Fatkhur Rohman Fajar Pramuji (4121600043)

Reza Zakaria (4121600048)

Control of Quadcopter

Team Information

Abstract

Teknologi UAV atau drone terus berkembang dalam aplikasi pemantauan. Quadcopter, dengan empat rotor yang manuver, membutuhkan sensor MPU6050 untuk menjaga stabilitas penerbangan. Pentingnya kontrol yang efisien mengundang pengembangan antarmuka pengguna grafis (GUI) intuitif. GUI yang baik memudahkan kontrol dan pemantauan data sensor secara real-time. Kombinasi sensor MPU6050 dan GUI yang efisien berpotensi meningkatkan pengendalian dan pemantauan penerbangan quadcopter. Penelitian ini bertujuan mengembangkan quadcopter menggunakan Arduino Mega, sensor MPU6050, dan GUI yang sederhana. Harapannya, proyek ini memberi kontribusi pada pengembangan quadcopter yang lebih canggih dan ramah pengguna.

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 3](#_Toc153368295)

[1.1 Project Context 3](#_Toc153368296)

[1.2 Initial Thought Process 4](#_Toc153368297)

[2. Requirement Analysis and Specification 5](#_Toc153368298)

[2.1 User Requirements 5](#_Toc153368299)

[2.2 System Requirements 5](#_Toc153368300)

[2.3 Tools and Technologies 6](#_Toc153368301)

[2.4 Target specification 6](#_Toc153368302)

[3. Conceptual Design 6](#_Toc153368303)

[3.1 System Architecture 6](#_Toc153368304)

[3.2 Interface Design 7](#_Toc153368305)

[3.3 Control Algorithm Design 7](#_Toc153368306)

[4. Detailed Design and Development 8](#_Toc153368307)

[4.1 Component Design 8](#_Toc153368308)

[4.2 Coding and Implementation 9](#_Toc153368309)

[4.3 Integration 14](#_Toc153368310)

[4.4 Unique Features 14](#_Toc153368311)

[5. Testing, Evaluation, and Optimization 15](#_Toc153368312)

[5.1 Testing Strategy 15](#_Toc153368313)

[5.2 Performance Evaluation 15](#_Toc153368314)

[5.3 Optimization 15](#_Toc153368315)

[6. Collaboration and Project Management 15](#_Toc153368316)

[6.1 Teamwork Dynamics 15](#_Toc153368317)

[6.2 Project Management 15](#_Toc153368318)

[7. Conclusion and Reflection 16](#_Toc153368319)

[7.1 Project Summary 16](#_Toc153368320)

[7.2 Future Work 16](#_Toc153368321)

[7.3 Personal and Group Reflections 17](#_Toc153368322)

[8. Appendices 17](#_Toc153368323)

[8.1 Bill of Materials 17](#_Toc153368324)

[8.2 Electrical Wiring and System Layout 18](#_Toc153368325)

[8.3 Code Repository 18](#_Toc153368326)

[8.4 Additional Documentation 18](#_Toc153368327)

[9. References 18](#_Toc153368328)

# Introduction and Initial Analysis

## 1.1 Project Context

1. Latar Belakang

Dalam era teknologi modern, pengembangan perangkat keras dan perangkat lunak semakin berkembang pesat, yang menghasilkan berbagai inovasi dalam berbagai bidang, termasuk dalam industri pesawat tak berawak (UAV). UAV atau yang lebih umum dikenal sebagai drone, telah menjadi perangkat yang sangat populer dalam berbagai aplikasi, seperti pemantauan lingkungan, pemetaan, fotografi udara, dan banyak lagi. Salah satu jenis drone yang cukup populer adalah quadcopter, yang dikenal dengan kemampuannya untuk mengangkat diri dan melakukan manuver di udara dengan empat rotor.

Dalam pengembangan quadcopter, salah satu aspek yang penting adalah pengendalian dan stabilisasi selama penerbangan. Untuk mencapai tujuan ini, sensor-sensor yang akurat dan perangkat keras yang andal sangat diperlukan. Sensor MPU6050, yang menggabungkan gyroskop dan akselerometer dalam satu paket, telah terbukti efektif dalam menjaga keseimbangan quadcopter selama penerbangan. Selain itu, pengendalian quadcopter melalui perangkat keras yang dapat diatur dengan mudah dan cepat adalah faktor kunci dalam kesuksesan proyek semacam ini.

Namun, untuk sebagian besar pengguna, pengendalian dan pengawasan quadcopter yang efisien memerlukan antarmuka pengguna yang intuitif dan mudah digunakan. Oleh karena itu, pengembangan sebuah antarmuka pengguna grafis (Graphics User Interface - GUI) yang berbasis perangkat keras dan perangkat lunak sangat relevan. Antarmuka pengguna yang baik akan memungkinkan pengguna untuk mengendalikan quadcopter, mengamati data dari sensor MPU6050, dan memantau kinerja secara real-time dengan cara yang mudah dimengerti.

Melalui kombinasi antara penggunaan sensor MPU6050 untuk menjaga keseimbangan quadcopter dan pengembangan GUI yang efisien, kita dapat mencapai pengendalian quadcopter yang lebih baik dan memungkinkan pemantauan penerbangan yang lebih baik. Oleh karena itu, penelitian ini bertujuan untuk merancang dan membangun sebuah quadcopter yang menggunakan Arduino Mega sebagai otak utama, sensor MPU6050 untuk pengendalian keseimbangan, dan GUI yang memungkinkan pengguna untuk mengontrol dan memantau quadcopter dengan mudah. Dengan demikian, proyek ini memiliki potensi untuk memberikan kontribusi signifikan dalam pengembangan teknologi quadcopter yang lebih canggih dan ramah pengguna.

1. Rumusan Masalah

Rumusan masalah yang akan dibahas pada penelitian ini:

* 1. Bagaimana merancang dan merealisasikan sistem kendali terbang drone quadcopter?
  2. Bagaimana penerapan pemprograman arduino pada drone quadcopter?
  3. Bagaimana pengaturan keseimbangan drone menggunakan sensor MPU6050?
  4. Bagaimana mengembangkan sebuah antarmuka pengguna grafis (GUI) yang dapat memudahkan pengguna dalam mengendalikan quadcopter dan memantau data secara real-time?

1. Tujuan

Tujuan dari penelitian ini adalah:

* 1. Merancang dan merealisasikan sistem kendali terbang drone quadcopter.
  2. Penerapan pemprograman arduino pada drone quadcopter.
  3. Pengaturan keseimbangan drone menggunakan sensor MPU6050.
  4. Mengembangkan Graphic User Interfaces yang memungkinkan untuk dapat mengontrol quadcopter, dan memonitor data seperti ketinggian, orientasi, dan respon system secara real-time

## 1.2 Initial Thought Process

Proses pembuatan quadcopter dimulai dengan tahap brainstorming. Dalam tahap ini, kami mengidentifikasi berbagai konsep dan ide awal terkait desain, penggunaan sensor, dan kontrol quadcopter. Kami mempertimbangkan berbagai fitur yang ingin ditambahkan, seperti kemampuan mengukur ketinggian dan menjaga stabilitas dengan sensor Gyro MPU6050 serta menambahkan GUI sebagai tampilan yang nantinya bisa diguakan untuk monitoring.

Setelah ide-ide teridentifikasi, kami melakukan analisis potensi tantangan dan peluang. Tantangan melibatkan aspek teknis seperti integrasi sensor, pengkodean kontrol, dan pengujian. Peluang mencakup penghematan biaya dan waktu dalam penggunaan quadcopter untuk pengamatan objek. Kami juga mempertimbangkan apakah penggunaan sensor-sensor tertentu dapat meningkatkan kinerja quadcopter dalam berbagai situasi.

Dengan mempertimbangkan hasil brainstorming dan analisis potensi tantangan serta peluang. Kami memutuskan untuk melanjutkan dengan merancang dan membangun quadcopter berbasis Arduino Uno dengan sensor gyroscope meggunakan MPU6050. Keputusan ini didasarkan pada potensi untuk meningkatkan kemampuan stabilitas quadcopter, yang penting dalam pengamatan objek. serta karena ini merupakan modul dari kampus yang sudah disediakan jadi kami hanya menggunakan komponen yang telah disediakan.

Secara keseluruhan, proses pembuatan *quadcopter* melibatkan proses kreatif, analisis yang cermat, dan pengambilan keputusan yang bijaksana. Ini memungkinkan kami untuk mengoptimalkan fitur-fitur quadcopter sesuai dengan tujuannya, sambil mengatasi tantangan teknis yang mungkin muncul dan memanfaatkan peluang yang ada.

# Requirement Analysis and Specification

## 2.1 User Requirements

Antarmuka Pengguna Grafis (GUI) adalah metode interaksi yang sangat efektif dalam sistem atau perangkat lunak karena menggunakan elemen grafis seperti ikon, tombol, dan jendela untuk mempermudah pengguna. Proses menentukan persyaratan GUI melibatkan beberapa langkah kunci: pertama, identifikasi kebutuhan pengguna; kedua, pembuatan wireframe desain; ketiga, pemilihan kontrol dan komponen GUI yang sesuai; keempat, definisi alur kerja; kelima, perhitungan ukuran dan resolusi layar; keenam, mempertimbangkan aspek aksesibilitas; dan terakhir, uji coba prototipe untuk memastikan kesesuaian dan keefektifan antarmuka yang dibuat.

## 2.2 System Requirements

Dalam pengembangan aplikasi atau perangkat lunak, langkah awal yang penting adalah pemilihan sistem kontrol yang tepat dan merancang logika kontrol untuk mengatur respons sistem terhadap input pengguna atau kondisi khusus. Keamanan sistem kontrol harus menjadi prioritas, memastikan keberadaan fitur-fitur yang memadai untuk melindungi data dan fungsi sistem dari akses yang tidak sah. Selain itu, pengujian respons sistem terhadap berbagai input dan kondisi diperlukan untuk menjamin kinerja yang stabil. Untuk mempertahankan kehandalan dan keamanan secara berkelanjutan, pembaruan dan perbaikan sistem kontrol secara berkala menjadi suatu keharusan.

## 2.3 Tools and Technologies

1. Software
   1. Qt Designer
   2. Arduino IDE
2. Hardware
   1. Arduino Mega
   2. MPU6050
   3. Electronic Speed Control (ESC)
   4. Receiver FS-IA6B
   5. Transmitter FS-I6
   6. Motor Brushless (BLDC)
   7. Baterai Lipo 3 cell 11,1v
   8. Telemetri 915 MHz

## 2.4 Target specification

Tabel . Tabel caption.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Description | Measurement Metric | Target Value |
| Example | Detail of the feature | Units/Scale/Methodology | Specified Value |
| Posisi X,Y,Z | Mengetahui posisi roll, pitch, & yaw pada quadcopter | mengakses imu 6 doff 3 axis untuk mengetahui posisi X,Y,Z | ... |

# Conceptual Design

## 3.1 System Architecture

Berikut adalah penjelasan komponen-komponen utama dalam ilustrasi level tinggi ini:

* **GRAPICHAL USER INTERFACE (GUI)**

Pada bagian ini, merupakan bagian yang menghubungkan antara pengguna dan sistem, dimana GUI dikembangkan dengan desain dari QT Designer dan memungkinkan pengguna untuk memberikan input dan menerima data dari sistem. Dalam projek ini, GUI didesain untuk dapat memantau posisi quadcopter dengan membaca data dari IMU, serta dapat memberikan input nilai PID pada system.

* **KOMUNIKASI ANTAR KOMPONEN**

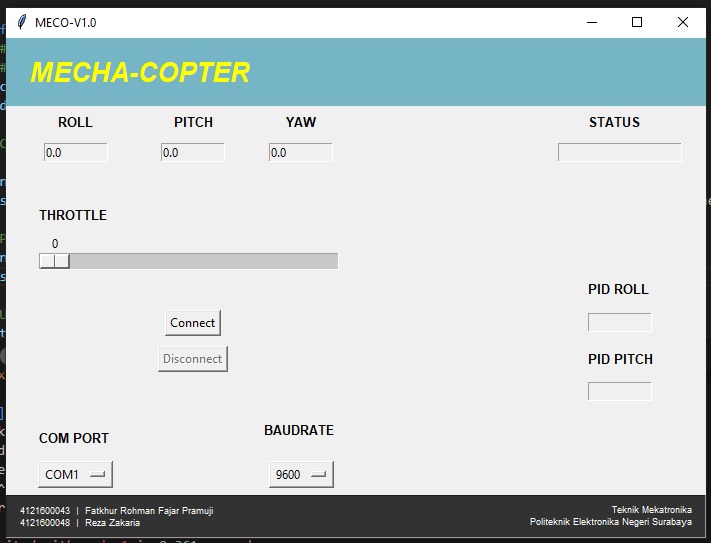
Komunikasi antara Modul Telemetry dan GUI dapat dilakukan melalui protokol komunikasi yang sesuai dengan kebutuhan aplikasi, seperti HTTP/HTTPS, MQTT, WebSocket, RESTful API, dan Serial Communication

* **HARDWARE**

Pada bagian ini, perangkat keras yang digunakan yaitu arduino MEGA, Buck Converter, Telemetry Modul, ESC, Motor Brushless, IMU.

## 3.2 Interface Design

Berikut desain dari GUI yang kami buat



## 3.3 Control Algorithm Design

Alur pemrosesan data dilakukan melalui komunikasi antara GUI dan Hardware. Dalam hal ini pada GUI dapat mengetahui nilai Output dari data pada IMU. Hardware akan mengirim data dimana data tersebut dapat dipantau melalui GUI yaitu data nilai dari IMU berupa pitch, roll, dan yaw.

# Detailed Design and Development

## 4.1 Component Design

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **KOMPONEN** | **FUNGSIONAL** | **Gambar Komponen** |
| 1. | Arduino MEGA | digunakan untuk proyek-proyek yang memerlukan banyak koneksi dan pengendalian yang kompleks. |  |
| 2. | MPU6050 | digunakan untuk mendeteksi gerakan, orientasi, dan percepatan pada suatu objek |  |
| 3. | Brushless motor | Menggerakkan propeller dan memberikan daya angkat pada done |  |
| 4. | ESC | mengatur kecepatan putaran motor brushless (BLDC) pada drone |  |
| 5. | Buck Converter | menurunkan tegangan input menjadi tegangan yang lebih rendah sesuai kebutuhan |  |
| 6. | LiPo Battery | Menyimpan tegangan listrik yang digunakan untukmenggerakkan perangkat elektronik.  Melindungi termal dari kerusakan  Mencegah polusi lingkungan karena kandungan kimia yang ada di dalamnya |  |
| 7 | Receiver | untuk menerima sinyal dari transmitter dan meneruskannya ke mikrokontroler atau sistem yang mengendalikan perangkat. |  |
| 8 | Transmitter | kendali jarak jauh untuk drone |  |
| 9 | Telemetri | digunakan untuk mengirim data dari suatu objek atau sistem ke stasiun kontrol atau pemantauan jarak jauh |  |

## 4.2 Coding and Implementation

// Inisialisasi Library ------------------------------------

#include <Wire.h>

#include <Arduino.h>

#include <MPU6050\_tockn.h>

// Inisialisasi MPU6050 ------------------------------------

TwoWire I2C\_Wire = Wire; // Inisialisasi objek TwoWire

MPU6050 MPU(I2C\_Wire); // Inisialisasi objek MPU6050 menggunakan konstruktor pertama

// Inisialisasi Pin PWM ESC ---------------------------------

#define prop1 3 // 1 3

#define prop2 5 // ^

#define prop3 6 // 4 2

#define prop4 7 // Konfigurasi Propeler (^ = Heading Y)

#define remote 8 // Pin PWM Remot

#define disarm 9 // Pin PWM Switch Remot

// Inisialisasi Variable Data MPU6050 -----------------------

float Gyro\_angle\_x, Gyro\_angle\_y, Gyro\_angle\_z;

int throttle;

// Inisialisaisi Kontrol PID --------------------------------

float setRoll = 0, setPitch = 0, setYaw = 0; // Sudut yang diinginkan

float kp = 0.7; // 0.7

float ki = 0.006; // 0.006

float kd = 1.2; // 1.2

// Inisialisasi Variabel Kontrol PID ------------------------

unsigned long last\_time = 0;

double time\_interval = 0.05; // Waktu interval (ms)

double last\_roll\_error = 0, roll\_P = 0, roll\_I = 0, roll\_D = 0;

double last\_pitch\_error = 0, pitch\_P = 0, pitch\_I = 0, pitch\_D = 0;

float roll\_PID, pitch\_PID;

float roll\_error, pitch\_error;

float roll\_previous\_error = 0, pitch\_previous\_error = 0;

void setup() {

Wire.begin();

Serial.begin(57600);

pinMode(remote, INPUT);

pinMode(disarm, INPUT\_PULLUP);

pinMode(prop1, OUTPUT); // Motor kiri depan

pinMode(prop2, OUTPUT); // Motor kanan belakang

pinMode(prop3, OUTPUT); // Motor kanan depan

pinMode(prop4, OUTPUT); // Motor kiri belakang

MPU.begin(); // Inisialisasi MPU6050

MPU.calcGyroOffsets(true); // Kalibrasi sensor MPU6050, lakukan saat sensor dalam keadaan diam

//delay(2000);

// Membaca data MPU6050

MPU.update();

Gyro\_angle\_x = MPU.getAngleX();

Gyro\_angle\_y = MPU.getAngleY();

Gyro\_angle\_z = MPU.getAngleZ();

/\*

// Setup BLDC

analogWrite(prop1, 150);

analogWrite(prop2, 175);

analogWrite(prop3, 150);

analogWrite(prop4, 150);

Serial.println(" ");

Serial.println("Setup ON");

delay(1000);

analogWrite(prop1, 0);

analogWrite(prop2, 0);

analogWrite(prop3, 0);

analogWrite(prop4, 0);

Serial.println("Setup OFF");

delay(1000);

\*/

}

void loop() {

int ch3 = pulseIn(remote, HIGH);

int read = digitalRead(disarm);

int set\_pwm = 0;

if (read == LOW){

analogWrite(prop1, 0);

analogWrite(prop4, 0);

analogWrite(prop2, 0);

analogWrite(prop3, 0);

Serial.print("Disarm");

}

else if (read == HIGH){

Serial.print("Arm");

if (ch3 < 1100){

analogWrite(prop1, 0);

analogWrite(prop4, 0);

analogWrite(prop2, 0);

analogWrite(prop3, 0);

}

else if (ch3 >= 1100){

//int set\_pwm = 200; // Set Ke 150 - 250;

set\_pwm = map(ch3, 970, 1980, 130, 254);

throttle = map(set\_pwm, 130, 254, 0, 100);

// Membaca data MPU6050 --------------------

MPU.update();

Gyro\_angle\_x = MPU.getAngleX();

Gyro\_angle\_y = MPU.getAngleY();

Gyro\_angle\_z = MPU.getAngleZ();

// Hitung Error ----------------------------

float roll\_er = Gyro\_angle\_x - setRoll;

float pitch\_er = Gyro\_angle\_y - setPitch;

roll\_error = abs(roll\_er);

pitch\_error = abs(pitch\_er);

// Mendefinisikan Waktu --------------------

unsigned long current\_time = millis();

double dt = (current\_time - last\_time) / 1000.0; // Waktu dalam detik

// Perhitungan PID -------------------------

roll\_D = kd \* ((roll\_error - last\_roll\_error) / dt);

roll\_I = ki \* (roll\_I + (roll\_error \* dt));

roll\_P = kp \* roll\_error;

pitch\_D = kd \* ((pitch\_error - last\_pitch\_error) / dt);

pitch\_I = ki \* (pitch\_I + (pitch\_error \* dt));

pitch\_P = kp \* pitch\_error;

float rollPID = roll\_P + roll\_I + roll\_D;

float pitchPID = pitch\_P + pitch\_I + pitch\_D;

roll\_PID = abs(rollPID);

pitch\_PID = abs(pitchPID);

// Menyimpan Data Terakhir -------------------

last\_roll\_error = roll\_error;

last\_pitch\_error = pitch\_error;

last\_time = current\_time;

// Inisialisasi Hasil PID Dengan Sinyal PWM

int ro\_pwm = set\_pwm + roll\_PID;

int pi\_pwm = set\_pwm + pitch\_PID;

int inv\_ro = set\_pwm - roll\_PID;

int inv\_pi = set\_pwm - pitch\_PID;

int roll\_pwm = constrain(ro\_pwm, 150, 254);

int pitch\_pwm = constrain(pi\_pwm, 150, 254);

int inv\_roll\_pwm = constrain(inv\_ro, 150, 254);

int inv\_pitch\_pwm = constrain(inv\_pi, 150, 254);

// Kontrol Roll -------------------------------

if (Gyro\_angle\_x > setRoll) {

analogWrite(prop1, inv\_roll\_pwm);

analogWrite(prop3, inv\_roll\_pwm);

analogWrite(prop2, roll\_pwm);

analogWrite(prop4, roll\_pwm);

}

else if (Gyro\_angle\_x < setRoll) {

analogWrite(prop1, roll\_pwm);

analogWrite(prop3, roll\_pwm);

analogWrite(prop2, inv\_roll\_pwm);

analogWrite(prop4, inv\_roll\_pwm);

}

// Serial.print(" | ");

// Kontrol Pitch ------------------------------

if (Gyro\_angle\_y > setPitch) {

analogWrite(prop1, inv\_pitch\_pwm);

analogWrite(prop4, inv\_pitch\_pwm);

analogWrite(prop2, pitch\_pwm);

analogWrite(prop3, pitch\_pwm);

}

else if (Gyro\_angle\_y < setPitch) {

analogWrite(prop1, pitch\_pwm);

analogWrite(prop4, pitch\_pwm);

analogWrite(prop2, inv\_pitch\_pwm);

analogWrite(prop3, inv\_pitch\_pwm);

/\*

Serial.print(Gyro\_angle\_y);

Serial.print(" ");

Serial.print(pitch\_pwm);

Serial.print(" ");

Serial.print(inv\_pitch\_pwm);

Serial.print(" 1 4 ");

\*/

}

//}

Serial.print("Arm ");

Serial.print(throttle);

Serial.print(" ");

Serial.print(Gyro\_angle\_x);

Serial.print(" ");

Serial.print(Gyro\_angle\_y);

Serial.print(" ");

Serial.print(Gyro\_angle\_z);

Serial.print(" ");

Serial.print(roll\_PID);

Serial.print(" ");

Serial.print(pitch\_PID);

//}

Serial.println(" ");

// Tambahkan delay jika diperlukan untuk kontrol PID yang lebih tepat

delay(40);

}

## 4.3 Integration

Dalam konteks integrasi GUI dengan sistem kontrol, peran GUI sangat signifikan dalam menampilkan dan mengelola data sensor Inertial Measurement Unit (IMU). Selain itu, GUI juga berfungsi sebagai media untuk mengirim parameter PID kepada perangkat keras.

Integrasi ini memungkinkan GUI untuk menyajikan informasi dari IMU secara visual, memberikan pemahaman yang lebih baik terhadap kondisi sistem. Pengguna dapat dengan mudah melihat data yang dihasilkan oleh sensor IMU melalui antarmuka yang intuitif.

Dengan adanya GUI, terjadi hubungan yang sinergis antara pengguna, Telemetry modul sebagai perantara komunikasi, dan perangkat keras dengan sensor-sensor yang beragam. GUI menjadi pusat komando yang memungkinkan pemantauan yang efektif, pengaturan kontrol, dan interaksi dua arah dengan sistem secara real-time.

Dalam keseluruhan integrasi ini, GUI tidak hanya memfasilitasi tampilan data sensor, tetapi juga memberdayakan pengguna untuk secara aktif berpartisipasi dalam pengelolaan dan pengendalian sistem. Memastikan bahwa sistem dapat beroperasi sesuai dengan kebutuhan dan preferensi yang ditetapkan oleh pengguna melalui antarmuka yang ramah pengguna tersebut.

## 4.4 Unique Features

Integrasi pembelajaran mesin dan kecerdasan buatan (AI), di mana algoritma pembelajaran mesin atau kecerdasan buatan (AI) digunakan untuk menyesuaikan dan mengoptimalkan perilakunya berdasarkan lingkungan yang berubah. Hal ini memungkinkan Quadcopter untuk belajar dan meningkatkan kinerjanya seiring berjalannya waktu. Dengan menggabungkan data dari IMU dan Nilai PID pada microcontroller. Pengaplikasian algoritma kontrol PID menjadi teknik kontrol yang tepat, sehingga membantu Quadcopter dalam menyesuaikan kecepatan dan arah motor berdasarkan umpan balik sensor. Kemampuan Quadcopter dalam mendeteksi navigasi otonom juga akan mempermudah sensor dalam mendeteksi rintangan secara real-time.

# Testing, Evaluation, and Optimization

## 5.1 Testing Strategy

Metode yang digunakan pada pengujian ini yaitu metode eksperimen dimana dilakukan trial and error pada program yang diujikan pada quadcopter uji coba.

## 5.2 Performance Evaluation

Untuk performa Quadcopter yang telah di program memiliki beberapa aspek kekurangan sehingga tidak mencapai target. Seperti untuk putaran motor yang tidak stabil . Sehingga quadcopter tidak dapat melakukan take-off secara baik, dan juga untuk imu yang kurang baik dalam pembacaan sehingga membuat komunikasi antara mikro dan esc terdapat bug pada saat melakukan komunikasi serial. Sehingga ada beberapa data yang terlewatkan dan mempersulit proses stabilizer pada quadcopter.

## 5.3 Optimization

Optimasi GUI pada Quadcopter dengan IMU bertujuan untuk meningkatkan pengalaman pengguna, mempermudah pemantauan posisi quadcopter saat di udara.

# Collaboration and Project Management

## 6.1 Teamwork Dynamics

|  |  |  |
| --- | --- | --- |
| **NO.** | **NAMA** | **PERAN** |
| 1. | Fatkhur Rohman Fajar Pramuji  (4121600043) | Akses IMU dan program micro Brushless |
| 2. | Reza Zakaria  (4121600048) | Desain GUI |

## 6.2 Project Management

|  |  |  |
| --- | --- | --- |
| No | Minggu | Pencapaian |
| 1 | Minggu 1 | Dapat mengontrol putaran pada Motor Brushless |
| 2 | Minggu 2 | Dapat mengakses sensor IMU |
| 3 | Minggu 3 | Membuat tampilan desain GUI |
| 4 | Minggu 4 | Mengkombinasikan Posisi IMU pada putaran Brushless |
| 5 | Minggu 5 | Mengkombinasikan Posisi IMU pada putaran Brushless |
| 6 | Minggu 6 | Trial and error |
| 7 | Minggu 7 | Membuat koneksi drone kepada GUI |
| 8 | Minggu 8 | Mengakses quadcopter dengan tampilan pada GUI |

# Conclusion and Reflection

## 7.1 Project Summary

Pencapaian dalam pemrograman drone quadcopter melibatkan kemampuan untuk mengembangkan program atau algoritma drone yang memungkinkan untuk dapat terbang dan mengirim posisi. Berikut adalah ringkasan pencapaian, pembelajaran, dan hasil utama dalam konteks ini:

1. Pencapaian

Menciptakan algoritma yang efektif untuk mengidentifikasi posisi dari sensor IMU secara akurat dan control pada drone menggunakan remote control. Dapat mengimplementasikan program ke dalam drone quadcopter..

1. Pembelajaran

Pemahaman mendalam tentang perangkat yang digunakan dalam drone quadcopter, seperti sensor mpu6050 dan remote control yang digunakan unntuk drone. Pemahaman tentang konsep-konsep dasar pemrograman robotika dan kontrol quadcopter.

1. Hasil Utama

Drone mampu mengirimkan data sensor IMU secara akurat. Drone dapat dikontrol dengan remote control. Drone masih belum mampu terbang karena beberapa perangkat yang kondisinya tidak stabil

## 7.2 Future Work

Drone Quadcopter dapat lebih berkembang pada desain GUI dengan tujuan untuk meningkatkan pengalaman pengguna dan mempermudah pengaturan serta pemantauan robot. Berikut beberapa perkembangan yang mungkin terjadi dalam desain GUI pada quadcopter :

1. Antarmuka Pengguna yang Intuitif

2. Visualisasi Lintasan

3. Pemantauan kecepatan pada setap motor

4. Pengaturan Parameter

5. Pemecahan Masalah

Perkembangan ini bertujuan untuk membuat pengguna drone lebih mudah, efisien, dan responsif, serta meningkatkan kapabilitas pemantauan dan kontrol secara keseluruhan. Dengan desain GUI yang lebih baik, pengguna dapat dengan cepat menyesuaikan drone dengan kebutuhan spesifik.

## 7.3 Personal and Group Reflections

Setelah kami melakukan pengujian terhadap quadcopter kami memiliki pengalaman, tantangan dan pengetahuan yang telah diperoleh, berikut kami tuliskan.

|  |  |  |
| --- | --- | --- |
| No | Nama | Pengalaman, tantangan dan pengetahuan yang diperoleh |
| 1 | Fatkhur Rohman Fajar Pramuji (4121600043) | Terkendala dalam pembuatan GUI dan mengintegrasikannya dengan hardware |
| 2 | Reza Zakaria (4121600048) | Terkendala dalam komponen yang tidak stabil |

# Appendices

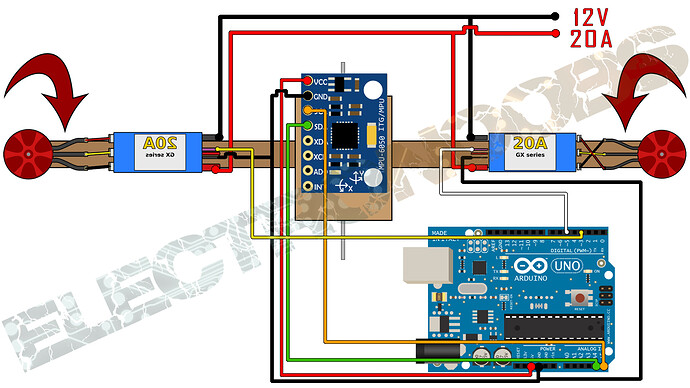
## 8.1 Bill of Materials

Drone Quadcopter ini merupakan drone dari Lab Vehicle PENS (JJ-102) sehingga kelompok kami tidak mengeluarkan biaya. Berikut merupakan rincian harga komponen setelah kami survei dipasaran :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Komponen | Harga satuan | Banyak | Harga total |
| 1 | Arduino Mega | 190.000 | 1 | 190.000 |
| 2 | MPU6050 | 50.000 | 1 | 50.000 |
| 3 | ESC | 180.000 | 4 | 720.000 |
| 4 | Motor BLDC | 156.000 | 4 | 624.000 |
| 5 | Transmitter FS-I6 | 600.000 | 1 | 600.000 |
| 6 | Receiver FS-IA6 | 100.000 | 1 | 100.000 |
| 7 | Baterai Lipo 3s 11,1v | 300.000 | 1 | 300.000 |
| 8 | Buck Converter | 10.000 | 1 | 10.000 |
| Total | | | | 2.594.000 |

## 8.2 Electrical Wiring and System Layout

Skematik dari drone quadcopter :



## 8.3 Code Repository

* Repositori Control Quadcopter :

[Quadcopter-GUI-Control/src/Control/Control\_QC\_K7\_Group2.ino at main · Reza3242/Quadcopter-GUI-Control (github.com)](https://github.com/Reza3242/Quadcopter-GUI-Control/blob/main/src/Control/Control_QC_K7_Group2.ino)

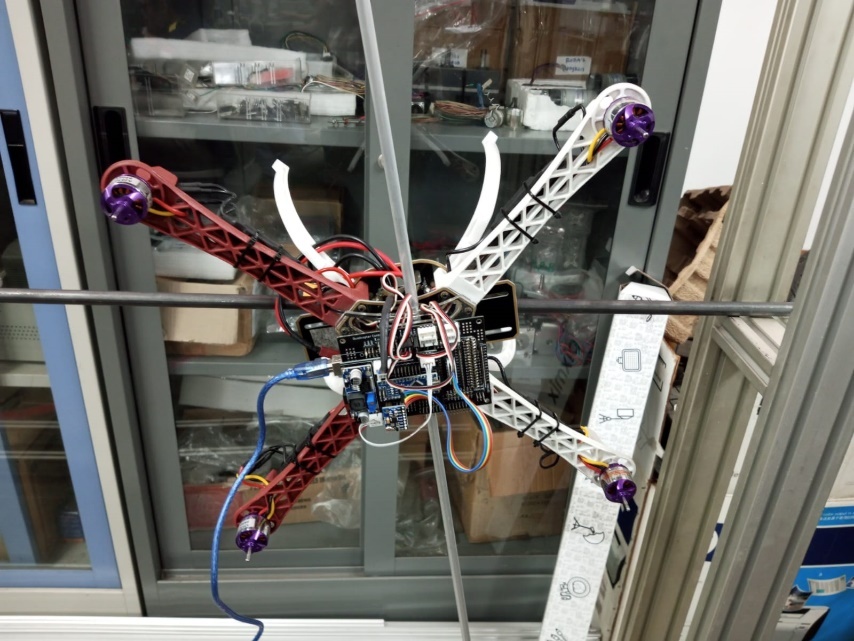
* Repositori GUI Quadcopter :

[Quadcopter-GUI-Control/src/GUI/GUI\_QC\_K7\_Group2.py at main · Reza3242/Quadcopter-GUI-Control (github.com)](https://github.com/Reza3242/Quadcopter-GUI-Control/blob/main/src/GUI/GUI_QC_K7_Group2.py)

## 8.4 Additional Documentation

Dokumentasi dari Drone Quadcopter :





# References

https://chat.openai.com/