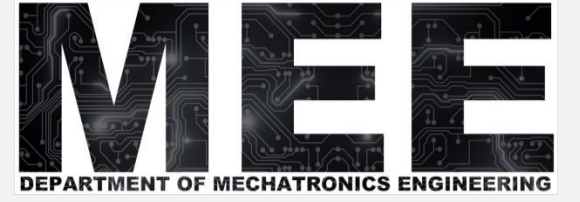


DESIGN AND DEVELOPMENT OF HOMOGENEOUS SELF RECONFIGURABLE ROBOTS



Mehmet Cahit KANKAYA, Ardl AKÇA, Doğukan TEZOĞLU
Supervisor: **Assoc. Prof. Dr. Erkin GEZGİN**



Abstract

The topic of our project is modular self-reconfigurable robots, abbreviated as MSRR. These robots are formed by the assembly of similar modules in different configurations, both in terms of appearance and structure. In addition to their capability to create different formations, they can easily self-replace malfunctioning, damaged or worn-out modules with new ones without the necessity of complete system or operation halt. Although possible assembly formations may vary with respect to different tasks, this study focuses on two different main formations as a mobile robot platform, and a multi degrees of freedom serial manipulator. In light of this, our objectives are ranging from pick-and-place tasks, which involve the actions of grabbing, carrying, and placing objects, to mobile robot manipulation.

1. Introduction

Our project focuses on Modular Self-Reconfigurable Robots (MSRR) that can be formed by combining similar modules in different configurations. The two main formations we aim to achieve are a dexterous mobile platform and multi degrees of freedom serial manipulator. The mobile platform consists of main modules, power modules, and wheel modules, functioning as a vehicle. The serial manipulator formation involves main modules, power modules, and an end effector for pick-and-place tasks. To enable movement and manipulation, the main module incorporates a gear mechanism with high-torque motors and motor drivers. Connections between modules are established through magnetic coupling and electrical connections using neodymium magnets and spring-loaded pins. The power module contains batteries, while Arduino Mega processors facilitate communication between modules. The prototype requires human assistance for formation changes, but future models will autonomously switch formations and perform various tasks concurrently. In conclusion, MSRR offers innovative possibilities in robotics, as demonstrated by our Mechatron project.

2. Conceptual Design

Coarse structural representations of selected modules can be seen below in preliminary project drafts along with detailed inner design.

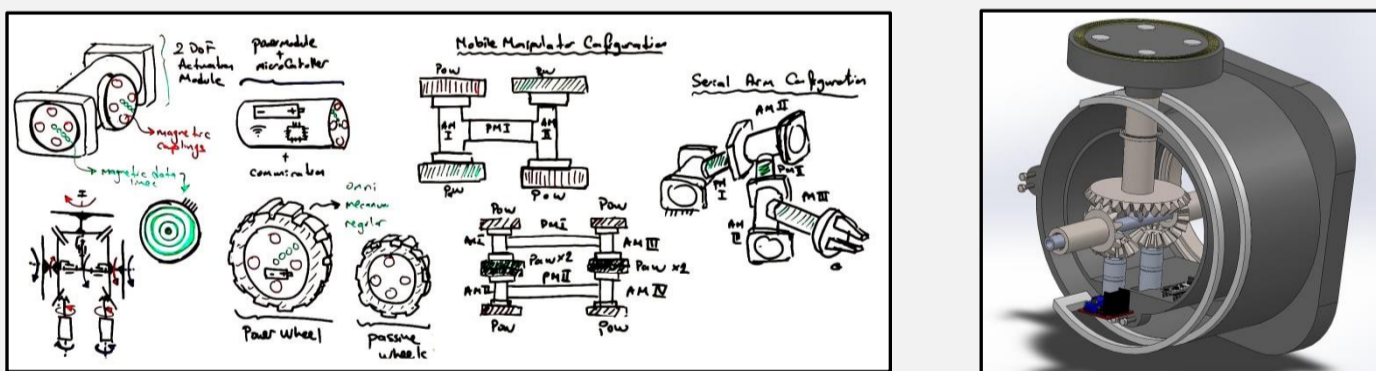


Figure 1: Conceptual Draft and Detailed Design

3. Theoretical Calculations

Diameters of Gears:

Small Gear: $D = 50\text{mm}$

Medium Gear: $D = 57\text{mm}$

Big Gear: $D = 110\text{mm}$

Circumferences of Gears:

Small Gear: $C = \pi D = 314.16\text{mm}$

Medium Gear: $C = \pi D = 358.14\text{mm}$

Big Gear: $C = \pi D = 691.15\text{mm}$

Angles of Gears:

Small Gear: $\alpha = (26.18/D) 180^\circ = 93.95^\circ$

Medium Gear: $\alpha = (25.58/D) 180^\circ = 80.78^\circ$

Big Gear: $\alpha = (24.68/D) 180^\circ = 40.49^\circ$

Angular Velocities of Gears:

Small Gear: $W = 100\text{ rpm}$ for motor

Mid Gear: $W = 100\text{ rpm}$ (N_1/N_2) = 85.71 rpm

Big Gear: $W = 100\text{ rpm}$ (N_1/N_3) = 42.86 rpm

Motor Torque:

$T = K_t I = 0.805\text{N.m/A} \cdot 5.5\text{A} = 4.428\text{ Nm}$

4. Experimental Setup

3D models of the proposed system modules were designed in SolidWorks prior to the prototype manufacturing.

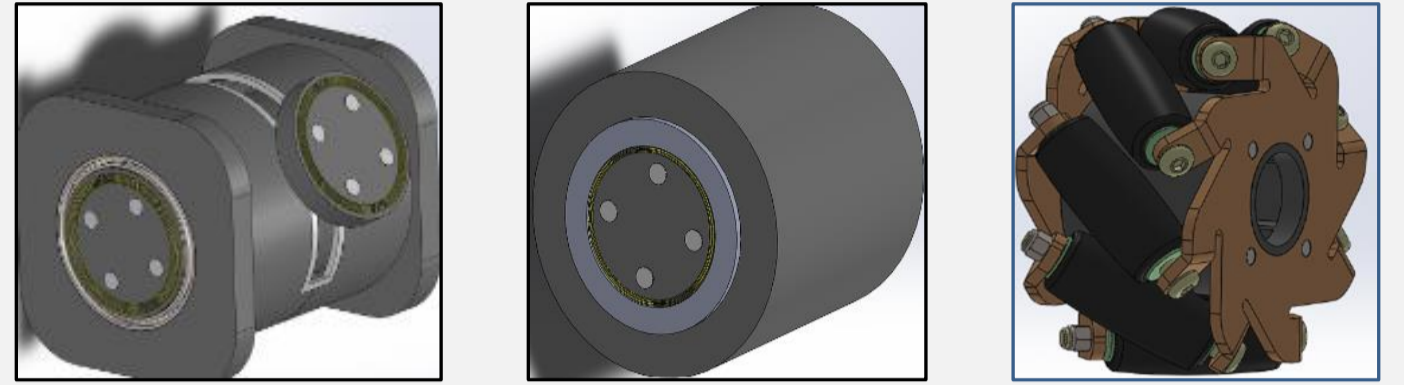


Figure 2: 3D CAD drawings of the main module, power module, and wheel module, respectively, are as follows.

The mobile manipulator configuration and serial arm configuration were created by the modules were conceptualized by using SolidWorks.



Figure 3: Serial Arm and Mobile Manipulator Configurations

This study introduces a master and slave code written to realize simultaneous control of two motors and also to communicate with other ESP32 devices. It is used to control motors and communicate with other devices using these ESP32 microcontrollers.

```
// ACCESS POINT MASTER
#include "WiFi.h"
const char* ssid = "Mekatronix";
const char* password = "mekatronix";
IPAddress local_ip(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);

WiFiServer server(80);

WiFiClient client(2);
String FORWARD = "F";
String BACKWARD = "B";
String STOP = "S";
String DOWN = "D";
String LEFTMODULE = "L";
String RIGHTMODULE = "R";

uint32_t aktif_pwm;
uint32_t pasif_pwm;

String sendData;
void setup() {
  // put your setup code here, to run once:
  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(local_ip, gateway, subnet);
  server.begin();
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (client) {
    Serial.println(client);
  }
  if (client) {
    for (int i = 0; i < 2; i++) {
      if (client[i] && client[i].connected()) { // Check if client is connected
        client[i].println(sendData);
      }
    }
  }
  sendData = combineData(IP, 192, 168, 1, 22);
  for (int i = 0; i < 2; i++) {
    if (client[i] && client[i].connected()) { // Check if client is connected
      client[i].println(sendData);
    }
  }
}

String combineData(String yont, uint32_t aktif, uint32_t pasif, String yont2, uint32_t aktif2, uint32_t pasif2) {
  return LEFTMODULE + yont + String(aktif) + 100 + String(pasif) + 100 + RIGHTMODULE + yont2 + String(aktif2) + 100 + String(pasif2) + 100;
}

// SLAVE
#include "WiFi.h"
#define MODULE_ID_INDEX 0
#define DIRECTION_INDEX 1
#define AKTIF_VELOCITY_FIRST_INDEX 2
#define AKTIF_VELOCITY_LAST_INDEX 5
#define PASIF_VELOCITY_FIRST_INDEX 5
#define PASIF_VELOCITY_LAST_INDEX 8

const char* ssid = "Mekatronix";
const char* password = "mekatronix";
const char* serverIP = "192.168.1.1"; // Replace with the IP address of the master
String receivedData;
const char FORWARD = 'F';
const char BACKWARD = 'B';
const char STOP = 'S';
const char UP = 'U';
const char DOWN = 'D';
const char LEFTMODULE = 'L';
const char RIGHTMODULE = 'R';
uint32_t aktif_hiz;
uint32_t pasif_hiz;
String hizver;
char direction;
char slaveID;

#define PASIF_A 36
#define PASIF_B 39
#define PASIF_EN 34
#define AKTIF_A 35
#define AKTIF_B 32
#define AKTIF_EN 33
#define PWM_FREQ 5000
#define PWMRES 8

WiFiClient client;

void setup() {
  ledcSetup(0, PWM_FREQ, PWMRES);
  ledcSetup(1, PWM_FREQ, PWMRES);
  ledcAttachPin(PASIF_EN, 0);
  ledcAttachPin(AKTIF_EN, 1);
  pinMode(PASIF_A, OUTPUT);
  pinMode(PASIF_B, OUTPUT);
}
```

Figure 4: Code for System Integration

5. Conclusions

This project has utilized a differential gear box system, enabling a successful theoretical transition from a mobile platform structure to an industrial robotic arm structure through its self-reconfiguration capabilities. Our designs have been created using the Solidworks application and successfully assembled in simulation environment. The next steps for this project will involve optimization of its physical dimensions and achieving autonomy. Once these stages are completed, the project will be able to be used in various fields of activity and can be remotely controlled. Looking at other modular robots, a similar differential system has been used in the Smores-ep, which assists in lifting other modules. Another aspects of future work includes to perform simulations of the entire system in a simulation environment where physics can be applied and rapid prototyping of whole system for hardware verifications.

References

[1] The SPHERES Guest Scientist Program (PDF), Mark O. Hilstad, John P. Enright, and Arthur G. Richards Swati Mohan, 2010-01-05.