Abstract

Perkembangan kendaraan udara tak berawak (UAV) telah berkembang secara signifikan dalam beberapa tahun terakhir, dengan quadcopters muncul sebagai platform yang populer dan serbaguna untuk berbagai aplikasi. Abstrak ini memberikan gambaran komprehensif tentang dinamika dan strategi pengendalian yang digunakan dalam quadcopter, menekankan pentingnya bidang-bidang seperti pengawasan udara, pertanian, pencarian dan penyelamatan, dan hiburan.

pemograman sistem mekatronika

QUADCOPTER

Achmad Bagus Okto Faerizqi (4121600054)

Junio Hangga Wicaksono (4121600046)

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

## Latar Belakang

Drone, khususnya quadcopter, sedang populer di kalangan kaum milenial dengan berbagai latar belakang. Quadcopter merupakan alat yang dapat dikendalikan tanpa awak dan memiliki empat motor penggerak yang berputar dalam arah yang berbeda untuk menjaga keseimbangan. Quadcopter memiliki potensi besar dalam pengamatan objek, mengurangi biaya dan waktu yang diperlukan. Namun, pengendalian dan stabilitas quadcopter masih merupakan tantangan.

Untuk mengatasi masalah ini, penelitian ini memanfaatkan sensor Gyro MPU6050 untuk mengendalikan keseimbangan quadcopter. MPU6050 menggunakan 6 axis motion untuk menjaga quadcopter agar tetap seimbang. Dengan penambahan sensor ini, sistem dapat mengenali keseimbangan quadcopter.

Selain itu, dalam rangka meningkatkan penggunaan quadcopter, juga perlu mempertimbangkan pengembangan GUI (Graphical User Interface) yang dapat digunakan oleh operator. GUI akan mempermudah pengendalian dan pemantauan drone quadcopter, serta memberikan informasi visual yang berguna selama penerbangan. Dengan demikian, penelitian ini bertujuan untuk merancang dan membangun drone quadcopter berbasis Arduino Uno dengan penambahan sensor-sensor yang mencakup ketinggian dan keseimbangan, serta GUI yang intuitif untuk mengoptimalkan penggunaannya.

## Rumusan Masalah

Rumusan masalah yang akan dibahas pada penelitian ini:

1. Bagaimana merancang dan merealisasikan sistem kendali terbang drone quadcopter?

2. Bagaimana penerapan pemrograman arduino pada drone quadcopter?

3. Bagaimana pengaturan keseimbangan drone menggunakan sensor MPU6050?

4. Bagaimana merancang dan mengintegrasikan antarmuka pengguna grafis (GUI) yang efisien dan intuitif untuk mengendalikan dan memantau drone quadcopter?

## 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. Merancang dan mengintegrasikan antarmuka pengguna grafis (GUI) yang efisien dan intuitif untuk mempermudah pengendalian dan pemantauan drone quadcopter.

## 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 dengan sensor BMP280 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 dan me. 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.

# 2 Requirement Analysis and Specification

## 2.1 User Requirements

Antarmuka Pengguna Grafis (GUI) adalah cara yang bagus untuk berinteraksi dengan sistem atau perangkat lunak karena menggunakan elemen grafis seperti ikon, tombol, dan jendela untuk memudahkan pengguna. Untuk menghitung dan menentukan persyaratan GUI, ada beberapa langkah yang bisa diambil:

* Identifikasi Kebutuhan Pengguna
* Desain Wireframe
* Pemilihan Kontrol dan Komponen GUI
* Definisikan Alur Kerja
* Hitung Ukuran dan Resolusi Layar
* Pertimbangkan Aksesibilitas
* Uji Prototipe

Persyaratan pengguna untuk quadcopter dapat bervariasi berdasarkan aplikasi spesifik dan kebutuhan pengguna. Namun, berikut adalah beberapa persyaratan umum pengguna yang biasanya dipertimbangkan ketika merancang atau memilih quadcopter:

* **Performa Penerbangan:**
* **Stabilitas:** Pengguna sering kali memerlukan quadcopter yang menunjukkan karakteristik penerbangan yang stabil untuk memastikan pengoperasian yang lancar dan terkendali**.**
* **Kemampuan Manuver:** Tergantung pada aplikasinya, pengguna mungkin memerlukan quadcopter untuk mampu melakukan manuver yang tepat atau gerakan yang lincah.
* **Jangkauan dan Daya Tahan:**
* **Jangkauan Penerbangan:** Pengguna mungkin memerlukan quadcopter dengan jangkauan yang cukup untuk mencakup area operasi yang diinginkan.
* **Daya Tahan:** Waktu penerbangan yang lebih lama mungkin diperlukan, terutama untuk tugas-tugas seperti survei udara atau pemantauan area yang luas.
* **Komunikasi dan Konektivitas:**
* **Jangkauan dan Keandalan:** Pengguna mungkin memiliki persyaratan khusus mengenai jangkauan komunikasi dan keandalan tautan kontrol antara pengguna dan quadcopter.
* **Transmisi Data:** Untuk aplikasi yang melibatkan pengumpulan data, pengguna mungkin memerlukan kemampuan transmisi data yang andal untuk pemantauan atau analisis waktu nyata.

## 2.2 System Requirements

Dalam mengembangkan aplikasi atau perangkat lunak, langkah awal adalah memilih sistem kontrol yang sesuai dan merancang logika kontrol untuk mengelola respons sistem terhadap input pengguna atau kondisi khusus. Penting untuk memastikan bahwa sistem kontrol dilengkapi dengan fitur keamanan yang memadai guna melindungi data dan fungsi sistem dari akses yang tidak sah. Selain itu, uji respons sistem terhadap berbagai input dan kondisi untuk memastikan kinerja yang stabil. Untuk menjaga kehandalan dan keamanan secara berkelanjutan, lakukan pembaruan dan perbaikan sistem kontrol secara berkala.

## 2.3 Tools and Technologies

Software yang digunakan pada project Pemrograman Sistem Mekatronik yaitu *Visual Studio Code*(Platform IO)/Arduino IDE dan juga menggunakan *QT* untuk mendesain tampilan pada GUI

Untuk komponen hardware yang dugunakan yaitu:

1. Arduino MEGA 2650
2. IMU (MPU6050)
3. Brushless motor
4. ESC *(Electronic System Control)*
5. Buck Converter
6. Battery

## 2.4 Target specification

Tabel 1. Tabel caption.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Description | Measurement Metric | Target Value |
| Posisi X,Y,Z | Mengetahui posisi roll, pitch, & yaw pada quadcopter | mengakses imu 6 doff 3 axis untuk mengetahui posisi X,Y,Z | Quadcopter melakukan take off dan landing secara vertical dan kemudian maju mundur. |

# 3 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**

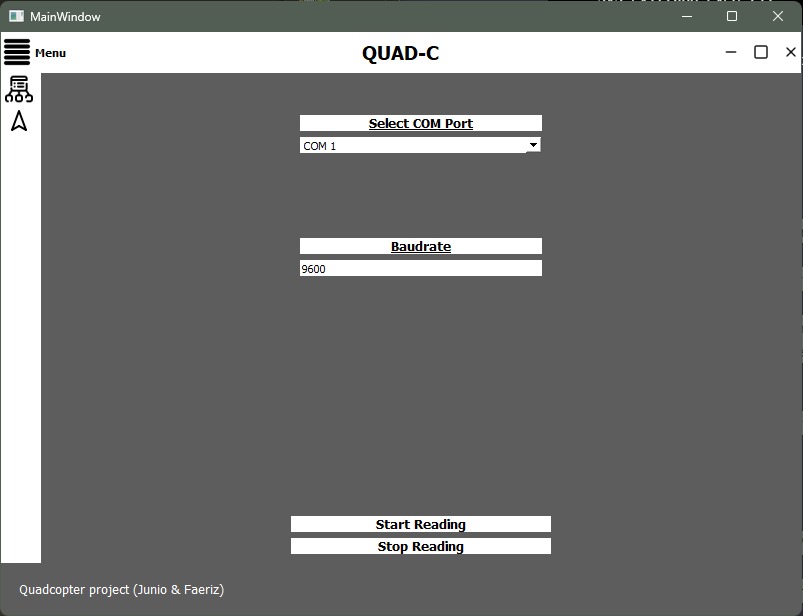
Komunikasi antara Modul Telemetry dan GUI dapat dilakukan melalui protokol komunikasi yang sesuai dengan kebutuhan aplikasi, seperti HTTP/HTTPS, MQTT, WebSocket, RESTful API, dan Serial Communication

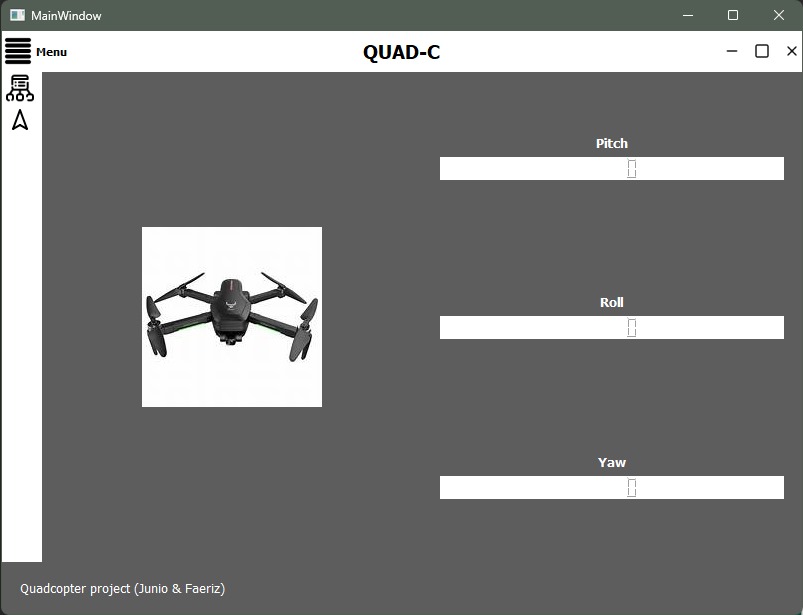
* **HARDWARE**

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

## 3.2 Interface Design

Sketch the preliminary design of the GUI, focusing on user interaction and experience.





## 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.

# 4 Detailed Design and Development

## 4.1 Component Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **KOMPONEN** | **MODUL** | **FUNGSIONAL** | **Gambar Komponen** |
| 1. | Arduino MEGA |  |  |  |
| 2. | IMU |  |  |  |
| 3. | Brushless |  |  |  |
| 4. | ESC |  |  |  |
| 5. | Buck Converter |  |  |  |
| 6. | LiPo Battery | Mencakup fitur-fitur seperti perlindungan baterai (overcharge, over-discharge, over-current), koneksi untuk pengisian, dan mungkin indikator tingkat daya. | Menyimpan tegangan listrik yang digunakan untukmenggerakkan perangkat elektronik.  Melindungi termal dari kerusakan  Mencegah polusi lingkungan karena kandungan kimia yang ada di dalamnya |  |

## 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(5000);

}

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);

/\*

Serial.print(Gyro\_angle\_x);

Serial.print(" ");

Serial.print(roll\_pwm);

Serial.print(" ");

Serial.print(inv\_roll\_pwm);

Serial.print(" 2 4 ");

\*/

}

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(Gyro\_angle\_x);

Serial.print(" ");

Serial.print(roll\_pwm);

Serial.print(" ");

Serial.print(inv\_roll\_pwm);

Serial.print(" 1 3 ");

\*/

}

// 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);

/\*

Serial.print(Gyro\_angle\_x);

Serial.print(" ");

Serial.print(pitch\_pwm);

Serial.print(" ");

Serial.print(inv\_pitch\_pwm);

Serial.print(" 2 3 ");

\*/

}

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(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.

# 5 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

# 6 Collaboration and Project Management

## 6.1 Teamwork Dynamics

Pembagian peran dalam kelompok kami robot line tracer yaitu sebagai berikut:

|  |  |  |
| --- | --- | --- |
| **NO.** | **NAMA** | **PERAN** |
| 1. | Achmad Bagus Okto Faeizqi | Akses IMU dan program micro Brushless |
| 2. | Junio Hangga WIcaksono | Desin 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 |

# 7 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

Quadcopter 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

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

1. Antarmuka Pengguna yang Intuitif

2. Visualisasi lingkungan

3. Pemantauan kecepatan pada setiap motor

4. Pengaturan Parameter

5. Pemecahan Masalah

Perkembangan ini bertujuan untuk membuat pengguna quadcopter 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 robot line tracer kami memiliki pengalaman, tantangan dan pengetahuan yang telah diperoleh, berikut kami tuliskan:

|  |  |  |
| --- | --- | --- |
| No | Nama | Pengalaman, tantangan dan pengetahuan yang diperoleh |
| 1 | Junio Hangga Wicaksono (4121600046) | Terkendala dalam pembuatan GUI dan mengintegrasikannya dengan hardware |
| 2 | Achmad Bagus Okto Faerizqi (4121600054) | Terkendala dalam komponen yang tidak stabil |

# 8 Appendices

## 8.1 Bill of Materials

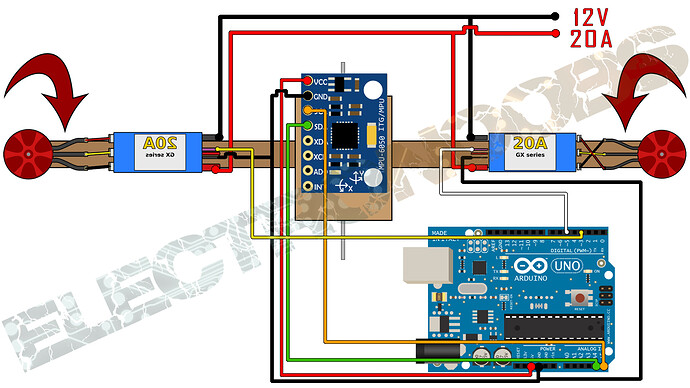
Quadcopter ini merupakan drone dari Lab Vehicle PENS (JJ-102) sehingga kelompok kami tidak mengeluarkan biaya. Berikut merupakan rincian harga komponen setelah kami survei dipasaran :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
| Total | | | | 2.584.000 |

## 

## 8.2 Electrical Wiring and System Layout

Provide diagrams, schematics, and layout information.



## 8.3 Code Repository

Include links to the code repository, version control, and change logs.

## 8.4 Additional Documentation

.

# 9 References

Cite all references, tools, libraries, and external resources used in the project.