Project Report: Control of Mecanum Wheels Robot

Madhupriya, Raj, Nithish

April 6, 2024

1 INTRODUCTION

The project aimed to design and implement a robotic platform capable of omnidirectional movement using Mecanum wheels. Mecanum wheels are unique in that they enable a robot to move in any direction without changing its orientation, making them ideal for applications requiring maneuverable motion in a small space. The objectives of this project included implementing basic movements, advanced maneuvers, integrating sensors for feedback, and enabling remote control via Bluetooth connectivity.

2 MECHANICS OF MECANUM WHEEL

The mechanics of Mecanum wheels provide unique capabilities for omnidirectional movement:

- **Roller Orientation:** Each Mecanum wheel has rollers mounted along its circumference at an angle of approximately 45 degrees to the wheel's axis. These rollers are typically made of rubber or hard plastic and are free to rotate about their own axes.
- **Vectorial Movement of wheel:** When a wheel moves forward or backward, its rotation creates a vectorial force in a particular direction. By varying the speed and direction of rotation of each wheel independently, the resultant vector can be controlled to achieve precise movement in any direction.

  In practice, the movement of the robot can be controlled by combining the motion of individual Mecanum wheels:

  - Running all four wheels in the same direction at the same speed will result in a forward/backward movement, as the longitudinal force vectors add up but the transverse vectors cancel each other out.
  - Running both wheels on one side in one direction while the other side in the opposite direction will result in a stationary rotation of the vehicle, as vectors couple to generate a torque around the central vertical axis of the vehicle.
  - Running the diagonal wheels in one direction while the other diagonal in the opposite direction will result in a sideways movement, as the transverse vectors add up but the longitudinal vectors cancel out.

3 COMPONENTS USED:

The primary components used in this project were:

- **Arduino Mega ADK:** Serving as the brain of the robot, the Arduino Mega provided the necessary computational power and I/O interfaces for controlling motors, sensors, and communication modules.
- **Motor Drivers:** Dual channel MDD10A Motor drivers were employed to regulate the speed and direction of the DC motors that powered the Mecanum wheels.
- **DC Motors model :** 12 Volt DC motors are used to drive the Mecanum wheels.
- **Mecanum Wheels :** Mecanum wheels consist of rollers set at an angle to the rotational axis, allowing for lateral motion without changing orientation.
- **12V 1100mAh LiPo Batteries:** Providing portable power, Lithium-Polymer batteries offer a lightweight and rechargeable esolution for powering the robot.
- **Bluetooth Module HC05:** This module facilitated wireless communication with a smartphone, enabling remote control of the robot.

- **'Arduino Car' App:** This mobile application provides an interface between an Android smartphone and the Bluetooth module.
- **MPU6050:** The Inertial Measurement Unit (IMU) measures acceleration and orientation, providing feedback for more accurate navigation and control.
- **Encoders :** These sensors monitor the rotation of the wheels, enabling precise motion control and odometry.

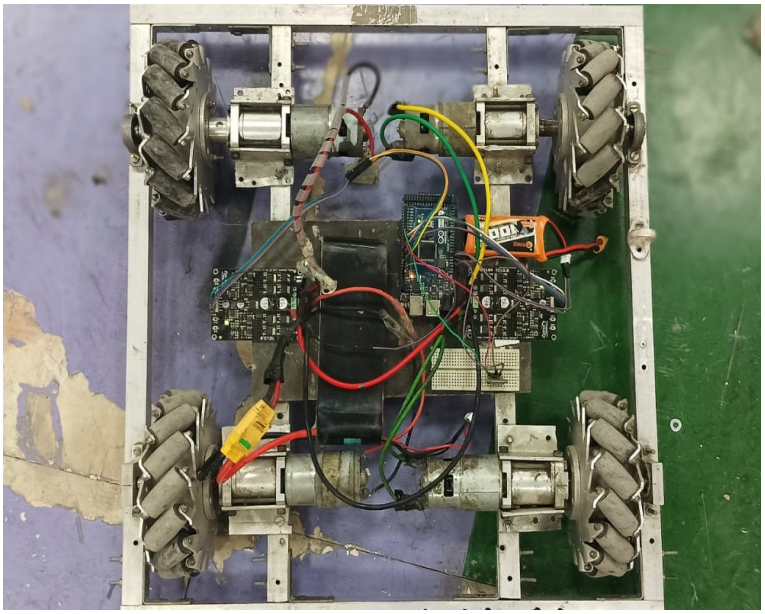

Figure 1: Mecanum wheel Setup

Additionally, a custom PCB was designed in Altium to replace the traditional breadboard setup, to enhance reliability and compactness. However, it was not used in the final project.

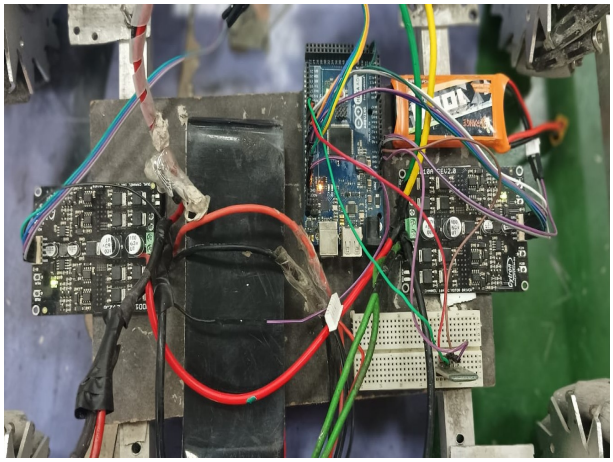

Figure 2: Circuit setup using Breadboard

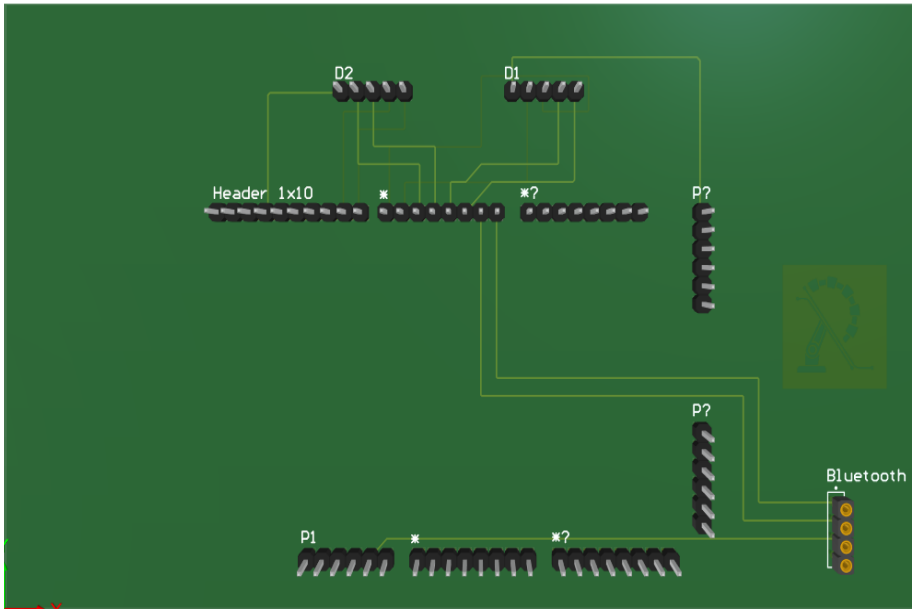

Figure 3: PCB design

4 IMPLEMENTED FUNCTIONS

The implemented functions encompassed a wide range of functionalities:

- **Basic Movements:**
  - `forward()`: Move the robot forward.
  - `backward()`: Move the robot backward.
  - `left()`: Move the robot to the left.
  - `right()`: Move the robot to the right.
- **Rotation:**
  - `rotateL()`: Rotate the robot to the anticlockwise.
  - `rotateR()`: Rotate the robot to the clockwise.
- **Sinusoidal Movement:**
  - `moveSineWave(float amplitude, float freq)`: Move the robot in a sinusoidal pattern with specified amplitude and frequency.
- **Bluetooth Connectivity:**
  - Functions responsible for receiving commands from a smartphone via Bluetooth and translating them into robot movements.

5 CODE

The Arduino code consisted of various functions responsible for interpreting commands received via Bluetooth and controlling the robot's movements accordingly. Functions were implemented to adjust speed, move in different directions, rotate, execute sinusoidal movements, and halt motion. Additionally, code snippets for testing IMU and encoders were developed separately, although full integration into the robot's control system was pending.

5.1 Code- Main

```
#define MotorFR 2
#define SpeedFR 3

#define MotorFL 4
#define SpeedFL 5

#define MotorBR 7
#define SpeedBR 6

#define MotorBL 8
#define SpeedBL 9

#include <math.h>

float rad;
float maxV;
float v1;
float v2;

// Adjust for desired maximum speed
float maxSpeed = 75;
float frequency = 0.20;

int speed=35;
int count =0;

void setup(){
  pinMode(MotorFR, OUTPUT);
  pinMode(SpeedFR, OUTPUT);

  pinMode(MotorFL, OUTPUT);
  pinMode(SpeedFL, OUTPUT);

  pinMode(MotorBR, OUTPUT);
  pinMode(SpeedBR, OUTPUT);

  pinMode(MotorBL, OUTPUT);
  pinMode(SpeedBL, OUTPUT);
  Serial.begin(9600);
}

void loop(){

  if(Serial.available() > 0){
    t = Serial.read();
    Serial.println(t);

    if(t=='J'){
      adjustSpeed();
    }
    if(t=='F'){
      forward();
    }
    if(t=='G'){
      backward();
    }
    if(t=='L'){
      left();
    }
    if(t=='R'){
      right();
    }
    if(t=='X'){
      stop();
```

```
    }
    if(t=='Y'){

    while (t !='X'){
    for (int i = 90; i >=0; i-=10) {
    moveSineWave(i);}
    for (int i = 360; i >=270; i-=10) {
    moveSineWave(i);
    }
    }
    for (int i = 270; i <=360; i+=5) {
    moveSineWave(i);}
    for (int i = 0; i <=90; i+=5) {
    moveSineWave(i);}
    }

    if (t == 'N') {
      if (count == 0) {
        rotateL();
        count = 1;
      }
      else {
        stop();
        count = 0;
      }
    }
    if (t == 'M') {
      if (count == 0) {
        rotateR();
        count = 1;
      }
      else {
        stop();
        count = 0;
      }
    }
  }
  }

// The Back-Left motor is connected opposite to convention, hence the code was modified accordingly

void forward(){
  analogWrite(SpeedFR,speed);
  analogWrite(SpeedFL,speed);
  analogWrite(SpeedBR,speed);
  analogWrite(SpeedBL,speed);

  digitalWrite(MotorFR,LOW);
  digitalWrite(MotorFL,LOW);
  digitalWrite(MotorBR,LOW);
  digitalWrite(MotorBL,HIGH);

  delay(1000);
  }


void backward(){
  analogWrite(SpeedFR,speed);
  analogWrite(SpeedFL,speed);
  analogWrite(SpeedBR,speed);
  analogWrite(SpeedBL,speed);

  digitalWrite(MotorFR,HIGH);
  digitalWrite(MotorFL,HIGH);
  digitalWrite(MotorBR,HIGH);
  digitalWrite(MotorBL,LOW);
```

```
  delay(1000);
  }


void left(){
  analogWrite(SpeedFR,speed);
  analogWrite(SpeedFL,speed);
  analogWrite(SpeedBR,speed);
  analogWrite(SpeedBL,speed);

  digitalWrite(MotorFR,LOW);
  digitalWrite(MotorFL,HIGH);
  digitalWrite(MotorBR,HIGH);
  digitalWrite(MotorBL,HIGH);

  delay(1000);
  }


void right(){
  analogWrite(SpeedFR,speed);
  analogWrite(SpeedFL,speed);
  analogWrite(SpeedBR,speed);
  analogWrite(SpeedBL,speed);

  digitalWrite(MotorFR,HIGH);
  digitalWrite(MotorFL,LOW);
  digitalWrite(MotorBR,LOW);
  digitalWrite(MotorBL,LOW);

  delay(1000);
  }


void rotateR(){
  analogWrite(SpeedFR,speed);
  analogWrite(SpeedFL,speed);
  analogWrite(SpeedBR,speed);
  analogWrite(SpeedBL,speed);

  digitalWrite(MotorFR,HIGH);
  digitalWrite(MotorFL,LOW);
  digitalWrite(MotorBR,HIGH);
  digitalWrite(MotorBL,HIGH);

  delay(200);
  }


void rotateL(){
  analogWrite(SpeedFR,speed);
  analogWrite(SpeedFL,speed);
  analogWrite(SpeedBR,speed);
  analogWrite(SpeedBL,speed);

  digitalWrite(MotorFR,LOW);
  digitalWrite(MotorFL,HIGH);
  digitalWrite(MotorBR,LOW);
  digitalWrite(MotorBL,LOW);
  }


void stop(){
  analogWrite(SpeedFR,0);
  analogWrite(SpeedFL,0);
```

```
  analogWrite(SpeedBR,0);
  analogWrite(SpeedBL,0);
}



void anyangle(float angle){

  rad = angle * M_PI / 180.0;
  maxV= 75;
  v1 = maxV *(sin(rad)+cos(rad))/2.0;
  v2 = maxV *(sin(rad)-cos(rad))/2.0;

  analogWrite(SpeedFR,abs(v2));
  analogWrite(SpeedFL,abs(v1));
  analogWrite(SpeedBR,abs(v1));
  analogWrite(SpeedBL,abs(v2));


  //to determine direction/ quadrant corresponding to angle

  if ((angle>0.0 and angle < 45.0) or (angle>315.0 and angle<360.0)){
  digitalWrite(MotorFR,HIGH);
  digitalWrite(MotorFL,LOW);
  digitalWrite(MotorBR,LOW);
  digitalWrite(MotorBL,LOW);
  }

  if (angle > 45.0 and angle <135.0){
  digitalWrite(MotorFR,LOW);
  digitalWrite(MotorFL,LOW);
  digitalWrite(MotorBR,LOW);
  digitalWrite(MotorBL,HIGH);
  }


  if (angle > 135.0 and angle <225.0){
  digitalWrite(MotorFR,LOW);
  digitalWrite(MotorFL,HIGH);
  digitalWrite(MotorBR,HIGH);
  digitalWrite(MotorBL,HIGH);
  }

  if (angle > 225.0 and angle <315.0){
  digitalWrite(MotorFR,HIGH);
  digitalWrite(MotorFL,HIGH);
  digitalWrite(MotorBR,HIGH);
  digitalWrite(MotorBL,LOW);
  }

  }

  void moveSineWave(float amplitude, float freq) {
  unsigned long currentTime = millis();
  float angle = currentTime * 2 * PI * freq / 1000.0; // Convert milliseconds to seconds

  // Calculate sine wave components for each wheel (considering mechanum wheel directions)
  float v1 = amplitude * (sin(angle) + cos(angle)) / 2.0;
  float v2 = amplitude * (sin(angle) - cos(angle)) / 2.0;

  // Set motor speeds and directions based on calculated components
  analogWrite(SpeedFR, abs(v2));
  analogWrite(SpeedFL, abs(v1));
  analogWrite(SpeedBR, abs(v1));
  analogWrite(SpeedBL, abs(v2));
```

```
  // Determine direction of rotation of wheel based on sign of velocity
  digitalWrite(MotorFR, v2 > 0 ? LOW : HIGH);
  digitalWrite(MotorFL, v1 > 0 ? LOW : HIGH);
  digitalWrite(MotorBR, v1 > 0 ? HIGH : LOW);
  digitalWrite(MotorBL, v2 > 0 ? LOW : HIGH);
  delay(500);
}
```

5.2 Code -IMU

This code reads accelerometer data from the IMU sensor and output the acceleration values along the X, Y, and Z axes. It can be used in combination with Encoder to implement PID control.

```
// Code to measure initial Acceleration in three dimenstions using IMU

#include <Wire.h>

const int MPU_addr = 0x68;

float accel_x_g, accel_y_g, accel_z_g;

void setup() {
  Wire.begin();
  Serial.begin(9600);

  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
}

void loop() {
  // Read accelerometer data
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true);
  int16_t accel_x_raw = Wire.read() << 8 | Wire.read();
  int16_t accel_y_raw = Wire.read() << 8 | Wire.read();
  int16_t accel_z_raw = Wire.read() << 8 | Wire.read();

  // Convert accelerometer data to acceleration in g
  const float sensitivity = 16384.0; // Sensitivity scale factor for 2g full scale
  accel_x_g = accel_x_raw / sensitivity;
  accel_y_g = accel_y_raw / sensitivity;
  accel_z_g = accel_z_raw / sensitivity;

  Serial.print("Accel X (g): "); Serial.print(accel_x_g);
  Serial.print(", Y (g): "); Serial.print(accel_y_g);
  Serial.print(", Z (g): "); Serial.println(accel_z_g);

  delay(1000);
}
```

5.3 Code - Encoder

The encoder code is designed to monitor the rotation of a motor shaft using an encoder sensor.

```
 //Code for measuring rotation of motor using Encoder

 #define outputA 6
 #define outputB 7

 int counter = 0;
```

```
int aState;
int aLastState;

void setup() {
  pinMode (outputA,INPUT);
  pinMode (outputB,INPUT);

  Serial.begin (9600);
  // Reads the initial state of the outputA
  aLastState = digitalRead(outputA);
}

void loop() {
 // Reads the current state of the outputA
  aState = digitalRead(outputA);

  // If the previous and the current state of the outputA are different, that means a Pulse has
      occured
  if (aState != aLastState){

    // If the outputB state is different to the outputA state, that means the encoder is rotating
        clockwise
    if (digitalRead(outputB) != aState) {
      counter ++;
    } else {
      counter --;
    }
    Serial.print("Position: ");
    Serial.println(counter);
  }
  aLastState = aState;
}
```

6 PROBLEMS ENCOUNTERED AND RESOLUTION

6.1 Problems

This project was not without setbacks. We faced several issues throughout the duration we worked on the robot. The major obstructions we encountered are:

- **Execution of SineWave motion:** To make the robot move in a sinusoidal path, precise control of the motor and wheels is required, along with accurate information of the spacial location of the robot. However, since we had not integrated IMU and Encoder with the system, obtaining the necessary information was not possible.
- **Enabling speed control during motion:** Currently, the speed of the robot has to be specified in the code itself, and cannot be changed using the Arduino car App while the robot moves. This problem arises due to an unidentified error in Serial communication.
- **Error in Mecanum wheel setup:** The Back-Right wheel of the setup was lagging as compared to the other three, which resulted in significant inaccuracies with the given expected output.

6.2 Solutions

To resolve these issues, we took the following steps:

- **Execution of SineWave motion:** We defined a function to move the robot at any angle with respect to X-axis. By running this function in a loop with varying angles from 90 ° to 0 °, and then from 360 ° to 270 °, and so on. This roughly mimics a sine wave.
- **Enabling speed control during motion:** Unfortunately, we were not able to resolve this issue. This function needs to be added in the future.
- **Error in Mecanum wheel setup:** The faulty wheel was dissembled. The lag had occurred due to a loose Allen key, which when replaced, functioned properly.

7 FUTURE WORK

- **Replacing breadboard setup with PCB:** This would make the robot more compact and aesthetic.
- **Enabling PID control:** The IMU and Encoders need to be attached to the system, which would make it possible to implement PID control for more smooth functioning.
- **Enabling speed control via Bluetooth during motion:**

8 CONCLUSION

The project successfully realized its objectives of creating a versatile robot capable of omnidirectional movement, advanced manoeuvres, and remote control via Bluetooth connectivity. Integration of additional sensors such as IMU and encoders would enable PID control, enabling more precise navigation and control. Overall, the project exemplified the potential of Arduino-based systems in developing agile and adaptable robotic platforms suitable for a wide range of applications.

9 REPOSITORY

`https://github.com/nrk-necro/Control-of-Mecanum-wheels.git`