Author’s Names

[Email address]

Abstract

The DDMR (Differential-Drive Mobile Robot) is a robot with two independently driven wheels. The DDMR robot has three ultrasonic sensors and an IMU sensor on the top. A preliminary analysis of the obtained DDMR robot is required in order to recognize the provided DDMR robot in order to create a program for monitoring the robot's work in real time. Therefore, a GUI is prepared to monitor the DDMR robot in terms of control of the target position desired by the user and detection of obstacles with the available ultrasonic sensors.

[Ddmr robot]

Moch. Abdul Aziz (4121600031)

M. Naufal Raihan R (4121600036)

Final System and Team Personnel (Insert Pictures)

Mechatronics System Design Journal.

A Technician’s Journal is a short and concise summary of the team’s journey from the initial task analysis through the final design solution.

The documentation should include enough detail for another person to look at your notebook and be able to build your system, or to at least follow the steps your team took to get to your final design solution.

Table of Contents

[1 Introduction and Initial Analysis 2](#_Toc149728741)

[1.1 Project Context 2](#_Toc149728742)

[1.2 Initial Thought Process 2](#_Toc149728743)

[2 Requirement Analysis and Specification 2](#_Toc149728744)

[2.1 User Requirements 2](#_Toc149728745)

[2.2 System Requirements 2](#_Toc149728746)

[2.3 Tools and Technologies 2](#_Toc149728747)

[3 Conceptual Design 2](#_Toc149728748)

[3.1 System Architecture 2](#_Toc149728749)

[3.2 Interface Design 2](#_Toc149728750)

[3.3 Control Algorithm Design 3](#_Toc149728751)

[4 Detailed Design and Development 3](#_Toc149728752)

[4.1 Component Design 3](#_Toc149728753)

[4.2 Coding and Implementation 3](#_Toc149728754)

[4.3 Integration 3](#_Toc149728755)

[4.4 Unique Features 3](#_Toc149728756)

[5 Testing, Evaluation, and Optimization 3](#_Toc149728757)

[5.1 Testing Strategy 3](#_Toc149728758)

[5.2 Performance Evaluation 3](#_Toc149728759)

[5.3 Optimization 3](#_Toc149728760)

[6 Collaboration and Project Management 3](#_Toc149728761)

[6.1 Teamwork Dynamics 3](#_Toc149728762)

[6.2 Project Management 3](#_Toc149728763)

[7 Conclusion and Reflection 3](#_Toc149728764)

[7.1 Project Summary 3](#_Toc149728765)

[7.2 Future Work 3](#_Toc149728766)

[7.3 Personal and Group Reflections 3](#_Toc149728767)

[8 Appendices 4](#_Toc149728768)

[8.1 Bill of Materials 4](#_Toc149728769)

[8.2 Electrical Wiring and System Layout 4](#_Toc149728770)

[8.3 Code Repository 4](#_Toc149728771)

[8.4 Additional Documentation 4](#_Toc149728772)

[9 References 4](#_Toc149728773)

# 1 Introduction and Initial Analysis

## 1.1 Project Context

## The development of DDMR robots has been the focus of research since decades ago. The challenge in the development of this robot is to create a motion mechanism from two motors (left and right motors) that is efficient and adaptable in its environment. The development of this robot is also related to the use of advanced sensors to improve its functionality because this robot is widely applied in technological developments and industrial needs. One example in our project, this DDMR robot uses three ultrasonic sensors and the use of IMU sensors. Therefore, the development of this project is very necessary in finding new innovations in its utilization.

## The problem that we have to solve is to provide the features of the DDMR robot that have been provided. These features include the target position of the robot movement by providing a user-defined position point, error detection of the target position that has been determined with the actual conditions, and the utilization of three ultrasonic sensors to detect obstacles.

## The purpose of this project is to gain experience in analyzing a robot that is already available, increase knowledge in programming robotic systems, and increase knowledge in GUI design for monitoring a robot. These things are to improve the abilities possessed by students in the field of mechatronics, especially mechatronics system programming.

## 1.2 Initial Thought Process

## The initial identification that we carried out was to identify every component in the DDMR robot. After that, analyze the schematic series of these components and then think about the concept of providing features to the robot that will be carried out. The initial idea of ​​each stage of the program on the DDMR robot is to move the robot with a target position value that has been determined in the program. After that, try to replace the target position value in the form of a value entered by the user. Followed by conducting trials to detect the error value of a target position value with its actual position. After that, access the ultrasonic sensors which are used to detect obstacles to the robot's journey in reaching its target position. A GUI design will also be prepared to access the robot by providing a target position in the Cartesian plane (x and y axes) in millimeters.

## The challenge in this project is to unite all the features in the above stages in one program and link them to the GUI design. This is a challenge in itself because we don't have experience in this matter. The short time involved in working on this project also meant that we had to be able to work effectively and efficiently.

## The existence of a platform that provides tutorials on this matter can be an opportunity for us to solve this challenge. Guidance from lecturers is also an opportunity for us to complete this DDMR robot project. Support from friends also helps us to learn and process together in working on our respective projects.

## 2 Requirement Analysis and Specification

## 2.1 User Requirements

## The DDMR robot will be operated by students as a learning module in the mechatronics system programming course. The GUI created will display information on data in real time. The data information displayed are robot position, user-defined target position, robot position error value, and ultrasonic sensors. The data displayed can be used by users to monitor and provide position targets for DDMR robots in millimeters.

## 2.2 System Requirements

In the GUI design, there will be a cartesian coordinate (x and y axis) in millimeters that displays the initial position of the robot (at point 0). After that, the user can provide a target position point by clicking a position on the cartesian coordinates. When the user provides the target position, the DDMR robot will go to the target position with a motor speed that can also be adjusted by the user. There is monitoring of the ideal and actual position errors of the DDMR robot which will be displayed on the GUI design.

If there are objects that block the robot lane in reaching the target position, the DDMR robot will stop. This can be monitored by displaying a distance data on the GUI design generated from the three ultrasonic sensors.

## 2.3 Tools and Technologies

## The list of software used, namely:

## a. Arduino IDE

## b. QT Designer

## List of hardware available on the DDMR robot, namely:

## a. Arduino Mega

## b. 3 HC-SR04 ultrasonic sensors

## c. 2 pieces of DC motor JGB37-520 12V

## d. Raspberry PI 3 model B V1.2

## e. MPU6050

## f. 2 pieces of XL4005

## g. Lipo Battery 11.1V 2200mah

## 2.4 Target specification

Tabel 1. Tabel caption.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Description | Measurement Metric | Target Value |
| Target position | The robot will reach a user-defined target in the form of a point on the cartesian coordinates in the GUI design. | Cartesian coordinates | Millimeters |
| Position error detection | The GUI design will display a target data of the ideal and actual position of the robot. | Scale | Milimeters |
| Avoiding obstacles | The robot will stop when the ultrasonic sensor detects an object near the robot | Scale | Centimeter |

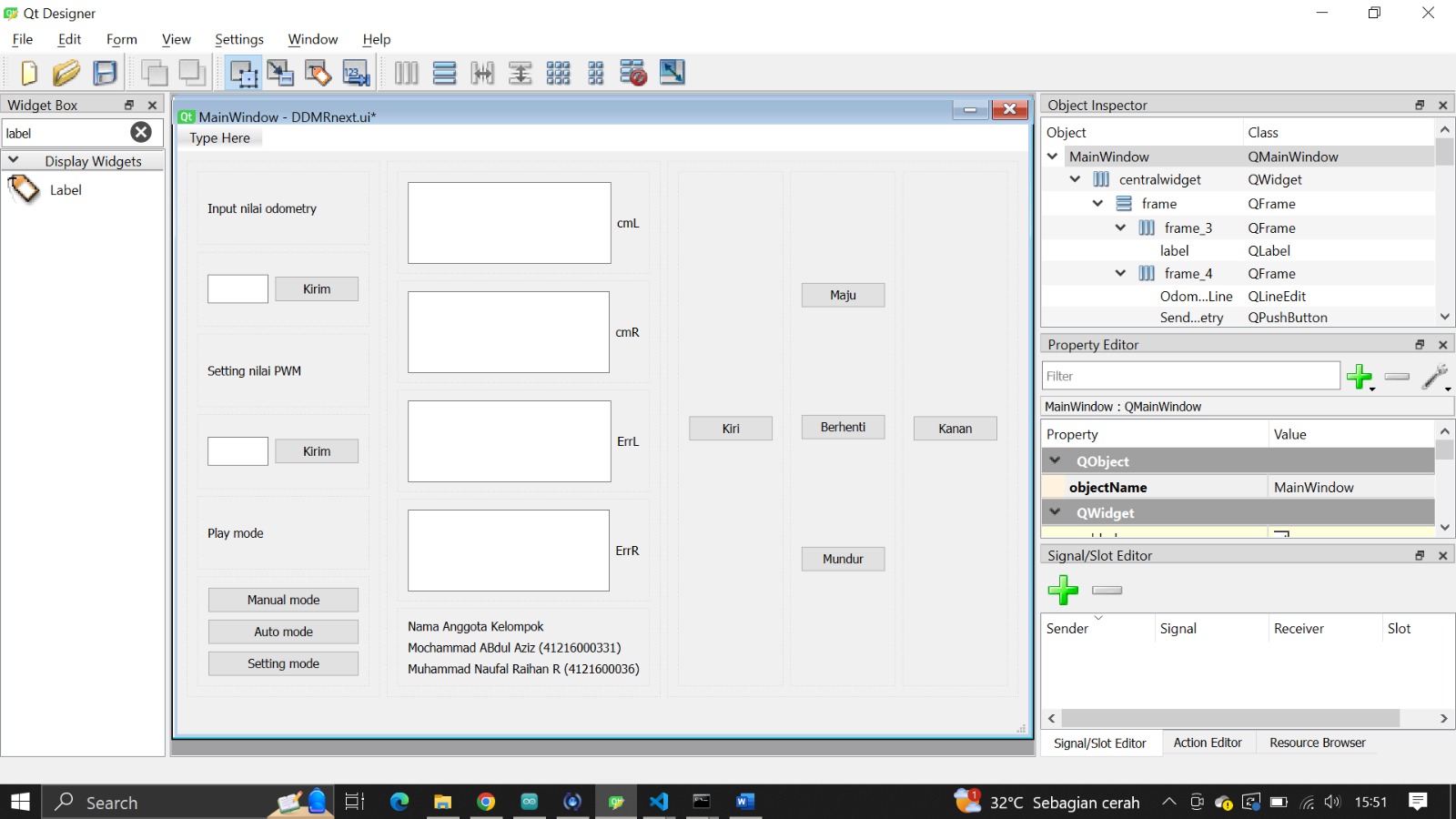
# 3 Conceptual Design

## 3.1 System Architecture

The GUI system will be designed with a display of x and y axis cartesian coordinates with millimeter units, robot on and off buttons, and several displays of robot monitor results such as ultrasonic sensors, IMU sensors, and motor controls. When the robot is in the on position, the position of the robot at that time will be coordinate 0 in the control system and GUI. If the user commands by giving coordinates on the cartesian coordinates, the robot will move towards the coordinate point. Coordinate commands can be given sequentially/more than one command. The ultrasonic sensor will work in a way that the robot will stop if it detects an obstacle that is in front in reaching its target position. This can be monitored with the display provided in the GUI design provided.

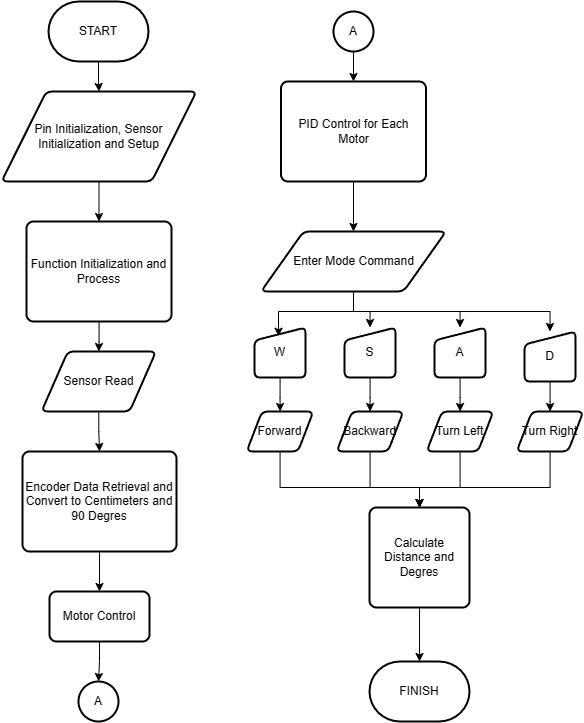
## 3.2 Interface Design

GUI Interface for DDMR Monitoring :



## 3.3 Control Algorithm Design

Flow Chart for Algortithm Design :



# 4 Detailed Design and Development

## 4.1 Component Design

The Arduino Mega functions as a motor control and movement of the DDMR robot, collecting data from sensors that are interpreted for understanding the environment around the robot, communication with the DDMR robot regarding position, speed, etc. information.

Ultrasonic sensors are used as detection and measurement of the distance from the robot to the obstacles in achieving its positioning goals. If the ultrasonic sensor detects an obstacle in front of it with a close distance, the robot will send data to stop the robot.

The DC motor is used as the main driver of the DDMR robot according to the commands given by the program. The motor encoder is also used in determining the distance of the robot's position to its initial position in performing a movement. The motor encoder will be converted into centimeter units in determining its position.

The MPU 6050 functions in performing odometry on the DDMR robot by obtaining the yaw, pitch, roll values of the robot which are useful for determining the position of the robot coordinates. The value that will be used from the acquisition of MPU 6050 data is the yaw value as the x and y angles of the facing angle of the DDMR robot in cartesian coordinates.

## 4.2 Coding and Implementation

Document the coding of Microcontroller and GUI

DDMR Code with ArduinoIDE :

|  |
| --- |
| #include <Arduino.h>  #include <MPU6050\_6Axis\_MotionApps20.h>  #include <I2Cdev.h>  #include <Wire.h>    #if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE  #endif  MPU6050 mpu;  #define OUTPUT\_READABLE\_YAWPITCHROLL  #define INTERRUPT\_PIN 20  //#include <SoftwareSerial.h>  //SoftwareSerial bluetooth(1,0);  #define INA1 9  #define INA2 10  #define INB1 8  #define INB2 7  #define PWMA 11  #define PWMB 6  #define ENCA1 2  #define ENCA2 4  #define ENCB1 3  #define ENCB2 5  #define TRIG\_PIN1 36 // Pin trigger sensor ultrasonik  #define ECHO\_PIN1 34 // Pin echo sensor ultrasonik  #define JARAK\_MAKSIMAL 20  unsigned char x,rpwm,lpwm;  int Rotary\_L,Rotary\_R,counter\_L=0,counter\_R=0;  int Error\_L,iError\_L,dError\_L,ErroSebelumnya\_L,K\_L,KD\_L,KI\_L,Target\_L,U\_L;  int Error\_R,iError\_R,dError\_R,ErroSebelumnya\_R,K\_R,KD\_R,KI\_R,Target\_R,U\_R;  int pwm;    // mengatur kecepatan motor 0 - 255  int speed;  int pulseL;  int pulseR;  int jarakcm;  float CML;  float CMR;  float cmL;  float cmR;  float errorcmL;  float errorcmR;  // char modeSetting = 0;  int modeSetting = 0;  int modeKontrol = 0;  int modeOtomatis = 0;  char currentmode;  char SETTING;  char OTOMATIS;  char MANUAL;  float x\_L,x\_R;  float bacaSensorUltrasonik(int trigPin1, int echoPin1) {    digitalWrite(trigPin1, LOW);    delayMicroseconds(2);    digitalWrite(trigPin1, HIGH);    delayMicroseconds(10);    digitalWrite(trigPin1, LOW);    long durasi = pulseIn(echoPin1, HIGH);    float jarak\_cm = durasi \* 0.034 / 2;    return jarak\_cm;  }  void EnkoderA(void){    //motor kanan    if(digitalRead(ENCA1)){      if(digitalRead(ENCA2)) counter\_L++;      else                   counter\_L--;    }    else{      if(digitalRead(ENCA2)) counter\_L--;      else                   counter\_L++;    }  }  void EnkoderB(void){    //motor kiri    if(digitalRead(ENCB1)){      if(digitalRead(ENCB2)) counter\_R--;      else                   counter\_R++;    }    else{      if(digitalRead(ENCB2)) counter\_R++;      else                   counter\_R--;    }  }  void setup() {    Serial.begin(9600);    //Wire.begin();    //mpu.initialize();    pinMode(INA1, OUTPUT);    pinMode(INA2, OUTPUT);    pinMode(INB1, OUTPUT);    pinMode(INB2, OUTPUT);    pinMode(PWMA, OUTPUT);    pinMode(PWMB, OUTPUT);    pinMode(ENCA1, INPUT);    pinMode(ENCA2, INPUT);    pinMode(ENCB1, INPUT);    pinMode(ENCB2, INPUT);    pinMode(TRIG\_PIN1, OUTPUT);    pinMode(ECHO\_PIN1, INPUT);    attachInterrupt(digitalPinToInterrupt(ENCA1),EnkoderA,CHANGE);    attachInterrupt(digitalPinToInterrupt(ENCB1),EnkoderB,CHANGE);  }  bool cekRintangan() {    float jarak\_cm1 = bacaSensorUltrasonik(TRIG\_PIN1, ECHO\_PIN1);    if (jarak\_cm1 < JARAK\_MAKSIMAL) {      Serial.println("Rintangan terdeteksi!");      return true;    }    return false;  }  void jalan\_maju(){            // PROGRAM JALAN MAJU    digitalWrite(INA1, 0);    digitalWrite(INA2, 1);    digitalWrite(INB1, 0);    digitalWrite(INB2, 1);    analogWrite(PWMA, pwm);    analogWrite(PWMB, pwm);  }  void jalan\_mundur(){          // PROGRAM JALAN MUNDUR    digitalWrite(INA1, 1);    digitalWrite(INA2, 0);    digitalWrite(INB1, 1);    digitalWrite(INB2, 0);    analogWrite(PWMA, pwm);    analogWrite(PWMB, pwm);  }  void belok\_kanan(){           // PROGRAM BELOK KANAN    digitalWrite(INB1, 0);    digitalWrite(INB2, 1);    analogWrite(PWMB, pwm);  }  void belok\_kiri(){            // PROGRAM BELOK KIRI    digitalWrite(INA1, 0);    digitalWrite(INA2, 1);    analogWrite(PWMA, pwm);  }  void berhenti(){              // PROGRAM BERHENTI    digitalWrite(INA1, 0);    digitalWrite(INA2, 0);    digitalWrite(INB1, 0);    digitalWrite(INB2, 0);    analogWrite(PWMA, 0);    analogWrite(PWMB, 0);  }  void siku\_kanan(){    float derajat = 90 \* 10.1;    while(counter\_L <= derajat){      belok\_kanan();      if (counter\_L >= derajat){        berhenti();        break;      }    }  }  void siku\_kiri(){    float derajat = 90 \* 10.1;    while(counter\_R <= derajat){      belok\_kiri();      if (counter\_R >= derajat){        berhenti();      }    }  }  void write\_otomatis(int jarak, char arah){ // ==================== INPUT MODE OTOMATIS ==========================      if(arah == 'w'){      jalan\_maju();      Serial.println("Robot Bergerak Maju");      int startpulseL = pulseL;      int startpulseR = pulseR;      cmL = 0;      cmR = 0;      errorcmL = 0;      errorcmR = 0;      counter\_L = 0;      counter\_R = 0;      while(cmL < jarak || cmR < jarak){        pulseL = counter\_L - startpulseL;        pulseR = counter\_R - startpulseR;        cmL = pulseL / 32.00;        cmR = pulseR / 32.00;        errorcmL = cmL - jarak;        errorcmR = cmR - jarak;        Serial.print("cmL = ");        Serial.print(cmL);        Serial.print("  cmR = ");        Serial.print(cmR);        Serial.print("  ErrL = ");        Serial.print(errorcmL);        Serial.print("  ErrR = ");        Serial.println(errorcmR);        if(cmL >= jarak && cmR >= jarak){          berhenti();          Serial.println("Robot Telah Mencapai Target.");          pulseL = 0; pulseR = 0;          counter\_L = 0;          counter\_R = 0;          break;        }      }      berhenti();    }    else if(arah == 's'){      jalan\_mundur();      Serial.println("Robot Bergerak Mundur");      int startpulseL = pulseL;      int startpulseR = pulseR;      cmL = 0;      cmR = 0;      errorcmL = 0;      errorcmR = 0;      counter\_L = 0;      counter\_R = 0;      while(cmL > -jarak || cmR > -jarak){        pulseL = counter\_L - startpulseL;        pulseR = counter\_R - startpulseR;        cmL = pulseL / 32.00;        cmR = pulseR / 32.00;        errorcmL = cmL - jarak;        errorcmR = cmR - jarak;        Serial.print("cmL = ");        Serial.print(cmL);        Serial.print("  cmR = ");        Serial.print(cmR);        Serial.print("  ErrL = ");        Serial.print(errorcmL);        Serial.print("  ErrR = ");        Serial.println(errorcmR);        if(cmL <= -jarak && cmR <= -jarak){          berhenti();          Serial.println("Robot Telah Mencapai Target.");          pulseL = 0; pulseR = 0;          counter\_L = 0;          counter\_R = 0;          break;        }      }      berhenti();    }    else if(arah == 'd'){      siku\_kanan();      jalan\_maju();      Serial.println("Robot Bergerak Belok Kanan");      int startpulseL = pulseL;      int startpulseR = pulseR;      cmL = 0;      cmR = 0;      errorcmL = 0;      errorcmR = 0;      counter\_L = 0;      counter\_R = 0;      while(cmL < jarak || cmR < jarak){        pulseL = counter\_L - startpulseL;        pulseR = counter\_R - startpulseR;        cmL = pulseL / 32.00;        cmR = pulseR / 32.00;        errorcmL = cmL - jarak;        errorcmR = cmR - jarak;        Serial.print("cmL = ");        Serial.print(cmL);        Serial.print("  cmR = ");        Serial.print(cmR);        Serial.print("  ErrL = ");        Serial.print(errorcmL);        Serial.print("  ErrR = ");        Serial.println(errorcmR);        if(cmL >= jarak && cmR >= jarak){          berhenti();          Serial.println("Robot Telah Mencapai Target.");          pulseL = 0; pulseR = 0;          counter\_L = 0;          counter\_R = 0;          break;        }      }      berhenti();    }    else if(arah == 'a'){      siku\_kiri();      jalan\_maju();      Serial.println("Robot Bergerak Belok Kiri");      int startpulseL = pulseL;      int startpulseR = pulseR;      cmL = 0;      cmR = 0;      errorcmL = 0;      errorcmR = 0;      counter\_L = 0;      counter\_R = 0;      while(cmL < jarak || cmR < jarak){        pulseL = counter\_L - startpulseL;        pulseR = counter\_R - startpulseR;        cmL = pulseL / 32.00;        cmR = pulseR / 32.00;        errorcmL = cmL - jarak;        errorcmR = cmR - jarak;        Serial.print("cmL = ");        Serial.print(cmL);        Serial.print("  cmR = ");        Serial.print(cmR);        Serial.print("  ErrL = ");        Serial.print(errorcmL);        Serial.print("  ErrR = ");        Serial.println(errorcmR);        if(cmL >= jarak && cmR >= jarak){          berhenti();          Serial.println("Robot Telah Mencapai Target.");          pulseL = 0; pulseR = 0;          counter\_L = 0;          counter\_R = 0;          break;        }      }      berhenti();    }    else {      Serial.println("Arah Tidak Valid!");    }  }  void tampilannilai(){    Serial.print(cmL);    Serial.print("  ");    Serial.println(cmR);    delay(200);  }  void loop() {    if (Serial.available()) {      char received\_input = Serial.read();      if (received\_input == 'j' || received\_input == 'J') {        currentmode = SETTING;        Serial.println("Mode SETTING");        while(currentmode == SETTING){          if(Serial.available() > 0){            pwm = Serial.parseInt();            if(pwm != 0 && pwm <= 255){              Serial.print("PWM Di Atur = ");              Serial.println(pwm);              break;            }            else {              Serial.println("NILAI TIDAK VALID! Inputkan Nilai PWM Kembali!");            }          }        }      }      else if (received\_input == 'k' || received\_input == 'K') {        currentmode = MANUAL;        Serial.println("Mode MANUAL");        while (currentmode == MANUAL) {          if (Serial.available()) {            char control = Serial.read();            if (control == 'w' || control == 'W') {              berhenti();              jalan\_maju();            } else if (control == 'a' || control == 'A') {              berhenti();              belok\_kiri();            } else if (control == 'd' || control == 'D') {              berhenti();              belok\_kanan();            } else if (control == 's' || control == 'S') {              berhenti();              jalan\_mundur();            } else if (control == 'q' || control == 'Q') {              berhenti();            } else if (control == 'l' || control == 'L') {              currentmode = OTOMATIS;              berhenti();              break;            } else if (control == 'j' || control == 'J'){              currentmode = SETTING;              berhenti();              break;            }          }        }      }      else if (received\_input == 'l' || received\_input == 'L') {        currentmode = OTOMATIS;        cmL = 0;        cmR = 0;        errorcmL = 0;        errorcmR = 0;        Serial.println("Mode OTOMATIS");        while (cekRintangan()){          berhenti();        }        while(currentmode == OTOMATIS){          if(Serial.available() >= 3){            int received\_jarak = Serial.parseInt();            char received\_arah = Serial.read();            write\_otomatis(received\_jarak, received\_arah);            delay(500);            }        }        }      }    } |

GUI Code for DDMR :

|  |
| --- |
| # -\*- coding: utf-8 -\*-  # Form implementation generated from reading ui file 'DDMRnext.ui'  #  # Created by: PyQt5 UI code generator 5.15.9  #  # WARNING: Any manual changes made to this file will be lost when pyuic5 is  # run again.  Do not edit this file unless you know what you are doing.  from PyQt5 import QtCore, QtGui, QtWidgets  import serial  s = serial.Serial('COM5', 9600, *timeout*=2)  *class* Ui\_MainWindow(object):  *def* setupUi(*self*, *MainWindow*):  *MainWindow*.setObjectName("MainWindow")  *MainWindow*.resize(576, 395)  *self*.centralwidget = QtWidgets.QWidget(*MainWindow*)  *self*.centralwidget.setObjectName("centralwidget")  *self*.horizontalLayout = QtWidgets.QHBoxLayout(*self*.centralwidget)  *self*.horizontalLayout.setObjectName("horizontalLayout")  *self*.frame = QtWidgets.QFrame(*self*.centralwidget)  *self*.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame.setObjectName("frame")  *self*.verticalLayout = QtWidgets.QVBoxLayout(*self*.frame)  *self*.verticalLayout.setObjectName("verticalLayout")  *self*.frame\_3 = QtWidgets.QFrame(*self*.frame)  *self*.frame\_3.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_3.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_3.setObjectName("frame\_3")  *self*.horizontalLayout\_2 = QtWidgets.QHBoxLayout(*self*.frame\_3)  *self*.horizontalLayout\_2.setObjectName("horizontalLayout\_2")  *self*.label = QtWidgets.QLabel(*self*.frame\_3)  *self*.label.setObjectName("label")  *self*.horizontalLayout\_2.addWidget(*self*.label)  *self*.verticalLayout.addWidget(*self*.frame\_3)  *self*.frame\_4 = QtWidgets.QFrame(*self*.frame)  *self*.frame\_4.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_4.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_4.setObjectName("frame\_4")  *self*.horizontalLayout\_3 = QtWidgets.QHBoxLayout(*self*.frame\_4)  *self*.horizontalLayout\_3.setObjectName("horizontalLayout\_3")  *self*.OdometryLine = QtWidgets.QLineEdit(*self*.frame\_4)  *self*.OdometryLine.setObjectName("OdometryLine")  *self*.horizontalLayout\_3.addWidget(*self*.OdometryLine)  *self*.SendOdometry = QtWidgets.QPushButton(*self*.frame\_4)  *self*.SendOdometry.setObjectName("SendOdometry")  *self*.horizontalLayout\_3.addWidget(*self*.SendOdometry)  *self*.verticalLayout.addWidget(*self*.frame\_4)  *self*.frame\_5 = QtWidgets.QFrame(*self*.frame)  *self*.frame\_5.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_5.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_5.setObjectName("frame\_5")  *self*.horizontalLayout\_4 = QtWidgets.QHBoxLayout(*self*.frame\_5)  *self*.horizontalLayout\_4.setObjectName("horizontalLayout\_4")  *self*.label\_2 = QtWidgets.QLabel(*self*.frame\_5)  *self*.label\_2.setObjectName("label\_2")  *self*.horizontalLayout\_4.addWidget(*self*.label\_2)  *self*.verticalLayout.addWidget(*self*.frame\_5)  *self*.frame\_7 = QtWidgets.QFrame(*self*.frame)  *self*.frame\_7.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_7.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_7.setObjectName("frame\_7")  *self*.horizontalLayout\_5 = QtWidgets.QHBoxLayout(*self*.frame\_7)  *self*.horizontalLayout\_5.setObjectName("horizontalLayout\_5")  *self*.PWMLine = QtWidgets.QLineEdit(*self*.frame\_7)  *self*.PWMLine.setObjectName("PWMLine")  *self*.horizontalLayout\_5.addWidget(*self*.PWMLine)  *self*.SendPWM = QtWidgets.QPushButton(*self*.frame\_7)  *self*.SendPWM.setObjectName("SendPWM")  *self*.horizontalLayout\_5.addWidget(*self*.SendPWM)  *self*.verticalLayout.addWidget(*self*.frame\_7)  *self*.frame\_8 = QtWidgets.QFrame(*self*.frame)  *self*.frame\_8.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_8.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_8.setObjectName("frame\_8")  *self*.horizontalLayout\_6 = QtWidgets.QHBoxLayout(*self*.frame\_8)  *self*.horizontalLayout\_6.setObjectName("horizontalLayout\_6")  *self*.label\_3 = QtWidgets.QLabel(*self*.frame\_8)  *self*.label\_3.setObjectName("label\_3")  *self*.horizontalLayout\_6.addWidget(*self*.label\_3)  *self*.verticalLayout.addWidget(*self*.frame\_8)  *self*.frame\_6 = QtWidgets.QFrame(*self*.frame)  *self*.frame\_6.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_6.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_6.setObjectName("frame\_6")  *self*.verticalLayout\_2 = QtWidgets.QVBoxLayout(*self*.frame\_6)  *self*.verticalLayout\_2.setObjectName("verticalLayout\_2")  *self*.pushButton = QtWidgets.QPushButton(*self*.frame\_6)  *self*.pushButton.setObjectName("pushButton")  *self*.verticalLayout\_2.addWidget(*self*.pushButton)  *self*.pushButton\_3 = QtWidgets.QPushButton(*self*.frame\_6)  *self*.pushButton\_3.setObjectName("pushButton\_3")  *self*.verticalLayout\_2.addWidget(*self*.pushButton\_3)  *self*.pushButton\_2 = QtWidgets.QPushButton(*self*.frame\_6)  *self*.pushButton\_2.setObjectName("pushButton\_2")  *self*.verticalLayout\_2.addWidget(*self*.pushButton\_2)  *self*.verticalLayout.addWidget(*self*.frame\_6)  *self*.horizontalLayout.addWidget(*self*.frame)  *self*.frame\_16 = QtWidgets.QFrame(*self*.centralwidget)  *self*.frame\_16.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_16.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_16.setObjectName("frame\_16")  *self*.verticalLayout\_4 = QtWidgets.QVBoxLayout(*self*.frame\_16)  *self*.verticalLayout\_4.setObjectName("verticalLayout\_4")  *self*.frame\_12 = QtWidgets.QFrame(*self*.frame\_16)  *self*.frame\_12.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_12.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_12.setObjectName("frame\_12")  *self*.horizontalLayout\_10 = QtWidgets.QHBoxLayout(*self*.frame\_12)  *self*.horizontalLayout\_10.setObjectName("horizontalLayout\_10")  *self*.textEdit = QtWidgets.QTextEdit(*self*.frame\_12)  *self*.textEdit.setObjectName("textEdit")  *self*.horizontalLayout\_10.addWidget(*self*.textEdit)  *self*.label\_4 = QtWidgets.QLabel(*self*.frame\_12)  *self*.label\_4.setObjectName("label\_4")  *self*.horizontalLayout\_10.addWidget(*self*.label\_4)  *self*.verticalLayout\_4.addWidget(*self*.frame\_12)  *self*.frame\_13 = QtWidgets.QFrame(*self*.frame\_16)  *self*.frame\_13.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_13.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_13.setObjectName("frame\_13")  *self*.horizontalLayout\_11 = QtWidgets.QHBoxLayout(*self*.frame\_13)  *self*.horizontalLayout\_11.setObjectName("horizontalLayout\_11")  *self*.textEdit\_2 = QtWidgets.QTextEdit(*self*.frame\_13)  *self*.textEdit\_2.setObjectName("textEdit\_2")  *self*.horizontalLayout\_11.addWidget(*self*.textEdit\_2)  *self*.label\_5 = QtWidgets.QLabel(*self*.frame\_13)  *self*.label\_5.setObjectName("label\_5")  *self*.horizontalLayout\_11.addWidget(*self*.label\_5)  *self*.verticalLayout\_4.addWidget(*self*.frame\_13)  *self*.frame\_14 = QtWidgets.QFrame(*self*.frame\_16)  *self*.frame\_14.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_14.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_14.setObjectName("frame\_14")  *self*.horizontalLayout\_12 = QtWidgets.QHBoxLayout(*self*.frame\_14)  *self*.horizontalLayout\_12.setObjectName("horizontalLayout\_12")  *self*.textEdit\_3 = QtWidgets.QTextEdit(*self*.frame\_14)  *self*.textEdit\_3.setObjectName("textEdit\_3")  *self*.horizontalLayout\_12.addWidget(*self*.textEdit\_3)  *self*.label\_6 = QtWidgets.QLabel(*self*.frame\_14)  *self*.label\_6.setObjectName("label\_6")  *self*.horizontalLayout\_12.addWidget(*self*.label\_6)  *self*.verticalLayout\_4.addWidget(*self*.frame\_14)  *self*.frame\_15 = QtWidgets.QFrame(*self*.frame\_16)  *self*.frame\_15.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_15.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_15.setObjectName("frame\_15")  *self*.horizontalLayout\_13 = QtWidgets.QHBoxLayout(*self*.frame\_15)  *self*.horizontalLayout\_13.setObjectName("horizontalLayout\_13")  *self*.textEdit\_4 = QtWidgets.QTextEdit(*self*.frame\_15)  *self*.textEdit\_4.setObjectName("textEdit\_4")  *self*.horizontalLayout\_13.addWidget(*self*.textEdit\_4)  *self*.label\_7 = QtWidgets.QLabel(*self*.frame\_15)  *self*.label\_7.setObjectName("label\_7")  *self*.horizontalLayout\_13.addWidget(*self*.label\_7)  *self*.verticalLayout\_4.addWidget(*self*.frame\_15)  *self*.horizontalLayout.addWidget(*self*.frame\_16)  *self*.frame\_2 = QtWidgets.QFrame(*self*.centralwidget)  *self*.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_2.setObjectName("frame\_2")  *self*.horizontalLayout\_7 = QtWidgets.QHBoxLayout(*self*.frame\_2)  *self*.horizontalLayout\_7.setObjectName("horizontalLayout\_7")  *self*.frame\_11 = QtWidgets.QFrame(*self*.frame\_2)  *self*.frame\_11.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_11.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_11.setObjectName("frame\_11")  *self*.horizontalLayout\_8 = QtWidgets.QHBoxLayout(*self*.frame\_11)  *self*.horizontalLayout\_8.setObjectName("horizontalLayout\_8")  *self*.LeftBtn = QtWidgets.QPushButton(*self*.frame\_11)  *self*.LeftBtn.setObjectName("LeftBtn")  *self*.horizontalLayout\_8.addWidget(*self*.LeftBtn)  *self*.horizontalLayout\_7.addWidget(*self*.frame\_11)  *self*.frame\_9 = QtWidgets.QFrame(*self*.frame\_2)  *self*.frame\_9.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_9.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_9.setObjectName("frame\_9")  *self*.verticalLayout\_3 = QtWidgets.QVBoxLayout(*self*.frame\_9)  *self*.verticalLayout\_3.setObjectName("verticalLayout\_3")  *self*.ForwardBtn = QtWidgets.QPushButton(*self*.frame\_9)  *self*.ForwardBtn.setObjectName("ForwardBtn")  *self*.verticalLayout\_3.addWidget(*self*.ForwardBtn)  *self*.StopBtn = QtWidgets.QPushButton(*self*.frame\_9)  *self*.StopBtn.setObjectName("StopBtn")  *self*.verticalLayout\_3.addWidget(*self*.StopBtn)  *self*.BackwardBtn = QtWidgets.QPushButton(*self*.frame\_9)  *self*.BackwardBtn.setObjectName("BackwardBtn")  *self*.verticalLayout\_3.addWidget(*self*.BackwardBtn)  *self*.horizontalLayout\_7.addWidget(*self*.frame\_9)  *self*.frame\_10 = QtWidgets.QFrame(*self*.frame\_2)  *self*.frame\_10.setFrameShape(QtWidgets.QFrame.StyledPanel)  *self*.frame\_10.setFrameShadow(QtWidgets.QFrame.Raised)  *self*.frame\_10.setObjectName("frame\_10")  *self*.horizontalLayout\_9 = QtWidgets.QHBoxLayout(*self*.frame\_10)  *self*.horizontalLayout\_9.setObjectName("horizontalLayout\_9")  *self*.RightBtn = QtWidgets.QPushButton(*self*.frame\_10)  *self*.RightBtn.setObjectName("RightBtn")  *self*.horizontalLayout\_9.addWidget(*self*.RightBtn)  *self*.horizontalLayout\_7.addWidget(*self*.frame\_10)  *self*.horizontalLayout.addWidget(*self*.frame\_2)  *MainWindow*.setCentralWidget(*self*.centralwidget)  *self*.menubar = QtWidgets.QMenuBar(*MainWindow*)  *self*.menubar.setGeometry(QtCore.QRect(0, 0, 576, 18))  *self*.menubar.setObjectName("menubar")  *MainWindow*.setMenuBar(*self*.menubar)  *self*.statusbar = QtWidgets.QStatusBar(*MainWindow*)  *self*.statusbar.setObjectName("statusbar")  *MainWindow*.setStatusBar(*self*.statusbar)  *self*.SendOdometry.clicked.connect(*self*.show\_line)  *self*.SendPWM.clicked.connect(*self*.show\_edittext)  *self*.ForwardBtn.clicked.connect(*self*.gerak\_maju)  *self*.BackwardBtn.clicked.connect(*self*.gerak\_mundur)  *self*.LeftBtn.clicked.connect(*self*.belok\_kiri)  *self*.RightBtn.clicked.connect(*self*.belok\_kanan)  *self*.StopBtn.clicked.connect(*self*.berhenti)  *self*.pushButton.clicked.connect(*self*.control\_mode)  *self*.pushButton\_2.clicked.connect(*self*.control\_setting)  *self*.pushButton\_3.clicked.connect(*self*.otomatis\_mode)  *self*.retranslateUi(*MainWindow*)          QtCore.QMetaObject.connectSlotsByName(*MainWindow*)  *def* show\_line(*self*):          print(*self*.OdometryLine.text())          odometryValue = *self*.OdometryLine.text()          print(odometryValue)          if odometryValue:              print(odometryValue)              s.write(odometryValue.encode())              print(*f*'Odometry Value "{odometryValue} sent to Arduino')          else:              print('Odometry value is empty. Please enter a value.')  *def* show\_edittext(*self*):          print(*self*.PWMLine.text())          PWMValue = *self*.PWMLine.text()          print(PWMValue)          if PWMValue:              print(PWMValue)              s.write(PWMValue.encode())              print(*f*'PWM value "{PWMValue}" sent to Arduino')          else:              print('PWM value is empty. Please enter a value.')  *def* gerak\_maju(*self*):          try:              s.write(*b*'w')              print("maju")          except Exception as e:              print(*f*'Something went wrong: {e}')    *def* gerak\_mundur(*self*):          try:              s.write(*b*'s')              print("mundur")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* belok\_kanan(*self*):          try:              s.write(*b*'d')              print("kanan")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* belok\_kiri(*self*):          try:              s.write(*b*'a')              print("kiri")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* berhenti(*self*):          try:              s.write(*b*'q')              print("stop")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* control\_mode(*self*):          try:              s.write(*b*'k')              print("Control Mode activated")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* control\_setting(*self*):          try:              s.write(*b*'j')              print("Control Setting activated")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* otomatis\_mode(*self*):          try:              s.write(*b*'l')              print("Control otomatis activated")          except Exception as e:              print(*f*'Something went wrong: {e}')  *def* retranslateUi(*self*, *MainWindow*):          \_translate = QtCore.QCoreApplication.translate  *MainWindow*.setWindowTitle(\_translate("MainWindow", "MainWindow"))  *self*.label.setText(\_translate("MainWindow", "Input nilai odometry"))  *self*.SendOdometry.setText(\_translate("MainWindow", "Kirim"))  *self*.label\_2.setText(\_translate("MainWindow", "Setting nilai PWM"))  *self*.SendPWM.setText(\_translate("MainWindow", "Kirim"))  *self*.label\_3.setText(\_translate("MainWindow", "Play mode"))  *self*.pushButton.setText(\_translate("MainWindow", "Manual mode"))  *self*.pushButton\_3.setText(\_translate("MainWindow", "Auto mode"))  *self*.pushButton\_2.setText(\_translate("MainWindow", "Setting mode"))  *self*.label\_4.setText(\_translate("MainWindow", "cmL"))  *self*.label\_5.setText(\_translate("MainWindow", "cmR"))  *self*.label\_6.setText(\_translate("MainWindow", "ErrL"))  *self*.label\_7.setText(\_translate("MainWindow", "ErrR"))  *self*.LeftBtn.setText(\_translate("MainWindow", "Kiri"))  *self*.ForwardBtn.setText(\_translate("MainWindow", "Maju"))  *self*.StopBtn.setText(\_translate("MainWindow", "Berhenti"))  *self*.BackwardBtn.setText(\_translate("MainWindow", "Mundur"))  *self*.RightBtn.setText(\_translate("MainWindow", "Kanan"))  if \_\_name\_\_ == "\_\_main\_\_":      import sys      app = QtWidgets.QApplication(sys.argv)      MainWindow = QtWidgets.QMainWindow()      ui = Ui\_MainWindow()      ui.setupUi(MainWindow)      MainWindow.show()      sys.exit(app.exec\_()) |

## 4.3 Integration

Integration GUI and Control :

* Buat antar muka GUI menggunakan platform QT Creator. Dalam pemrograman ini kami menggunakan Bahasa Python.
* Tentukan protocol komunikasi antara GUI dan Arduino. Pastikan untuk memperhatikan baud rate, bit data, bit stop, dan paritas saat mengonfigurasi komunikasi serial.
* Implementasi GUI untuk komunikasi serial
* Uji coba dan debug.

## 4.4 Unique Features

# We haven't added any unique features in this system yet. We still use the system of monitoring wheel mileage error data obtained from the encoder on the GUI. The next step, we will add monitoring and planning of DDMR trajectory.

# 5 Testing, Evaluation, and Optimization

## 5.1 Testing Strategy

**Test Methodology:**

1. Motor Functionality Testing:

* Make sure each motor can move according to the command from the Arduino.
* Test motor revs and speed to ensure consistency and reliability.

1. Encoder Testing:

* Check the encoder readings to ensure that each motor provides correct information about the rotation position and speed.
* Validate the accuracy of the encoder by comparing the measured displacement with the actual displacement.

1. Motor and Encoder Calibration:

* Calibrate each motor and encoder to compensate for differences in hardware characteristics and to improve displacement accuracy.

1. PID (Proportional-Integral-Derivative) Controller Testing:

* Implement a PID controller to control the movement of the robot.
* Adjust PID parameters and test the robot's response to positional commands.

1. Position Traking Testing:

* Give specific position commands to the robot.
* Record encoder data and comparison of actual position with desired position.
* Evaluate the robot's performance in achieving and maintaining the target position.

1. Navigation Testing:

* Conduct tests in different environments to assess the robot's ability to move and adapt to changing conditions.

**Test Cases for Position Validation:**

1. **Static Position Testing:**
   * Determine a target position and check if the robot can reach this position with adequate accuracy.
2. **Dynamic Position Testing:**
   * Test the robot in dynamic movement to see if it can maintain the target position during motion.
3. **Position Testing with Additional Load:**
   * Add a load to the robot and test if it can still achieve and maintain the target position.

**Tools Used:**

1. **Arduino Mega 2560:**
   * As the controller's brain for the robot.
2. **Motors with Encoders:**
   * Controllable motors equipped with encoders to monitor motor position and speed.
3. **Position Sensor (e.g., GPS):**
   * If possible, use a position sensor to validate the robot's absolute position.
4. **Distance Measurement Tool or Measuring Tape:**
   * Used to manually measure the distance traveled by the robot and compare it with encoder readings.
5. **Data Analysis Software:**
   * Use software to analyze encoder data and compare it with target positions.
6. **Testing Arena:**
   * Place the robot in a suitable and safe testing area to conduct a series of tests.
7. **PID Tuning Instrument:**
   * If necessary, use a tool to measure and optimize PID parameters.

# 6 Collaboration and Project Management

## 6.1 Teamwork Dynamics

The DDMR project team consists of two people who work together on the project. One person is in charge of programming the DDMR system according to the predetermined target. The other person is in charge of creating the GUI design and connecting it with the program that has been made by the DDMR system programmer. This team also helps each other if there is an obstacle or difficulty in doing their respective tasks. Making the contents of the report is done with the collaboration of this team by filling in what each person has done in their duties.

## 6.2 Project Management

The schedule for this DDMR project is carried out once a week in the Mechatronics System Programming course meeting. The work is in terms of programming the DDMR system and the GUI system that has been assigned to each person in the team. Work outside the schedule is carried out in terms of working on reports and GUI designs. The achievements of this team in working on this project, namely having succeeded in providing odometry features on DDMR robots by utilizing motor encoders where users can provide target robot destination positions, have succeeded in making DDMR robot manual controls, successfully connecting robot programs with GUI designs that have been made, and have not succeeded in providing robot systems to avoid obstacles in reaching their targets.

# 7 Conclusion and Reflection

## 7.1 Project Summary

**Key Achievements:**

1. **Functional Motor Operation:**
   * Successful testing confirmed that each motor can operate according to commands from the Arduino.
   * Motor rotation and speed were consistent and reliable.
2. **Encoder Accuracy:**
   * Encoder readings were accurate, providing reliable information about motor position and rotational speed.
   * Calibration improved encoder accuracy, compensating for hardware differences.
3. **PID Controller Implementation:**
   * The PID controller was successfully implemented, enabling precise control of the robot's movement.
   * Fine-tuning PID parameters improved the robot's response to position commands.
4. **Position Tracking Success:**
   * Position tracking tests demonstrated the robot's ability to reach and maintain target positions accurately.
   * Dynamic and static tests validated the robot's performance in different scenarios.
5. **Navigation Capability:**
   * Navigation tests showcased the robot's adaptability to various environments and its ability to navigate through dynamic conditions.

**Key Learnings:**

1. **Motor and Encoder Calibration Importance:**
   * Calibrating motors and encoders is crucial to achieving accurate displacement readings and consistent performance.
2. **PID Controller Optimization:**
   * Understanding and fine-tuning PID parameters is essential for achieving precise and responsive control over the robot's movement.
3. **Dynamic Positioning Challenges:**
   * Testing in dynamic scenarios revealed challenges in maintaining position accuracy during movement, highlighting areas for improvement.

**Key Outcomes:**

1. **Position Validation Success:**
   * The robot successfully validated target positions in both static and dynamic scenarios, meeting the desired accuracy.
2. **Improved Navigation Capability:**
   * The robot demonstrated enhanced navigation capabilities, adapting well to changes in its environment.
3. **Identification of Areas for Enhancement:**
   * Testing identified areas for improvement, such as addressing challenges in dynamic positioning and further optimizing control algorithms.
4. **Documentation and Data Analysis:**
   * Thorough documentation and data analysis were essential in evaluating the robot's performance, providing valuable insights for future development.

## 7.2 Future Work

**Further Enhancements:**

1. **Sensor Fusion:**
   * Integrate additional sensors, such as inertial measurement units (IMUs) or cameras, for sensor fusion. This can enhance the robot's perception and improve its ability to navigate and interact with the environment.
2. **Advanced Control Algorithms:**
   * Explore and implement advanced control algorithms beyond PID, such as Model Predictive Control (MPC) or Reinforcement Learning. These can optimize the robot's performance in various conditions.
3. **Real-time Path Planning:**
   * Develop a real-time path planning algorithm to enable the robot to autonomously plan and adapt its path based on environmental changes or obstacles.
4. **Wireless Communication:**
   * Implement wireless communication modules, such as Bluetooth or Wi-Fi, to enable remote control and monitoring of the robot. This can enhance the robot's flexibility and usability.
5. **Energy Efficiency Optimization:**
   * Research and implement energy-efficient strategies for motor control and movement to extend the robot's operational time between battery charges.

## 7.3 Personal and Group Reflections

**Challenges Faced:**

1. **Motor and Encoder Calibration:**
   * Calibrating the motors and encoders to ensure accurate and consistent readings posed a significant challenge. Achieving precise displacement measurements required careful adjustments and a thorough understanding of the hardware.
2. **PID Tuning:**
   * Implementing and tuning the PID controller to strike the right balance between responsiveness and stability was a challenging aspect. Iterative adjustments were necessary to optimize the robot's response to position commands.
3. **Dynamic Positioning:**
   * Testing the robot's ability to maintain position accuracy during dynamic movements revealed challenges. Fine-tuning the control algorithms for dynamic scenarios was necessary to improve overall performance.

**Acquired Knowledge:**

1. **Motor Control:**
   * Gain a deep understanding of motor control principles, including the significance of calibration for achieving precise and consistent movement.
2. **PID Controller Implementation:**
   * Acquire hands-on experience in implementing and tuning PID controllers, recognizing the importance of parameter adjustments for achieving optimal performance.
3. **Sensor Integration:**
   * Explore the integration of sensors, such as encoders, for accurate feedback and understand the role of sensor data in enhancing the robot's navigation capabilities.
4. **Testing Methodologies:**
   * Learn systematic testing methodologies for validating the robot's functionalities, including static and dynamic positioning tests, and the importance of thorough documentation.
5. **Navigation Challenges:**
   * Understand the challenges associated with dynamic positioning and navigation, leading to insights into areas for improvement in control algorithms and system robustness.

**Lessons Learned:**

1. **Iterative Development:**
   * Recognize the iterative nature of robotics development. Continuous testing, adjustment, and refinement are essential for achieving optimal performance.
2. **Interdisciplinary Knowledge:**
   * Appreciate the interdisciplinary nature of robotics, requiring knowledge in mechanics, electronics, control theory, and software development.
3. **Real-world Application Considerations:**
   * Consider the practical challenges of deploying a robotic system in real-world scenarios, such as environmental variations and unexpected obstacles.
4. **Documentation Importance:**
   * Emphasize the significance of thorough documentation for troubleshooting, knowledge transfer, and future enhancements.

# 8 Appendices

## 8.1 Bill of Materials

|  |  |  |  |
| --- | --- | --- | --- |
| Parts | Sum | Price | Total |
| Arduino MEGA 2560 | 1 | Rp625.000 | Rp625.000 |
| Motor DC with Encoder | 2 | Rp135.000 | Rp270.000 |
| L298N | 1 | Rp15.000 | Rp15.000 |
| Power Supply Board with Lm2596 | 1 | Rp53.000 | Rp53.000 |
| Lipo Battery 12V 2200mAh | 1 | Rp280.000 | Rp280.000 |
| DDMR Wheel | 2 | Rp13.500 | Rp17.000 |
| DDMR Chasis | 1 | Rp50.000 | Rp50.000 |

## 8.2 Electrical Wiring and System Layout

Sebuah gambar berisi teks, diagram, Rencana, skematis

Deskripsi dibuat secara otomatis

## 8.3 Code Repository

Github :

## 8.4 Additional Documentation

Github :