# Motion Subsystem

## Solutions and Relevant Algortihms

### Plan A

Motion subsystem consists of motors for driving the robot on the play field, wheels, other assembly parts such as encoders and gearboxes attached to these motors and also driver IC which convert inputs from the Main Processor Subsystem into meaningful inputs for the hardware part of the motion subsystem. Motion susbystem also sends feedback to Main Processor Subsystem.

#### Motor Selection

Motor selection is highly crucial for building a stable and accurately driven robot. Before choosing the most suitable featured motor we have to decide which motor type will be most suitable for our purpose. Available motors in the market are DC Motors, Stepper Motors and Servo Motors.

##### Servo Motors

Using servo motors for driving the robots seems unreasonable since the working principle of servo motors are rotating the shaft to a desired angle with respect to its input duty cycle.

##### Stepper Motors

Since steppers move in precise repeatable steps, they excel in applications requiring precise positioning such as 3D printers. However, we don’t need precise positioning in motion subsystem. Normal DC motors don't have very much torque at low speeds. A stepper motor has maximum torque at low speeds, so they are a good choice for applications requiring low speed with high precision. This feature is also not applicable for motion subsystem.

##### DC Motors

DC motors in the market are the ones that we will most likely use. We can modify these with respect to our torque and speed regulations. These are also easy to control with a simple encoder integration. Other than reading how many turns have been made by the motors we may also consider using a motor driver IC to control how fast our motor turns as well. For DC motors driver IC simply turns the voltage on and off to vary the speed of the motor. PWM is used as driver input which can be easily produced using an Arduino.

Therefore, we concluded that for motion subsystem DC motor is the best option. Since we need higher torque but lower rotational speed, brushed DC motor is chosen as the best fit.

Before selecting the suitable motor we need to consider some calculations. Such as total weight, torque and speed estimations of the robot. Total weight estimation for the robots exluding the motion subsystem motors are as in Table xx. Suitable motor for shooting subsystem is included in Shooting Components section.

Table 1:Weight Estimation Table of the Robot

|  |  |
| --- | --- |
| **Components** | **Approximate Weight (g)** |
| Arduino MEGA | 37 |
| Powerbank | 200 |
| LiPo Battery | 230 |
| Chassis | 150 |
| Radio | 10 |
| Shooting Components | 500 |
| Cables | 50 |
| Other Circuitry | 100 |
| **Total Weight** | **1227** |

For speed estimation we started the process with a reasonable assumption. For our purpose we assume that the **nominal robot speed** is **0.4m/s**. It should be enough. The maximum speed will depend on other factors, which are yet to be identified. In order to calculate the rpm of the shaft needed to drive the robot with 0.4 m/s velocity we can use the following formula:

Where NT is traction wheel rotation speed, vN is nominal robot speed and DW is wheel diameter. We take DW as 0.08 m for starter. Therefore, calculated NT is approximately 95 rpm on the wheel side. Since we are planning to use gearbox in order to increase the torque and decrease revolution per minute, rpm of the shaft will be 48 times the gear box ratio and the torque produced by the shaft should be 1/ (gearbox ratio) times the needed torque to move the robot with desired acceleration. We chose gearbox ratio as 60:1 at first. Then needed rpm for the desied nominal speed became 95 x 60 = 5700 rpm. Considering DC motor charateristics in Figure xx which satisfies our speed and torque requirements. At ~5500 rpm produced torque is ~2.5 oz-in which is equal to 0.0176 Nm. One important thing is that this torque is on the shaft side. Since we took gear ratio as 60:1, torque on the wheel side becomes 60 x 0.0176 = 1.056 Nm. Considering the radius of the wheel, produced force at this speed is equal to 1.056 Nm / 0.04m = 26.4 N. However since the efficiency is 70 % at this level of speed, only 70 % of the torque and force will be transferred (~18 N)which is highly enough for driving our robot taking the overall weight as 2kg. Since without the DC motors weight was as calculated in Table xx. We took each DC motor as approximately 300 gr.



Figure 1: RS-555 12 V 6100 rpm Brushed DC Motor Characteristic [5]

#### Encoder Selection

After deciding on DC motor specs we need to choose an encoder to give feedback to the Main Processor Subsystem. Therefore we can directly send commands to the driver IC how many turns that the shaft should complete to reach to the desired location. After some research on encoder types there are two possible types of encoders in the field which are:

* Incremental Encoders
* Absolute Encoders

Both type of encoder components and a pulse train produced by Incremental Encoder is as in Figure xx.

Figure 2: Incremental (Right) & Absolute Encoder (Left) Components [6]

First type is used for speed and position tracking. In this type turn count is measured easily but position accuracy is not available.

Second type of encoders are used for accurate position tracking since each position have specific ID so that we can simply know the exact position of the shaft attached to the encoder.

Incremental encoder seems to be more useful for our purpose. Since in motion subsystem we will command the motors to turn specific number of turns to reach the desired position or achieve the desired turn to the left or right. The encoder type we consider using, has Hall-effect sensor in it and counts up to 44 for one single turn but it should be multiplied with the gear ratio. Hall effect encoders use magnetic phased arrays that contain hall sensor elements arranged in a pattern to match a magnetic wheel. A signal is produced as the sensor passes over the magnetic field which is then interpolated to the desired resolution. The representation is as in Figure xx.



Figure 3: Magnetic Encoder Representation with Hall-effect Sensor [7]

For our specific type of encoder and gearbox ratio (60:1) every 2640 count means one full turn. Both CW and CCW is available. This module will be used in motor driving part. It will also prevent the turn mismatches between the right and the left wheel for example. We will use this encoder count as a feedback between tires to make our robot move more accurate and straight. Test code and related outputs are presented in the following parts.

#### Wheel Selection

There are different variety of wheels in the market such as **standard, orientable, ball, and omnidirectional wheels. Our main solution for wheel selection is two standard wheels and also two ball wheels to stabilize the robot chassis. Wheels-chassis integration is represented as in Figure xx.**

#### Ä°lgili resim

Figure 4: Wheel & Chassis Integration Representation [8]

By using the integration illustrated in Figure xx above, we will drive the robot via Differential Drive method. This method can be visualized as in Figure xx.



Figure 5: Differential Drive Kinematics [9]

#### Motor Driver Selection

Our main solution includes using L298N driver which can be easily integrated with Arduino. The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A. We will drive two DC motors using this IC with respect to the pin connections as seen in Figure xx. Related test codes and results are presented in the following parts.



Figure 6: Pin Connections of L298N Driver IC [10]

## Level Risk Assessments

There are several risks of our main solution approaches such as motor torque and speed specs may be different in practice therefore, we may need to reconsider our DC motor selection. Other than that, standard wheels may not be more efficient than omnidirectional wheels or we may not need two ball wheels. Encoders we chose may not be durable or since our main solution is for using magnetic encoders it may be affected from outside effects therefore it may give wrong readings to our Main Processor Subsystem. This will prevent our robot from moving accurately.

### Plan B

Alternative solutions for Motion Subsystem are as below:

#### Motor Selection

DC motors with different specs such as rpm, torque, power and efficiency values can be used alternatively.

#### Encoder Selection

Encoders with different CPR (counts per revolution) values or different types can be used to isolate the encoders from environmental effects, with DC motors.

#### Wheel Selection

Omnidirectional or orientable wheels can also be used with DC motors.

#### Motor Driver Selection

L293DNE IC as in Figure xx, can be used to drive the DC motors. This IC is a quadruple high current half-H driver. These devices are designed to drive a wide array of inductive loads such as relays, solenoids, DC and bipolar stepping motors.



Figure 7: L293DNE Pinout [11]

## Error Sources

Possible error sources for motion subsystem are as below:

* Since we are using two DC motors, we may face calibration issues. Even though these two motors are identical theoretically, in practice they will have minor differences. Therefore, we should consider this and update our algorithm respectively. A feedback should be available to prevent this calibration differences.
* Opening voltages of these DC motors can be slightly different. Therefore when we apply a voltage for starter one motor can start rotating whereas the other does not. We should conduct relevant tests to compare two DC motors.
* We should adjust the center of gravity therefore weight should be equally transferred into the wheels and motors respectively. We may not place the center of gravity well enough which will cause one motor to turn more than the other. We may need to use a PID controller and tune it to obtain optimal results.

## Test Results

In Motion Subsystem we implemented a tutorial for both encoder readings and also motor driving. Encoder reading test code is as in Figure xx & Figure xx.

A screenshot of a cell phone

Description automatically generated with medium confidence

Figure : Encoder Reading Code Part 1



Figure : Encoder Reading Code Part 2

As explained before we must see 2640 after one full turn is completed both in clockwise and also in counterclockwise direction. The results were close enough to our expectation. There is an error margin at the results. Therefore, we need to take these error margins into consideration when driving the DC motors and writing our main algorithm.

Second tutorial was for driving the motors using L298N. Test Algorithm can be seen in Figure xx.



Figure 8: DC Motor Driving Tutorial Code Part 1



Figure 9 : DC Motor Driving Tutorial Code Part 2

We controlled the speed of the motor using a potentiometer and change the rotation direction using a push button. In the loop section we start by reading the potentiometer value and then map the value that we get from it which is from 0 to 1023, to a value from 0 to 255 for the PWM signal, or that’s 0 to 100% duty cycle of the PWM signal. Then using the analogWrite() function we send the PWM signal to the Enable pin of the L298N board, which actually drives the motor.

Tutorial code mentioned above is implemented with the circuit schematic as in Figure xx.



Figure 10: Circuit Schematic for Motor Driving Tutorial [12]

# [5] RS-555 12V 6100 RPM Brushed DC Motor, Retrieved from https://www.robotshop.com/ ca/en/rs-555-12v-6100-rpm-brushed-dc-motor.html

[6] , [7] Encoder Guide , Retrieved from https://www.anaheimautomation.com/manuals/forms/encoder-guide.php

# [8]  Robot Tracing Strong Magnetic Motor Car RT-4 , Retrieved from https://www.aliexpress.com/item/Smart-Car-Chassis-2wd-Robot-Tracing-Strong-Magnetic-Motor-Car-RT-4-Avoidance-Car-with-Code/32809659306.html

# [9] Differential drive with continuous rotation servos and Arduino, Retrieved from https://42bots.com/tutorials/differential-steering-with-continuous-rotation-servos-and-arduino/

# [10] , [12] Arduino DC Motor Control Tutorial – L298N | PWM | H-Bridge, Retrieved from https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

## [11] L293D Dual H- Bridge Motor Driver IC, Retrieved from https://kitskart.com/product/l293d-motor-drive-dip-ic/