Gate 1 closes and do the above procedure.

Now the robot in Point 2.

The robot will be waiting at Point 2 to check whether Gate 2 is closed or not.

If closed,

Use ultra sonic sensor to measure the distance between Gate 2 and the robot (S4),

If S4 < 20,

(Then the robot can go through the Gate 2)

After the gate is opened go until finds the End point

Else,

Go 20cm and stop

If S4 – 20 < 20,

(Then the robot can go through the Gate 2)

After the gate is opened go until finds the End point

Else,

Go 20cm and stop

After the gate is opened go until finds the End point

Else,

Wait till the Gate 2 closes and do the above procedure.

4. Every motor type which is encountered during the Robot Motion lectures

	DC Motors		Servo Motors	Stepper Motors
	Brushed Motors	Brushless Motors		
Torque	Moderately flat	flat	Good	Good
Speed	Moderate	Higher	Low	Low
Phases	Single phase	Up to three phases	Three phases	Two phases or Five phases
Commutation	Brushed commutation	Hall Commutation	Hall Commutation	External commutation
Rotor	Winding	Magnet	Winding	Magnet
Stator	Magnet	Winding	Magnet	Winding
Terminals	Two	Three	Three	Five or six
Magnetic field generation	Permanent magnet	Stator	Stator	Rotor
Angular resolution	Depends on encoder	Poor	Depends on encoder	Good
Motor complexity	Simple	Complex	Moderate	Complex
Control mechanism	Simple & inexpensive	Complex & expensive	Simple	Complex
Control complexity	Not essential	Controller is always required	Controller is always required	Controller is always required
Use of H bridges	Not essential	Always required	Not essential	Always required
Driving modes	N/A	N/A	180 or 360 degrees	Full stepping or micro stepping
Cost	Low	High	Moderate	High
Advantages	Low cost	Operate in high speeds	Best in constant load	Breaking system on shaft
Disadvantages	Generate electromagnetic noise	Control complexity	Poor in load variation	High power consumption
Commercially available product	Easily available	Available in selected stores	Available in selected stores	Available in selected stores