Hardware Challenge - STRATEC

1. Introduction

For this challenge I created a small Arduino project having the following components: Arduino UNO board, L298N motor driver, TT Gearbox motor, rack and pinion (3D printed), battery, jumper wires. The functionality of this system is to control the movement of the pinion using input from a user and data, e.g. speed of the shaft and current position of the pinion, to be displayed on a simple UI. All of these components were immediately available, coming also with constrains and quite a challenge to find the optimal solution with those.

1.1 Block Diagram

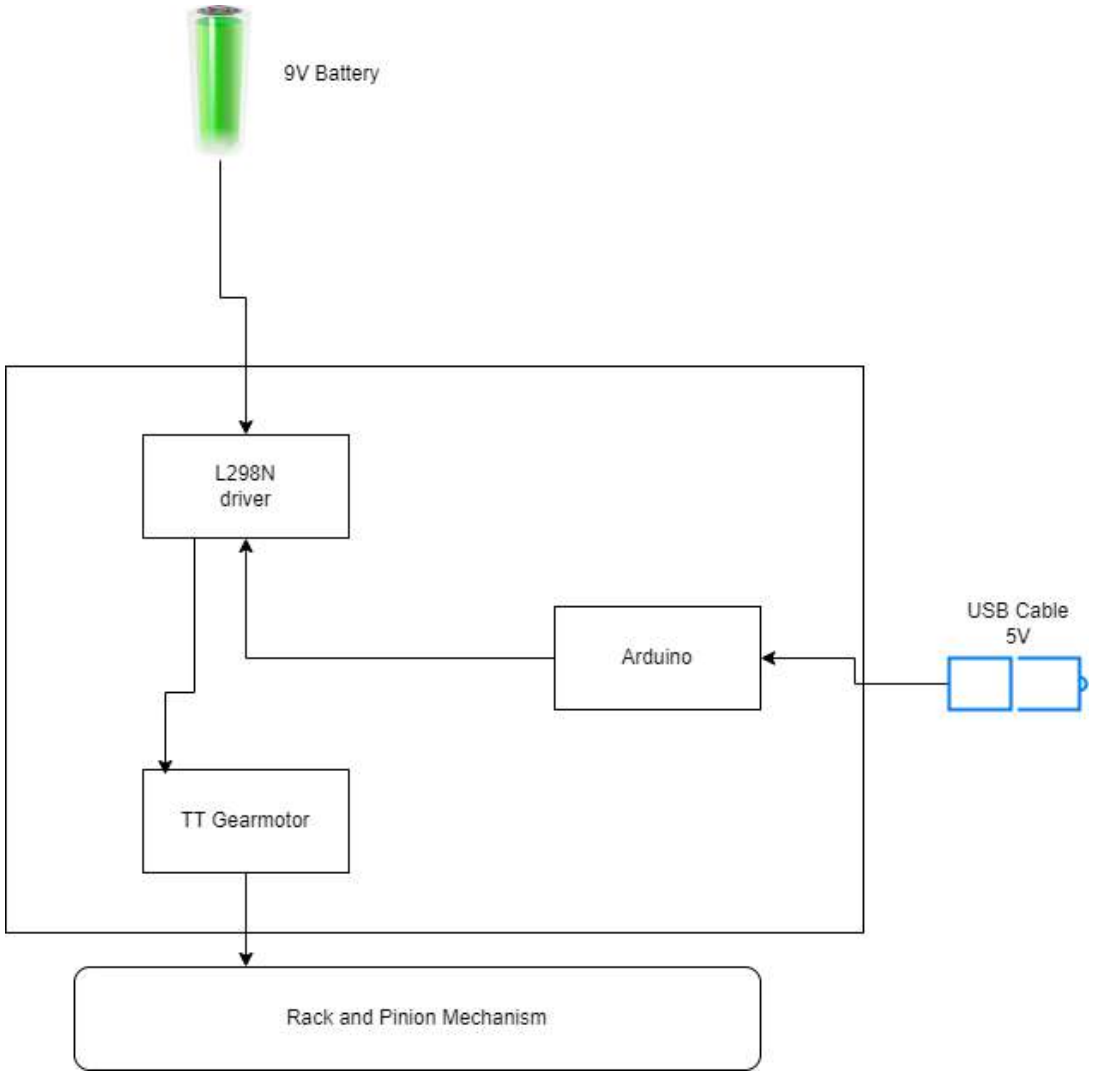

Figure 1.1: Diagram of the whole system

The power supply for this system consists of 5V from the USB cable, connected to a PC USB Port, that powers the Arduino UNO and a 9V battery for the motor driver.

2. Design and implementation

The hardware components are the ones mentioned in the 1. The happy flow is the following: the Arduino UNO board received user's input, the next position in which the pinion should arrive, confirms its and then data is transmitted to the L298N driver, which controls the gearbox motor, to spin the pinion at a specific value and time, recording the shaft's velocity, reaching and indicating the current position. All other warning messages or data to be computed are done through the Serial Monitor, available from Arduino IDE, as a basic User Interface. The positions chosen are '1', '0' and '-1' due to limited length of the rack, torque of the gearmotor and are chosen the same as values are represented on the Ox axis.

Right now, the pinion is arriving at the specified position, but we really do not know if it is the exact position, so, as an addition, a feedback mechanism can be added in which an Infrared (IR) sensor is placed at one of the ends of the rack, measuring the distance from it to the pinion (at the current position and at the desired position) and through a simple subtract from current displayed to the next one is the travelled distance and knowing the speed and time in which the pinion is moving, compare it and if the computed distance, from the equation: $d = \frac{v}{t}$ (d - distance, v - velocity, t - time), is equal to the measured one from the sensor, then the pinion is at the desired destination.

The system also uses an internal ROM memory (EEPROM) to store the current position in order to save it if there are power cycles or unexpected shutdowns.

2.1 Source code

Will be introduced into the A section.

3. Tests and Validations

3.1 Validations

Most of the validations are made to ensure that as most corner cases as possible are covered, also fro the hardware ones that are in coordination with software. Those are made to guarantee that the pinion goes only to known position, not to exceed the rack's limits, not moving the pinion if the same position is introduced, input the right number and confirming letter and also, if the position is disaffirmed, the system starts from the beginning.

3.2 Testing

Images with data from the Serial Monitor and the states of the physical system will be presented:

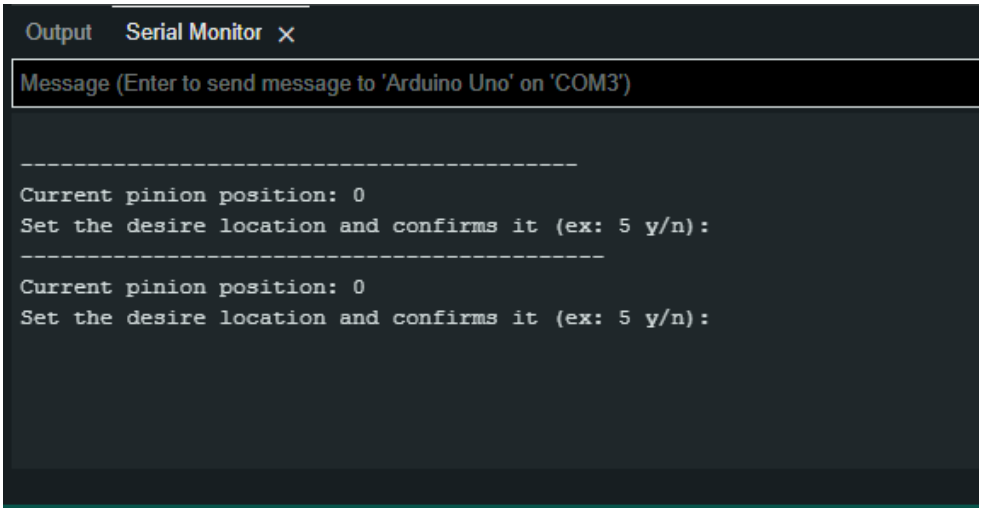

Figure 3.1: Initial position 0 - Virtual view

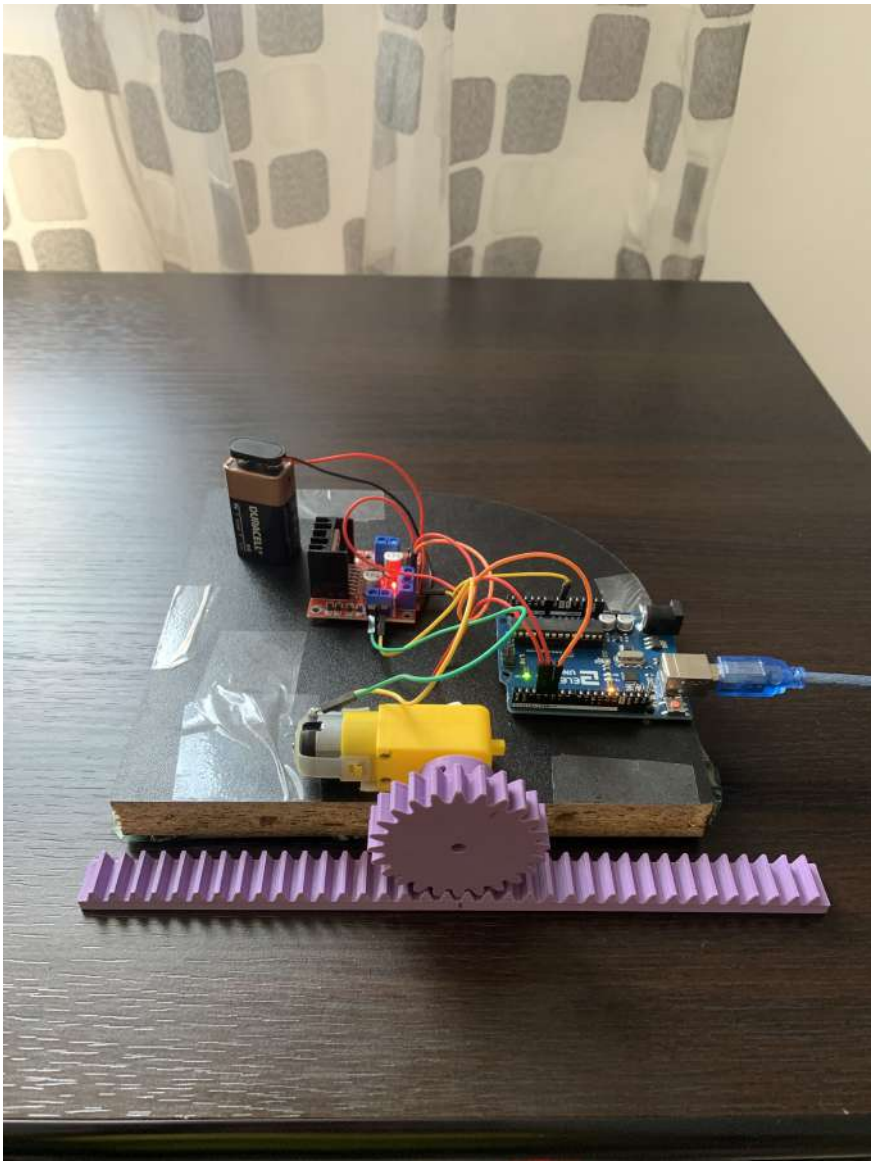

Figure 3.2: Initial position 0 - Physical view

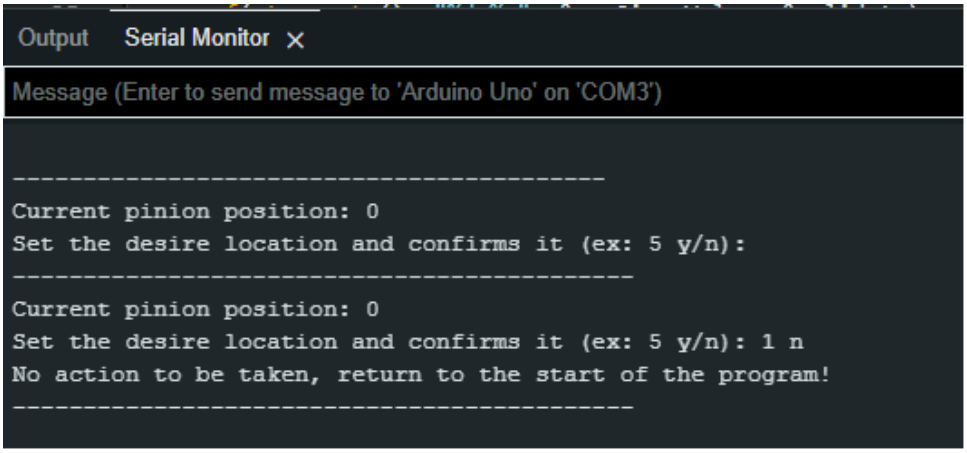

Figure 3.3: Not confirming the desired choice

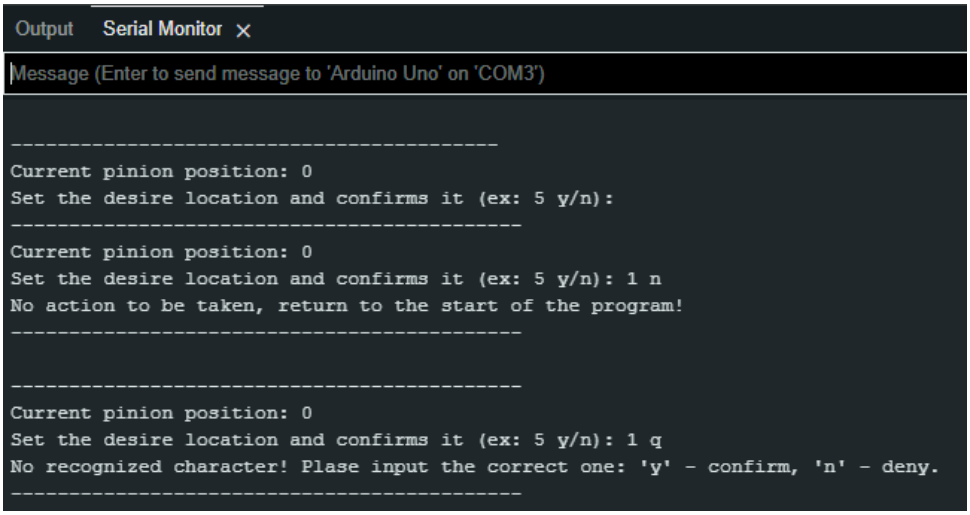

Figure 3.4: Wrong characters inputted

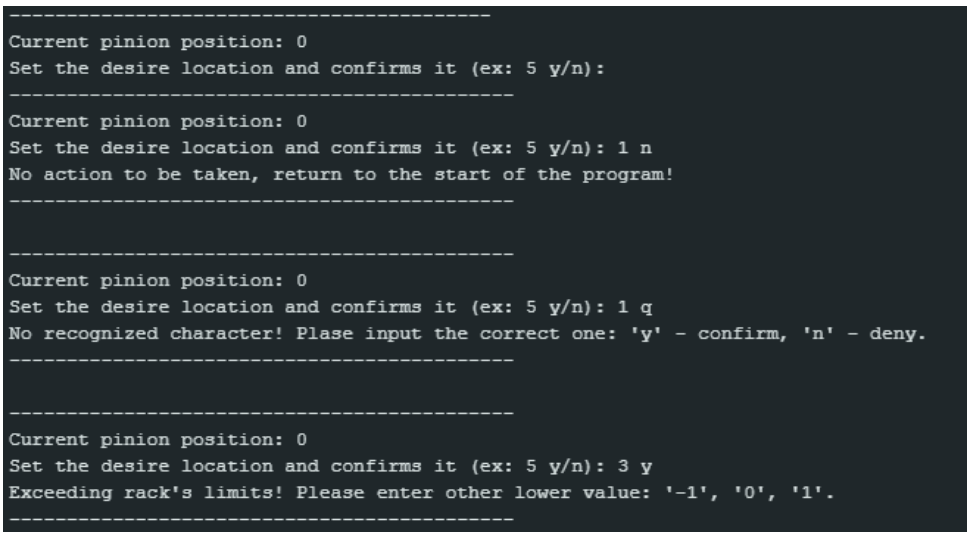

Figure 3.5: Unknown position

```
-----------------------------------------
Current pinion position: 0
Set the desire location and confirms it (ex: 5 y/n): 1 y
The speed is (cm/s): 17.97 0.
-----------------------------------------

-----------------------------------------
Current pinion position: 1
Set the desire location and confirms it (ex: 5 y/n):
```

Figure 3.6: Choosing the known position 1

Figure 3.7: Physical view of the position 1

```
-----------------------------------------
Current pinion position: 1
Set the desire location and confirms it (ex: 5 y/n): 1 y
No change in position, staying steady!
-----------------------------------------


-----------------------------------------
Current pinion position: 1
Set the desire location and confirms it (ex: 5 y/n): -1 y
The speed is (cm/s): 17.97 0.
-----------------------------------------


-----------------------------------------
Current pinion position: -1
Set the desire location and confirms it (ex: 5 y/n): 0 y
The speed is (cm/s): 17.97 0.
-----------------------------------------
```

Figure 3.8: Choosing position 1 again - no action and input position -1

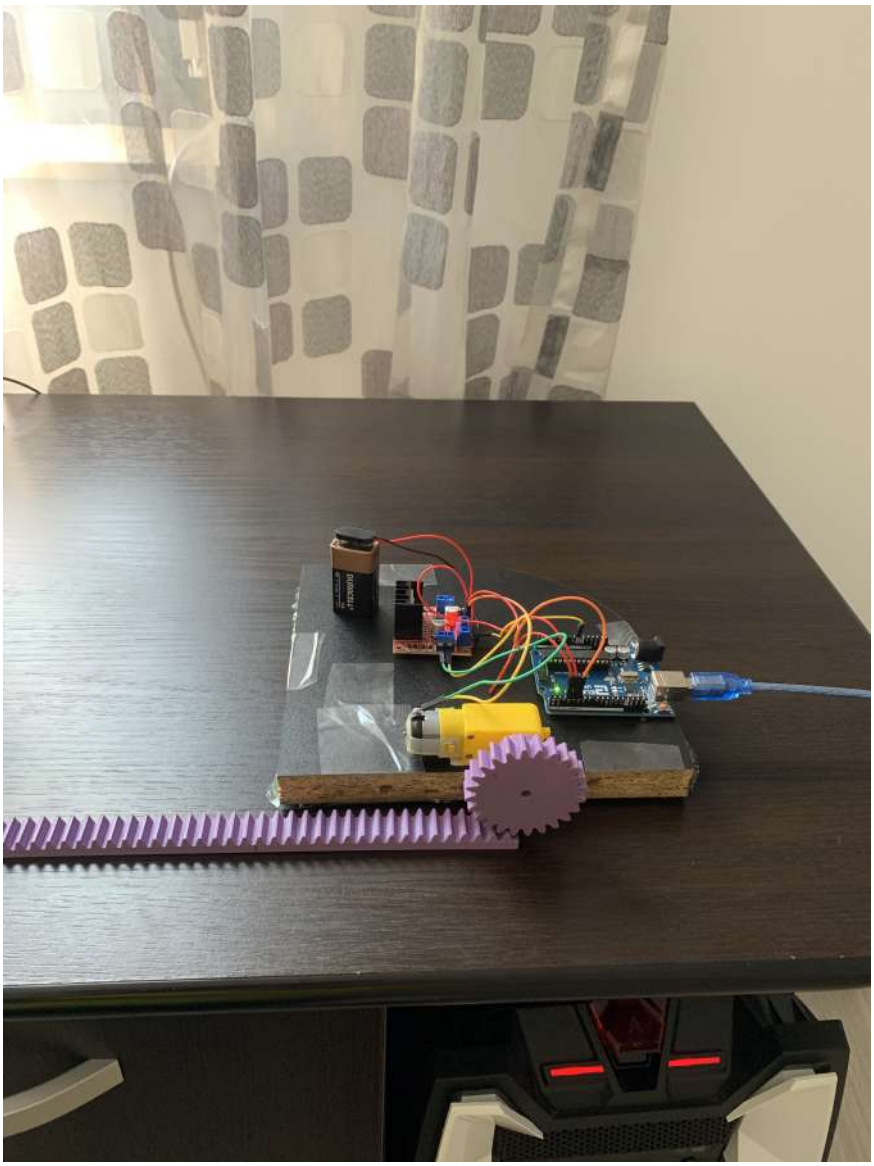

Figure 3.9: Physical view of the position -1

A. Source code of the project

```
#include <EEPROM.h>

const int IN1 = 3;
const int IN2 = 4;
const int ENA = 5;
const int address = 0;
const float radius = 0.026;
int rotationVal = 74;

int posGivenValue;
int8_t currentPos;
char validate;


void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  Serial.begin(115200);
  while (!Serial) {
    ;  // wait for serial port to connect. Needed for native USB port only
  }
}

void loop() {
  Serial.println("\n-----------------------------------------");
  Serial.print("Current pinion position: ");
  currentPos = (int8_t)EEPROM.read(address);
  Serial.println(currentPos);

  Serial.print("Set the desire location and confirms it (ex: 5 y/n): ");
  while (Serial.available() == 0) {}
  //wait for input into the Serial Monitor
  String str = Serial.readString();
  sscanf(str.c_str(), "%d %c", &posGivenValue, &validate);
  Serial.print(String(posGivenValue) + " ");
  Serial.println(validate);

  switch (validate) {
    case 'n':
      Serial.print("No action to be taken, return to the start of
      the program!");
      break;
    case 'y':
      rotaryMotionToLinerMotion();
      break;
    default:
      Serial.print("No recognized character!
```

```
      Plase input the correct one: 'y' - confirm, 'n' - deny.");
      break;
  }

  Serial.println("\n-----------------------------------------");
}

void rotaryMotionToLinerMotion() {
  if (posGivenValue > 1 || posGivenValue < -1) {
    Serial.print("
    Exceeding rack's limits! Please enter other lower value: '-1', '0', '1'
    .");
  } else if (currentPos == posGivenValue) {
    Serial.print("No change in position, staying steady!");
  } else if (currentPos > posGivenValue) {
    gearMotorMotion(HIGH, LOW);  //Counterclockwise - CCW
  } else {
    gearMotorMotion(LOW, HIGH);  //Clockwise - CW
  }
}

void gearMotorMotion(uint8_t val1, uint8_t val2) {
  int rpmVal = map(rotationVal, 0, 255, 0, 230);
  float linerVelocity = (2 * PI * radius * rpmVal * 100) / 60;
  //formula to transform rpm to linear velocity
  //from m/s to cm/s
  Serial.print("The speed is (cm/s): " + String(linerVelocity) + " ");

  digitalWrite(IN1, val1);
  digitalWrite(IN2, val2);
  analogWrite(ENA, rotationVal);

  int sum = 0;
  if (val1 == HIGH) {
    sum = (currentPos - posGivenValue) * 600;
  } else sum = (posGivenValue - currentPos) * 600;  //600 ms is the perfect
                                    // delay not to exceed the rack's limits
  delay(sum);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);

  Serial.print(String(0) + ".");  //at this stage, the motor is surely
  // not functioning, thus the value is hardcoded to be 0

  EEPROM.update(0, posGivenValue);
}
```