Technical Journal

Pendidikan Robotics Software Control
Aksantara ITB 2026

Nama: [**Muhammad Rafiandhi Suryadinata / Reyy**]
NIM: [**19625129**]
Divisi: **Robotics Software Control**

January 30, 2026

Contents

1 Catatan Harian

1.1 DAY 1 - Kamis, 29 Januari 2025

Hari pertama Pendidikan RSC Aksantara 2026 menjadi fondasi awal untuk memahami arah, budaya, serta ruang lingkup teknis yang akan dijalani sebagai Ca-RSC. Materi yang disampaikan tidak hanya bersifat pengenalan, tetapi juga membangun *big picture* mengenai bagaimana peran software bekerja dalam sistem UAV secara menyeluruh.

Pengenalan Departemen Robotics Software Control (RSC)

Pada sesi awal, kami diperkenalkan dengan Departemen Robotics Software Control (RSC), departemen yang didirikan pada tahun 2020 dan bertanggung jawab atas seluruh aspek **perangkat lunak sistem UAV**. Cakupan kerja RSC meliputi pengembangan **algoritma kendali otonom**, sistem persepsi berbasis **computer vision**, hingga perancangan antarmuka **Ground Control Station (GCS)**.

Saat ini, RSC dipimpin oleh **Kak Z. Nayaka Athadiansyah (IF'23)**. Struktur internal RSC terdiri dari tiga divisi utama yaitu **Divisi Manajemen Pengetahuan**, **Divisi Penelitian dan Pengembangan**, dan **Divisi Manajemen Sumber Daya Manusia**. Selain itu, anggota RSC terbagi ke dalam dua jurusan besar yaitu: **Control and Perception** dan **Ground Control Station** yang kemudian dikelompokkan ke dalam tim UAV seperti **VTOL**, **TD**, **FW**, dan **LELA**.

Peran Divisi

- **Divisi Manajemen Pengetahuan**: bertanggung jawab atas penyusunan materi dan sistem penilaian selama pendidikan, serta mengelola **RSCWiki** sebagai repositori dokumentasi teknis.
- **Divisi Penelitian dan Pengembangan**: membantu proses pendidikan dan mengelola riset serta repositori GitHub.
- **Divisi Manajemen Sumber Daya Manusia**: menangani presensi, technical journal, perizinan, serta sistem internal seperti **RSCBoarding** dan **progress report**.

Jurusan dan Fokus Teknis

- **Control and Perception** berfokus pada pengembangan sistem navigasi otonom berbasis **ROS2** dan sistem persepsi menggunakan **computer vision**.
- **Ground Control Station** berfokus pada pengembangan aplikasi GCS, mulai dari **UI/UX**, aplikasi berbasis web, hingga komunikasi menggunakan **MAVLink**.

Gambaran Tim UAV

TD berfokus pada pengembangan GCS dan computer vision, **VTOL** pada sistem otonom dan presisi misi, **LELA** pada deteksi titik api, serta **FW** pada dukungan pemetaan berbasis **GIS**.

Tata Tertib Budaya komunikasi yang santai namun tetap sopan sangat ditekankan, dengan larangan penggunaan frasa militeristik.

Arsitektur Sistem UAV

Pada materi ini, aku mempelajari arsitektur sistem UAV secara menyeluruh, mulai dari komponen fisik hingga logika operasi dan mekanisme keselamatan. Materi ini membantu membangun pemahaman bahwa UAV bukan sekadar alat terbang, melainkan sebuah sistem kompleks yang terdiri dari banyak subsistem yang saling terintegrasi.

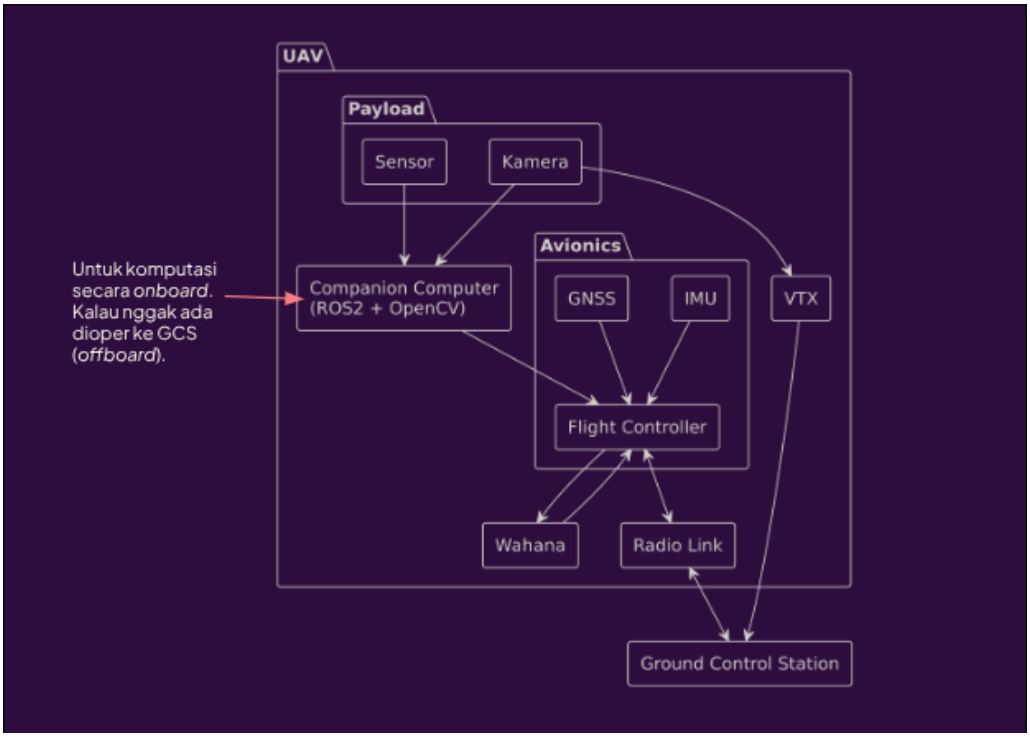

Figure 1: Diagram arsitektur sistem UAV

Gambaran Umum Sistem Secara umum, sistem UAV terdiri dari wahana, avionics, payload, sistem komunikasi, dan companion computer. Seluruh komponen ini bekerja bersama untuk memungkinkan UAV terbang, bernavigasi, berkomunikasi, serta menjalankan misi secara manual maupun otonom. Diagram arsitektur UAV digunakan untuk menunjukkan bagaimana data dan perintah mengalir antar komponen.

Wahana dan Sistem Propulsi

Wahana merupakan struktur fisik utama UAV. Untuk UAV jenis HTOL dan hybrid, wahana terdiri dari fuselage, sayap, dan ekor. Sementara itu, UAV VTOL menggunakan frame atau center plate dengan arm sebagai penopang motor. Sistem propulsi terdiri dari propeller, motor, Electronic Speed Controller (ESC), dan baterai Lithium Polymer. Propeller menghasilkan gaya angkat atau dorong melalui rotasi, motor mengubah energi listrik menjadi energi mekanik, dan ESC mengatur kecepatan serta arah putaran motor berdasarkan perintah dari Flight Controller. Saat ini, ESC yang digunakan merupakan ESC Alcentara, yaitu ESC buatan internal Aksantara.

Avionics dan Flight Controller

Avionics berfungsi sebagai sistem kendali utama UAV, dengan Flight Controller (FC) sebagai pusat pemrosesan. FC dapat dianalogikan sebagai sistem saraf motorik UAV

karena bertugas memproses data sensor dan menjalankan algoritma kontrol penerbangan, baik secara manual maupun otonom. FC yang digunakan antara lain Antares (FC buatan Aksantara) dan Matek F722-HD. Di dalamnya terdapat Microcontroller Unit (MCU) sebagai prosesor utama, barometer untuk estimasi ketinggian, serta Inertial Measurement Unit (IMU) yang terdiri dari accelerometer dan gyroscope untuk menentukan orientasi dan pergerakan UAV. Selain itu, terdapat blackbox yang menyimpan log penerbangan untuk keperluan analisis pasca-terbang. Sistem navigasi dilengkapi dengan modul GNSS yang menerima sinyal dari GPS, GLONASS, Galileo, dan BeiDou. Magnetometer yang biasanya terintegrasi dengan modul GNSS digunakan untuk menentukan arah mata angin, sedangkan airspeed sensor berfungsi mengukur kecepatan relatif UAV terhadap udara menggunakan pitot tube.

Payload

Payload merupakan komponen tambahan yang disesuaikan dengan kebutuhan misi. Contoh payload yang digunakan antara lain kamera FPV untuk video real-time, kamera depth untuk pemetaan kedalaman dan titik 3D, kamera tracking untuk estimasi posisi tanpa GPS, LiDAR untuk pemetaan presisi tinggi, serta sensor partikel untuk pemantauan kualitas udara.

Sistem Komunikasi Radio

Sistem Komunikasi Radio memungkinkan interaksi antara UAV dan Ground Control Station (GCS). Komunikasi ini membentuk *link*, yaitu jalur pertukaran data melalui frekuensi radio tertentu. Dalam sistem UAV, terdapat dua link utama. Link pertama adalah *Command and Control (C2) Link*, yang bertanggung jawab atas pengendalian UAV. Link ini terdiri dari uplink untuk mengirim perintah dari GCS ke UAV dan downlink untuk mengirim data telemetri seperti attitude, posisi, dan status baterai ke GCS. Link kedua adalah *Payload Link*, yang digunakan khusus untuk mengirim data payload seperti video dari VTX ke VRX secara real-time.

Companion Computer (CC)

Companion Computer (CC) berperan sebagai unit komputasi tingkat tinggi yang melengkapi Flight Controller. CC digunakan ketika misi membutuhkan pemrosesan lanjutan seperti perencanaan jalur, penghindaran rintangan, dan pemrosesan citra berbasis AI secara langsung di wahana. CC terhubung ke FC melalui komunikasi serial dan berkomunikasi menggunakan protokol MAVLink. Sensor dan payload mengirimkan data ke CC untuk diolah, kemudian CC mengirimkan perintah hasil komputasi ke FC untuk dieksekusi. Contoh perangkat CC yang digunakan adalah NVIDIA Jetson Nano dan Raspberry Pi.

Metode Komunikasi dan Protokol

Pengiriman data dalam sistem UAV dapat dilakukan secara *wired* maupun *wireless*. Komunikasi wired digunakan untuk jarak pendek antarkomponen di dalam UAV dengan keunggulan latensi rendah dan laju transfer tinggi. Sementara itu, komunikasi wireless digunakan untuk jarak jauh, seperti antara UAV dan GCS, dengan mobilitas lebih tinggi namun lebih rentan terhadap interferensi. Protokol komunikasi berfungsi sebagai aturan pengiriman data. Pada level perangkat keras, protokol yang umum digunakan antara lain UART, I2C, dan SPI. UART bersifat asinkron dan sering digunakan untuk

menghubungkan FC dengan modul telemetri, GNSS, dan Companion Computer. I2C dan SPI bersifat sinkron, dengan SPI digunakan untuk komponen berkecepatan tinggi seperti IMU dan blackbox. Pada level perangkat lunak, MAVLink digunakan sebagai protokol komunikasi pesan antar komponen UAV dan GCS. MAVLink bersifat ringan dan efisien, serta lebih sering digunakan di atas UDP untuk meminimalkan latensi. Setiap UAV memiliki System ID dan Component ID sebagai identitas, serta secara berkala mengirimkan pesan *heartbeat* untuk memastikan koneksi tetap aktif.

Logika Operasi dan Failsafe

Materi ini juga membahas logika operasi UAV melalui konsep *system state* dan *flight mode*. Sebelum terbang, UAV melewati tahapan initialization, pre-arm check, disarm, dan arm. Setelah itu, UAV dapat beroperasi dalam berbagai flight mode seperti Stabilize, Alt-Hold, Loiter, RTL, Auto, dan Guided. Untuk mengantisipasi kegagalan sistem, UAV dilengkapi dengan mekanisme failsafe. Pemicu failsafe dapat berupa hilangnya sinyal RC, terputusnya koneksi GCS, kondisi baterai kritis, gangguan sensor navigasi, atau pelanggaran geofence. Dalam kondisi tersebut, FC dapat mengambil tindakan otomatis seperti Return to Launch (RTL), mendarat, atau menghentikan pergerakan UAV.

Materi arsitektur sistem UAV ini sangat membuka wawasan karena menunjukkan betapa kompleksnya integrasi antara hardware, software, dan komunikasi. Walaupun informasinya cukup padat dan sempat terasa overwhelming, materi ini membantu membangun pemahaman fundamental yang penting untuk pengembangan software UAV di RSC.

Dasar Pemrograman (OOP, Design Pattern, dan Concurrency)

Materi ini membahas dasar-dasar pemrograman yang menjadi fondasi pengembangan software di RSC, khususnya untuk sistem UAV dan Ground Control Station (GCS). Fokus utama materi meliputi permasalahan pemrograman prosedural, pemrograman berorientasi objek (OOP), design pattern, error handling, serta pemrograman konkuren.

Permasalahan Pemrograman Prosedural

Pendekatan pemrograman prosedural cenderung menggunakan banyak fungsi global dan struktur data yang saling bergantung. Pada sistem kompleks seperti UAV dan GCS, pendekatan ini sering menimbulkan kode yang sulit dibaca, sulit dirawat, dan rawan error, terutama ketika sistem berkembang dan jumlah fitur bertambah. Kondisi ini sering disebut sebagai *spaghetti code*.

Pemrograman Berorientasi Objek (Object-Oriented Programming)

OOP diperkenalkan sebagai solusi untuk mengelola kompleksitas sistem dengan memodelkan komponen software sebagai objek yang merepresentasikan entitas nyata di sistem UAV.

Konsep utama OOP yang dipelajari meliputi:

- **Class dan Object**. Class berfungsi sebagai blueprint, sedangkan object merupakan instansi dari class tersebut di memori.
- **Encapsulation**. Data dan fungsi dibungkus dalam satu kesatuan untuk mencegah akses langsung yang tidak diinginkan.

- **Abstraction**. Menyembunyikan detail implementasi dan hanya menampilkan antarmuka yang relevan.
- **Inheritance**. Memungkinkan class baru mewarisi atribut dan method dari class lain untuk menghindari duplikasi kode.
- **Polymorphism**. Satu interface dapat memiliki banyak implementasi dengan perilaku yang berbeda.

Contoh Implementasi OOP

Sebagai contoh, koneksi UAV pada sistem GCS dapat dimodelkan sebagai sebuah class.

Listing 1: Contoh class koneksi UAV

```
class DroneConnection:
    def __init__(self, system_id):
        self.system_id = system_id
        self._connected = False

    def connect(self):
        self._connected = True

    def disconnect(self):
        self._connected = False
```

Dengan pendekatan ini, setiap UAV dapat direpresentasikan sebagai object yang terpisah dan dikelola secara modular.

Design Pattern

Design Pattern diperkenalkan sebagai solusi umum yang sudah teruji untuk permasalahan desain software yang sering muncul. Design pattern membantu engineer menulis kode yang lebih rapi, scalable, dan mudah dipahami oleh orang lain.

Contoh design pattern yang relevan dengan sistem UAV dan GCS antara lain:

- **Singleton**, untuk memastikan hanya ada satu instance, misalnya pada modul koneksi atau logger.
- **Observer**, untuk memantau perubahan data seperti telemetri UAV dan memperbarui UI secara otomatis.
- **Factory**, untuk membuat object UAV atau payload tanpa bergantung pada implementasi spesifik.

Error Handling (Try-Catch / Exception)

Materi ini juga membahas error handling menggunakan mekanisme exception. Pendekatan ini memungkinkan program menangani error secara terstruktur tanpa menghentikan seluruh sistem.

Listing 2: Contoh error handling menggunakan try-except

```
try:
    drone.connect()
except Exception as e:
    print(f"Connection␣error:␣{e}")
```

Error handling sangat penting pada sistem UAV karena kegagalan komunikasi, sensor, atau hardware dapat terjadi sewaktu-waktu.

Pemrograman Konkuren (Multithreading)

Pemrograman konkuren memungkinkan program menjalankan beberapa task secara bersamaan. Dalam sistem GCS, concurrency diperlukan untuk menangani telemetri, input pengguna, dan pengiriman perintah secara paralel tanpa saling menghambat.

Listing 3: Contoh multithreading sederhana

```
import threading

def telemetry_task():
    pass

telemetry_thread = threading.Thread(target=telemetry_task)
telemetry_thread.start()
```

Race Condition

Race condition terjadi ketika beberapa thread mengakses dan memodifikasi resource yang sama secara bersamaan tanpa sinkronisasi yang tepat. Hal ini dapat menyebabkan data menjadi tidak konsisten dan sulit dideteksi.

Mutex dan Sinkronisasi

Untuk mencegah race condition, digunakan mekanisme mutex atau lock. Mutex memastikan bahwa hanya satu thread yang dapat mengakses resource tertentu pada satu waktu.

Listing 4: Contoh penggunaan mutex untuk mencegah race condition

```
import threading

lock = threading.Lock()
shared_data = {}

def update_data(data):
    with lock:
        shared_data.update(data)
```

Relevansi dengan Sistem UAV dan GCS

Seluruh konsep pada materi ini sangat relevan dengan pengembangan software UAV dan GCS. OOP dan design pattern membantu membangun sistem yang terstruktur dan mudah dikembangkan, sedangkan concurrency memungkinkan sistem real-time berjalan secara responsif dan stabil.

Materi dasar pemrograman ini menunjukkan bahwa tantangan utama dalam pengembangan software UAV bukan hanya pada algoritma, tetapi juga pada desain sistem dan pengelolaan kompleksitas. Walaupun beberapa konsep seperti OOP cukup familiar, topik concurrency dan design pattern menuntut pemahaman yang lebih dalam karena dampaknya sangat besar terhadap stabilitas dan maintainability sistem.

Hari pertama ini cukup melelahkan namun membuka banyak perspektif baru. Selain memahami struktur RSC dan gambaran besar sistem UAV, aku juga belajar bahwa dokumentasi teknis adalah bagian penting dari proses engineering. Walaupun sempat mengalami kesulitan saat belajar LaTeX, proses tersebut menjadi latihan kemandirian dalam mencari solusi dan memahami tools yang akan sering digunakan ke depannya.