LEAF: PENGEMBANGAN KERANGKA KERJA AGENTIK MINIM KODE YANG DAPAT DIPERLUAS UNTUK OBSERVABILITAS KOMPUTASI AWAN MODERN

PROPOSAL TUGAS AKHIR

Ditulis sebagai Syarat untuk Pengajuan Tugas Akhir pada Program Studi Teknologi Informasi Fakultas Teknik Universitas Negeri Yogyakarta

Oleh:
MUHAMMAD NAUFAL KHOIRUL IMAMILHAQ ALHIFDI
NIM 22051130001

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
2026

LEAF: PENGEMBANGAN KERANGKA KERJA AGENTIK MINIM KODE YANG DAPAT DIPERLUAS UNTUK OBSERVABILITAS KOMPUTASI AWAN MODERN

PROPOSAL TUGAS AKHIR

Ditulis sebagai Syarat untuk Pengajuan Tugas Akhir pada Program Studi Teknologi Informasi Fakultas Teknik Universitas Negeri Yogyakarta

Oleh:
MUHAMMAD NAUFAL KHOIRUL IMAMILHAQ ALHIFDI
NIM 22051130001

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
2026

LEMBAR PERSETUJUAN

Proposal Tugas Akhir dengan Judul

LEAF: PENGEMBANGAN KERANGKA KERJA AGENTIK MINIM KODE YANG DAPAT DIPERLUAS UNTUK OBSERVABILITAS KOMPUTASI AWAN MODERN

Disusun oleh:
Muhammad Naufal Khoirul Imamilhaq Alhifdi
NIM 22051130001

telah memenuhi syarat dan disetujui oleh Dosen Pembimbing untuk dilaksanakan Seminar Proposal Tugas Akhir bagi yang bersangkutan.

Yogyakarta, 30 Maret 2026

Mengetahui,	Disetujui,
Koordinator Program Studi,	Dosen Pembimbing TA,
Nurkhamid, S.Si., M.Kom., Ph.D.	Muslikhin, S.Pd., M.Pd., Ph.D.
NIP. 19680707 199702 1 001	NIP. 19850101 201404 1 001

DAFTAR ISI

DAFTAR SINGKATAN

FWHM	:	*Full width half maximum*
rms	:	*root mean square*
RFS	:	*Rotary forcespinning*
PVP	:	Polivinil pirolidon
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional
SI	:	Satuan Internasional

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Perkembangan teknologi *cloud computing* telah mengubah fundamental organisasi dalam membangun dan mengoperasikan sistem perangkat lunak. Sistem modern yang berjalan di platform seperti Kubernetes dapat terdiri dari ratusan hingga ribuan *container* yang saling berinteraksi, menciptakan kompleksitas operasional yang belum pernah terjadi sebelumnya (Cloud Native Computing Foundation, 2024). Dalam konteks ini, praktik *Site Reliability Engineering* (SRE) yang diperkenalkan oleh Google menjadi pendekatan standar untuk memastikan keandalan sistem berskala besar (Beyer et al., 2016).

Untuk memahami dan mengelola sistem terdistribusi yang kompleks, konsep *observability* telah berevolusi dari pemantauan tradisional menjadi pendekatan yang lebih komprehensif (Sridharan, 2018). *Observability* modern dibangun di atas tiga pilar utama: *metrics* (data numerik tentang kondisi sistem), *logs* (catatan kejadian terstruktur), dan *traces* (pelacakan alur permintaan antar layanan) (Picoreti et al., 2018). Namun, ketiga pilar ini memiliki keterbatasan ketika digunakan secara terpisah; *metrics* tidak menjelaskan penyebab masalah, *logs* sulit dikorelasikan lintas layanan, dan *traces* memerlukan instrumentasi yang kompleks (Soldani et al., 2022).

Arsitektur Kubernetes dan *microservices* memperparah tantangan ini karena sifatnya yang dinamis dan terdistribusi. Sebuah permintaan pengguna dapat melewati puluhan layanan sebelum menghasilkan respons, dan kegagalan di satu komponen dapat memiliki efek domino ke seluruh sistem (Burns et al., 2018). Hal ini menciptakan kebutuhan mendesak akan *unified observability*, yaitu kemampuan untuk mengkorelasikan data dari ketiga pilar secara otomatis untuk memahami kondisi sistem secara holistik.

Untuk mengatasi kompleksitas ini, industri telah mengembangkan solusi *Artificial Intelligence for IT Operations* (AIOps) yang menerapkan teknik *machine learning* untuk mengotomatiskan tugas-tugas operasional (Notaro et al., 2021).

Platform komersial seperti Datadog, Dynatrace, dan Splunk menawarkan kemampuan deteksi anomali, korelasi kejadian, dan prediksi kegagalan berbasis AI (Dang et al., 2019). Namun, solusi-solusi ini memiliki keterbatasan fundamental yang menghambat adopsi dan efektivitasnya.

Pertama, arsitektur AIOps saat ini bersifat *monolithic*—menggabungkan *data ingestion*, analisis, dan aksi dalam satu platform *vendor* tunggal (Lyu et al., 2021). Organisasi tidak dapat dengan mudah mengganti atau menggabungkan komponen dari *vendor* berbeda untuk mendapatkan solusi terbaik. Kedua, masalah *vendor lock-in* membuat organisasi bergantung pada ekosistem tertutup, mengurangi fleksibilitas dan meningkatkan biaya jangka panjang.

Menanggapi kebutuhan akan interoperabilitas, standar komunikasi baru untuk agen AI telah muncul. *Model Context Protocol* (MCP) yang dikembangkan oleh Anthropic menyediakan antarmuka standar bagi agen untuk mengakses konteks dari peralatan-peralatan lain, seperti data dari Kubernetes, Prometheus, atau sistem *logging*, tanpa perlu implementasi API spesifik untuk setiap sumber data (LF Projects, 2025). Sementara itu, protokol *Agent-to-Agent* (A2A) dari Google memungkinkan komunikasi terstruktur antar agen yang berbeda, terlepas dari *framework* atau bahasa pemrograman yang digunakan (Google, 2024).

Kemunculan protokol-protokol ini menandai pergeseran paradigma dari sistem AI yang terisolasi menuju ekosistem agen yang dapat berinteroperasi (Xi et al., 2023). Beberapa *framework* pengembangan agen telah mengadopsi standar ini, misalnya Google Agent Development Kit (ADK) dan LangChain yang telah mendukung integrasi dengan MCP dan A2A secara *native* (Google Cloud, 2024; LangChain Inc., 2024).

Meskipun standar komunikasi telah tersedia dan diadopsi oleh *framework* modern, terdapat kesenjangan signifikan dalam aksesibilitas. Semua *framework* agen yang ada saat ini, termasuk Google ADK dan LangChain, memerlukan keahlian pemrograman yang substansial untuk digunakan. Pengembang harus memahami pola-pola SDK, menulis kelas agen dalam bahasa pemrograman Python atau JavaScript, mengimplementasikan *binding* untuk *tools*, dan mengelola logika koordinasi yang kompleks, semuanya dalam kode (LangChain Inc., 2024).

Situasi ini menciptakan hambatan masuk (*entry barrier*) yang tinggi ke era agentik. Pakar bidang tertentu seperti praktisi SRE, analis data, atau tim operasional yang memahami alur kerja bidang spesifik mereka dengan baik, tidak dapat membangun aplikasi agentik tanpa terlebih dahulu menjadi pengembang perangkat lunak. Terdapat dilema pilihan yang semu (*false choice*): menerima keterbatasan alat *no-code* yang sederhana, atau berkomitmen untuk menguasai pemrograman secara penuh. Tidak ada *framework* yang menawarkan kesederhanaan deklaratif dengan ekstensibilitas berbasis standar.

Dari perspektif teoritis, sistem multi-agen (*Multi-Agent Systems*/MAS) telah dipelajari secara ekstensif dalam literatur kecerdasan buatan. Wooldridge (2009) mendefinisikan agen sebagai entitas yang otonom, reaktif terhadap lingkungan, proaktif dalam mencapai tujuan, dan mampu berinteraksi secara sosial dengan agen lain. Jennings et al. (1998) memetakan berbagai paradigma koordinasi agen, termasuk sistem *blackboard*, *contract nets*, dan dekomposisi tugas hierarkis.

Literatur sistem multi-agen menawarkan berbagai pola koordinasi seperti *supervisor*, *blackboard*, dan *contract nets* (Jennings et al., 1998). LEAF mengadopsi pendekatan yang fleksibel: menyediakan pola orkestrasi bawaan yang umum digunakan (*supervisor*, *sequential*, *parallel*) sambil memungkinkan implementasi pola kustom melalui mekanisme ekstensibilitas A2A. Pendekatan ini memisahkan keputusan protokol (yang diopinikan oleh LEAF) dari keputusan pola koordinasi (yang diserahkan kepada pengguna sesuai kebutuhan domain).

Berdasarkan analisis kesenjangan di atas, penelitian ini mengusulkan **LEAF** (*Low-code Extensible Agentic Framework*)—sebuah *framework* agentik *low-code* yang dirancang untuk mendemokratisasi pengembangan aplikasi multi-agen. LEAF menggunakan pendekatan deklaratif berbasis manifes YAML/JSON, analog dengan cara Kubernetes mengabstraksi manajemen infrastruktur melalui konfigurasi deklaratif (Artac et al., 2017; Burns et al., 2018; Morris, 2020).

Kontribusi utama LEAF adalah menyediakan model kompleksitas progresif (*progressive complexity model*) dengan tiga tingkat:

1. **Level 1 (Deklaratif Murni)**: Pengguna mendefinisikan agen sepenuhnya dalam YAML, mereferensikan *tools* MCP dari komunitas, tanpa perlu

menulis kode sama sekali.

2. **Level 2 (Deklaratif + Referensi)**: Manifes YAML dapat mereferensikan agen A2A eksternal atau server MCP kustom yang dikembangkan secara terpisah.
3. **Level 3 (Implementasi Kustom)**: Pengembang yang memerlukan kontrol penuh dapat menulis agen kustom menggunakan LEAF SDK, yang tetap dapat berinteroperasi melalui standar A2A/MCP.

Dengan demikian, standar MCP/A2A berfungsi sebagai "*escape hatch*"—memungkinkan kesederhanaan *low-code* tanpa menjadi pembatas yang mengekang. Pakar domain dapat memulai dengan konfigurasi deklaratif untuk kasus penggunaan umum (Level 1), kemudian berkembang sesuai kebutuhan tanpa harus memulai dari awal.

Arsitektur LEAF dibangun di atas konsep "*Opinionated Grid*" yang memisahkan penalaran AI (*reasoning*) dari pipa infrastruktur (*plumbing*) melalui protokol komunikasi terstandarisasi: *Southbound* (MCP untuk akses *tools*), *Northbound* (A2A untuk interaksi pengguna), dan *East-West* (A2A untuk koordinasi antar agen). Detail arsitektur grid dijelaskan lebih lanjut pada Bab II (Google, 2024; LF Projects, 2025).

LEAF mengorganisasi agen dalam taksonomi empat lapis dengan hierarki *Orchestrator* $\rightarrow$ *Processor* $\rightarrow$ *Sensor*/*Executor*. *Orchestrator* sendiri merupakan tipe agen yang mengoordinasi agen-agen *Processor*, yang kemudian mengelola agen *Sensor* (pengumpulan data) dan *Executor* (implementasi aksi). LEAF menyediakan pola orkestrasi bawaan: *supervisor* (perutean dinamis berbasis LLM), *sequential* (*pipeline* berurutan), dan *parallel* (eksekusi bersamaan). Pengguna dapat mengimplementasikan pola kustom melalui mekanisme Level 2/3.

Pada lapisan *tools*, LEAF membedakan kapabilitas *sensor* (operasi baca) dan kapabilitas *executor* (operasi tulis/mutasi). Pembatasan ini ditegakkan pada lapisan agen: agen *Sensor* hanya dapat menggunakan kapabilitas baca, sedangkan agen *Executor* hanya dapat menggunakan kapabilitas mutasi. Sebagai prinsip desain inti, LEAF menekankan pengawasan manusia (*human-in-the-loop*); berdasarkan pedoman interaksi manusia-AI (Amershi et al., 2019; Shneiderman, 2020), semua

aksi *Executor* memerlukan persetujuan eksplisit sebelum dieksekusi.

Kontribusi penelitian ini adalah sebagai *framework* agentik *low-code* pertama yang menggabungkan desain berbasis manifes deklaratif dengan ekstensibilitas berbasis standar (MCP/A2A). LEAF memungkinkan pakar domain untuk membangun sistem multi-agen tanpa pemrograman, sambil mempertahankan "*escape hatch*" untuk kustomisasi lanjutan oleh pengembang. Penelitian ini mencakup: (1) cetak biru arsitektur, (2) spesifikasi manifes deklaratif (JSON Schema), dan (3) implementasi referensi.

Ruang Lingkup Tesis: Meskipun LEAF dirancang sebagai *framework* tujuan umum yang dapat diterapkan pada domain apa pun yang memerlukan koordinasi multi-agen (misalnya pengembangan perangkat lunak, *pipeline* data, otomatisasi bisnis), tesis ini secara spesifik berfokus pada ***observability* Kubernetes dan operasi SRE** sebagai studi kasus implementasi untuk memvalidasi pola-pola inti *framework*.

B. Identifikasi Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat diidentifikasi beberapa permasalahan sebagai berikut:

1. Kompleksitas sistem *cloud computing* modern, khususnya arsitektur Kubernetes dan *microservices*, menciptakan tantangan *observability* yang sulit diatasi dengan pendekatan pemantauan tradisional.
2. Arsitektur platform AIOps yang bersifat *monolithic* menyebabkan *vendor lock-in*, membatasi fleksibilitas organisasi dalam mengadopsi solusi terbaik dari berbagai penyedia.
3. *Framework* agentik yang ada saat ini memerlukan keahlian pemrograman yang substansial, menciptakan hambatan masuk (*entry barrier*) yang tinggi bagi pakar domain non-programmer.
4. Tidak tersedia pendekatan deklaratif/*low-code* pada *framework* agentik yang ada, memaksa pengguna memilih antara kesederhanaan tanpa fleksibilitas atau fleksibilitas dengan kompleksitas tinggi.
5. Meskipun standar komunikasi agen (MCP, A2A) telah tersedia, adopsi praktis dalam bentuk *framework* yang mudah digunakan masih terbatas.

C. Batasan Masalah

Dari berbagai masalah yang teridentifikasi, penelitian ini membatasi ruang lingkup pada aspek-aspek berikut:

1. Penelitian berfokus pada pendekatan agentik berbasis *Large Language Model* (LLM), bukan pada *pipeline machine learning* tradisional atau sistem berbasis aturan.
2. *Framework* yang dikembangkan menggunakan pendekatan *low-code* dengan manifes deklaratif (YAML/JSON), bukan antarmuka *no-code* berbasis GUI.
3. Validasi dan implementasi referensi difokuskan pada domain *observability* Kubernetes dan operasi SRE sebagai studi kasus, meskipun *framework* dirancang untuk tujuan umum.
4. Ekstensibilitas *framework* dibangun di atas protokol standar MCP dan A2A, tidak mengembangkan protokol komunikasi proprietary baru.

Pembatasan pada domain Kubernetes/SRE dipilih karena kompleksitas yang representatif dan ketersediaan infrastruktur pengujian, sementara masalah kompleksitas *observability* secara umum telah ditangani oleh berbagai alat yang ada.

D. Rumusan Masalah

Berdasarkan identifikasi dan batasan masalah di atas, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana merancang arsitektur *framework* agentik *low-code* yang memanfaatkan protokol standar MCP dan A2A untuk interoperabilitas?
2. Bagaimana menyusun spesifikasi format manifes deklaratif yang memungkinkan definisi agen tanpa pemrograman?
3. Bagaimana mengimplementasikan dan memvalidasi *framework* tersebut pada domain *observability* Kubernetes dan operasi SRE?

E. Tujuan Penelitian

Sesuai dengan rumusan masalah yang telah ditetapkan, tujuan penelitian ini adalah:

1. Merancang arsitektur *framework* agentik *low-code* berbasis protokol MCP

dan A2A yang memungkinkan interoperabilitas antar komponen.

2. Menyusun spesifikasi format manifes deklaratif (JSON Schema) untuk definisi agen, *tools*, dan orkestrasi tanpa memerlukan pemrograman.
3. Mengimplementasikan dan memvalidasi *framework* LEAF melalui studi kasus aplikasi *observability* Kubernetes dan operasi SRE.

F. Manfaat Penelitian

Hasil penelitian ini diharapkan dapat memberikan manfaat baik secara teoritis maupun praktis.

1. Manfaat Teoritis

1. Memberikan kontribusi pada pengembangan pola desain *framework* agentik *low-code*, memperkaya literatur di bidang sistem multi-agen dan *human-computer interaction*.
2. Menyediakan arsitektur referensi untuk menggabungkan paradigma deklaratif dengan standar komunikasi agen (MCP/A2A), yang dapat menjadi dasar penelitian lanjutan.

2. Manfaat Praktis

1. Bagi praktisi SRE dan tim operasional: menyediakan alat yang memungkinkan pembangunan sistem *observability* berbasis agen tanpa memerlukan keahlian pemrograman mendalam.
2. Bagi organisasi: mengurangi hambatan adopsi otomatisasi agentik, memungkinkan pakar domain untuk langsung berkontribusi dalam pengembangan solusi.
3. Bagi komunitas akademik UNY: menyediakan implementasi referensi yang dapat digunakan untuk penelitian dan pengembangan lebih lanjut di bidang sistem multi-agen dan AIOps.

BAB II
TINJAUAN PUSTAKA

A. Teori Dasar Komponen Elektronika

Bagian ini membahas teori dasar mengenai komponen elektronika yang digunakan dalam perancangan sistem. Setiap komponen memiliki karakteristik khusus yang mempengaruhi kinerja keseluruhan sistem. Pemahaman tentang karakteristik komponen-komponen ini sangat penting dalam mengoptimalkan fungsi dan stabilitas rangkaian.

1. Jenis dan Karakteristik Komponen Pasif

Komponen pasif seperti resistor, kapasitor, dan induktor memiliki fungsi dasar dalam pengaturan arus dan tegangan dalam sirkuit. Bagian ini menguraikan jenis-jenis komponen pasif serta karakteristik utama yang mempengaruhi performa dan fungsi komponen tersebut dalam sirkuit elektronika.

2. Jenis dan Karakteristik Komponen Aktif

Komponen aktif, seperti transistor, dioda, dan IC, memainkan peran penting dalam penguatan dan pengaturan sinyal. Bagian ini menjelaskan berbagai jenis komponen aktif yang digunakan dalam proyek serta karakteristik utamanya, yang menentukan efektivitas dan efisiensi sistem elektronika.

3. Peran dan Fungsi Modul dalam Sistem Elektronika

Modul-modul elektronika memberikan fungsionalitas tambahan yang membantu dalam memperkuat performa sistem. Bagian ini mengulas modul-modul yang sering digunakan, seperti modul daya atau komunikasi, serta peran masing-masing dalam mendukung integrasi sistem yang lebih efisien.

4. Analisis Daya dan Efisiensi Komponen

Analisis daya dan efisiensi komponen adalah aspek penting dalam desain sistem yang hemat energi. Bagian ini membahas cara-cara mengevaluasi dan mengoptimalkan daya yang dikonsumsi oleh komponen, yang berperan dalam meningkatkan efisiensi energi dari sistem secara keseluruhan.

B. Sistem dan Teknik Rangkaian Elektronika

Bagian ini membahas berbagai sistem dan teknik yang digunakan dalam perancangan rangkaian elektronika, baik analog maupun digital. Setiap teknik ini memungkinkan sistem berfungsi dengan lebih efektif sesuai dengan kebutuhan aplikasi.

1. Konsep Dasar Rangkaian Analog

Rangkaian analog digunakan untuk memproses sinyal kontinu dan memainkan peran penting dalam berbagai aplikasi. Bagian ini menjelaskan prinsip-prinsip dasar yang digunakan dalam rangkaian analog, termasuk elemen-elemen utamanya dan penggunaannya.

2. Konsep Dasar Rangkaian Digital

Rangkaian digital beroperasi dengan sinyal diskrit, cocok untuk pemrosesan informasi digital. Bagian ini menguraikan prinsip dasar rangkaian digital serta komponen-komponen utama yang mendukung fungsi-fungsi digital dalam proyek ini.

3. Teknik Pengolahan Sinyal pada Sistem Elektronika

Pengolahan sinyal adalah proses penting untuk interpretasi informasi dari lingkungan. Bagian ini membahas metode umum dalam pengolahan sinyal yang diterapkan pada sistem elektronika, termasuk teknik yang digunakan dalam pemfilteran atau pemrosesan data.

4. Pengkabelan dan Pengaturan Sirkuit untuk Keandalan Sistem

Pengkabelan dan tata letak yang baik meningkatkan keandalan sistem secara keseluruhan. Bagian ini menguraikan teknik pengkabelan dan pengaturan sirkuit yang efektif, serta bagaimana hal ini dapat mempengaruhi performa sistem.

5. Pengendalian dan Penggerak (Motor Driver, Relay, dsb.)

Bagian ini menjelaskan penggunaan penggerak seperti motor driver dan relay untuk menggerakkan komponen mekanis. Diperlukan teknik pengendalian khusus untuk memastikan bahwa setiap penggerak bekerja sesuai dengan tujuan sistem.

C. Teknologi yang Digunakan

Bagian ini mengulas teknologi yang umum digunakan dalam proyek berbasis elektronika, seperti mikrokontroler, sensor, aktor, dan teknologi komunikasi nirkabel. Teknologi ini memungkinkan sistem untuk merespons lingkungan dan berinteraksi dengan pengguna.

1. Mikrokontroler dan Mikroprosesor

Mikrokontroler dan mikroprosesor berfungsi sebagai unit pemrosesan utama dalam sistem elektronika. Bagian ini menguraikan arsitektur dasar, bahasa pemrograman yang relevan, serta protokol komunikasi yang sering digunakan dalam proyek ini.

2. Sensor dan Aktuator

Sensor dan aktuator memungkinkan interaksi sistem dengan lingkungannya. Bagian ini membahas jenis-jenis sensor yang digunakan, cara kerja, dan integrasinya ke dalam sistem agar sistem dapat mengumpulkan data dan merespons secara aktif.

3. Teknologi Nirkabel

Teknologi nirkabel seperti Bluetooth dan Wi-Fi memungkinkan komunikasi jarak jauh dalam sistem IoT. Bagian ini menguraikan jenis teknologi nirkabel yang relevan, termasuk protokol komunikasi dan aspek keamanan yang perlu dipertimbangkan.

D. Metode Kontrol dan Kecerdasan Buatan

Bagian ini membahas metode kontrol dan kecerdasan buatan yang diterapkan dalam sistem elektronika untuk mencapai pengendalian yang lebih cerdas dan otomatis, seperti kontrol PID, logika fuzzy, dan deep learning.

1. Pengendalian PID (Proportional-Integral-Derivative)

PID adalah metode kontrol yang efektif dalam mengatur respons sistem. Bagian ini menjelaskan prinsip dasar PID, aplikasinya dalam pengaturan sistem elektronika, serta teknik tuning yang dapat meningkatkan stabilitas dan respons sistem.

2. Fuzzy Logic Control

Logika fuzzy adalah metode kontrol fleksibel yang sering digunakan dalam sistem nonlinear. Bagian ini menjelaskan konsep dasar logika fuzzy, serta cara implementasi dan manfaatnya dalam pengendalian sistem yang kompleks.

3. Deep Learning

Deep learning memungkinkan sistem untuk belajar dari data, yang sangat berguna dalam aplikasi otomatisasi. Bagian ini menguraikan algoritma dasar dalam deep learning, seperti CNN dan RNN, serta penerapannya dalam pengembangan sistem IoT.

4. Perbandingan dan Pemilihan Metode yang Sesuai

Bagian ini membahas perbandingan antara berbagai metode kontrol yang tersedia, menjelaskan kelebihan dan kekurangannya masing-masing, serta bagaimana memilih metode yang paling sesuai untuk aplikasi proyek ini.

E. Konsep Engineering Design Process

Engineering Design Process adalah metodologi sistematis yang digunakan untuk merancang sistem secara efektif. Bagian ini menguraikan prinsip utama dan langkah-langkah dari Engineering Design Process dalam konteks pengembangan sistem elektronika.

1. Pengertian dan Langkah-langkah Engineering Design Process

Engineering Design Process adalah proses desain iteratif yang mencakup beberapa tahapan untuk mencapai desain yang optimal. Bagian ini menjelaskan langkah-langkah dasar yang terlibat dan bagaimana proses ini diadaptasi dalam proyek ini.

2. Aplikasi Engineering Design Process pada Proyek Elektronika

Bagian ini membahas penerapan Engineering Design Process dalam proyek elektronika untuk mencapai hasil desain yang optimal. Diuraikan langkah-langkah praktis dalam menerapkan metodologi ini.

3. Studi Kasus Implementasi Engineering Design Process dalam Desain Elektronika

Studi kasus ini menunjukkan contoh penerapan Engineering Design Process dalam desain sistem yang relevan dengan proyek. Dengan studi kasus ini, pembaca dapat memahami implementasi proses desain secara nyata.

4. Teknik Evaluasi dan Optimasi Desain

Teknik evaluasi dan optimasi desain merupakan langkah penting dalam proses desain yang berkelanjutan. Bagian ini menjelaskan metode yang digunakan untuk mengevaluasi dan menyempurnakan desain agar mencapai hasil terbaik.

F. Penelitian Terdahulu yang Relevan

Bagian ini berisi ulasan terhadap penelitian terdahulu yang relevan dengan proyek ini. Tujuannya adalah untuk melihat pendekatan yang telah digunakan, menemukan kelebihan dan kekurangannya, serta mengidentifikasi inovasi yang dapat dikembangkan.

1. Tinjauan Penelitian Terdahulu tentang Proyek Serupa

Bagian ini membahas penelitian terdahulu yang serupa dengan proyek ini. Tinjauan ini bertujuan untuk memahami bagaimana proyek ini dapat memberikan kontribusi yang berbeda atau lebih baik.

2. Analisis Kekurangan dan Kelebihan Metode pada Penelitian Terdahulu

Setiap metode yang digunakan dalam penelitian terdahulu memiliki kelebihan dan kekurangan. Bagian ini mengidentifikasi aspek yang perlu diperbaiki atau dikembangkan lebih lanjut berdasarkan analisis metode-metode tersebut.

3. Inovasi dan Kontribusi yang Dibawa dalam Penelitian Ini

Penelitian ini membawa inovasi tertentu yang berkontribusi dalam memperkaya hasil penelitian sebelumnya. Bagian ini menjelaskan kontribusi utama proyek ini terhadap bidang elektronika, serta perbedaan yang ditawarkan.

BAB III

KONSEP RANCANGAN ALAT DAN PENGUJIAN

A. Metode Pengerjaan Project Berbasis Engineering Design Process

Bagian ini menjelaskan metode pengerjaan proyek yang mengadopsi pendekatan Engineering Design Process. Pendekatan ini digunakan untuk memastikan perancangan dan pengembangan proyek dilakukan secara sistematis, mulai dari identifikasi masalah hingga evaluasi akhir. Langkah-langkah dalam metode ini membantu dalam mencapai solusi yang optimal dan terukur.

1. Identifikasi Masalah

Tahap identifikasi masalah bertujuan untuk menguraikan permasalahan utama yang dihadapi dan memerlukan solusi. Pada tahap ini, dilakukan analisis untuk memahami aspek-aspek penting dari masalah dan menentukan faktor-faktor yang perlu diatasi melalui proyek ini.

2. Definisi Kebutuhan

Berdasarkan masalah yang telah diidentifikasi, tahap ini berfokus pada definisi kebutuhan proyek secara jelas dan terstruktur. Kebutuhan ini meliputi spesifikasi teknis, fungsi yang diinginkan, serta kriteria-kriteria lain yang harus dipenuhi agar solusi dapat berfungsi dengan baik.

3. Generasi Ide dan Solusi

Pada tahap ini, berbagai ide dan solusi alternatif dikembangkan dan dievaluasi. Bagian ini menjelaskan proses brainstorming untuk menghasilkan ide yang inovatif, termasuk analisis terhadap kelebihan dan kekurangan dari setiap alternatif solusi yang diusulkan.

4. Perencanaan dan Desain Awal

Desain awal sistem dikembangkan berdasarkan solusi yang dipilih, dengan mempertimbangkan aspek teknis dan kebutuhan yang telah didefinisikan. Bagian ini menyajikan perencanaan mengenai struktur sistem, alat, dan bahan yang akan digunakan, serta jadwal kerja proyek secara keseluruhan.

5. Pembuatan Prototipe

Prototipe dibuat untuk menguji konsep dan desain awal dari sistem. Bagian ini menguraikan langkah-langkah dalam proses pembuatan prototipe, termasuk alat dan bahan yang diperlukan, serta tantangan yang mungkin dihadapi selama proses.

6. Pengujian dan Evaluasi

Prototipe yang telah dibuat kemudian diuji untuk menilai kinerjanya terhadap kebutuhan dan spesifikasi yang telah ditetapkan. Bagian ini menjelaskan prosedur pengujian yang diterapkan, metode pengumpulan data, serta analisis terhadap hasil yang diperoleh.

7. Perbaikan dan Penyempurnaan

Berdasarkan hasil pengujian dan evaluasi, dilakukan perbaikan untuk menyempurnakan sistem. Bagian ini menjelaskan penyesuaian yang dilakukan untuk meningkatkan kinerja sistem, serta proses iterasi yang dilakukan hingga mencapai hasil yang optimal.

B. Perancangan Sistem Elektronika

Bagian ini menguraikan perancangan dari sisi elektronika yang menjadi inti dari sistem. Perancangan ini mencakup blok diagram, pemilihan komponen, dan perancangan rangkaian.

1. Blok Diagram Sistem

Blok diagram sistem memberikan gambaran umum mengenai arsitektur sistem secara keseluruhan. Bagian ini menyajikan diagram beserta penjelasan fungsi setiap blok yang terdapat di dalam sistem, termasuk bagaimana setiap blok berinteraksi.

2. Pemilihan dan Spesifikasi Komponen Elektronika

Pemilihan komponen elektronika dilakukan berdasarkan kebutuhan dari desain sistem. Bagian ini menjelaskan spesifikasi teknis dari setiap komponen yang digunakan, seperti mikrokontroler, sensor, aktuator, dan komponen pendukung lainnya.

3. Perancangan Rangkaian Elektronika

Perancangan rangkaian elektronika bertujuan untuk mencapai fungsionalitas yang diinginkan dari sistem. Bagian ini menguraikan skema rangkaian, penjelasan aliran arus dan tegangan, serta hubungan antar komponen dalam sistem.

C. Perancangan Mekanik

Bagian ini membahas perancangan mekanik dari sistem yang mendukung komponen elektronik secara fisik, termasuk struktur dan pemilihan bahan.

1. Spesifikasi Desain Mekanik

Desain mekanik disusun berdasarkan kebutuhan fisik sistem untuk memastikan bahwa komponen elektronika terlindungi dan dapat berfungsi dengan baik. Bagian ini menjelaskan spesifikasi teknis desain mekanik yang digunakan.

2. Pemilihan Bahan dan Komponen Mekanik

Pemilihan bahan didasarkan pada kriteria seperti kekuatan, ketahanan, dan biaya. Bagian ini menguraikan bahan dan komponen mekanik yang digunakan untuk konstruksi sistem, serta alasan pemilihan bahan tersebut.

3. Desain Struktur dan Konstruksi

Struktur dan konstruksi sistem dirancang untuk memberikan dukungan fisik yang stabil. Bagian ini menjelaskan proses desain dan konstruksi dari struktur mekanik sistem, serta tantangan yang mungkin dihadapi.

D. Perancangan Perangkat Lunak

Bagian ini mencakup perancangan perangkat lunak yang mengendalikan sistem, meliputi diagram alir, pengembangan kode, dan pengujian perangkat lunak.

1. Flowchart atau Diagram Alir Perangkat Lunak

Diagram alir menggambarkan alur kerja dari perangkat lunak yang mengontrol sistem. Bagian ini menyajikan flowchart lengkap yang menunjukkan logika dan struktur kontrol dari perangkat lunak.

2. Pemrograman dan Pengembangan Kode

Kode perangkat lunak dikembangkan untuk mendukung fungsionalitas sistem. Bagian ini menguraikan struktur dan logika dari program yang dibuat, bahasa pemrograman yang digunakan, serta strategi pengembangan kode.

3. Pengujian Kode Perangkat Lunak

Setelah kode perangkat lunak dikembangkan, dilakukan pengujian untuk memastikan bahwa perangkat lunak berfungsi sesuai yang diharapkan. Bagian ini menjelaskan metode pengujian, jenis uji (misalnya, uji unit dan uji integrasi), serta hasil yang diperoleh.

E. Perancangan Integrasi Sistem

Bagian ini menjelaskan proses integrasi antara komponen elektronik, mekanik, dan perangkat lunak agar sistem dapat bekerja sebagai satu kesatuan.

1. Integrasi Komponen Elektronika, Mekanik, dan Perangkat Lunak

Proses integrasi bertujuan untuk menyatukan seluruh komponen sistem agar berfungsi sebagai satu kesatuan. Bagian ini menguraikan langkah-langkah dalam proses integrasi dan memastikan semua komponen bekerja secara sinkron.

2. Pengujian Awal dan Penyempurnaan Integrasi

Pengujian awal dilakukan setelah integrasi untuk menilai performa sistem secara keseluruhan. Bagian ini menjelaskan hasil pengujian integrasi dan modifikasi yang diperlukan untuk meningkatkan kinerja sistem.

F. Rencana Pengujian

Bagian ini merencanakan pengujian yang komprehensif terhadap sistem untuk memastikan bahwa semua komponen dan fungsi bekerja dengan baik.

1. Metode Pengujian Sistem

Metode pengujian dipilih berdasarkan tujuan dan kebutuhan proyek. Bagian ini menjelaskan berbagai metodologi pengujian yang dirancang, seperti uji kinerja, uji stabilitas, dan uji kompatibilitas.

2. Prosedur Pengujian

Prosedur pengujian disusun untuk menguji sistem secara menyeluruh, mulai dari pengaturan awal hingga pelaksanaan uji. Bagian ini menjelaskan langkah-langkah pengujian secara detail untuk menjamin konsistensi dan keandalan hasil.

3. Kriteria Keberhasilan Pengujian

Kriteria keberhasilan ditentukan untuk mengevaluasi kinerja sistem berdasarkan parameter-parameter tertentu. Bagian ini menguraikan kriteria-kriteria keberhasilan yang digunakan, seperti ketepatan, keandalan, dan efisiensi sistem.

Pada bagian ini akan dijelaskan beberapa tahap utama dalam proses pengembangan proyek dan beberapa rekomendasi peningkatan yang dapat dilakukan. Daftar tahapan pengembangan proyek akan disajikan dalam bentuk bernomor untuk menunjukkan urutan logis dari proses, sementara rekomendasi peningkatan akan disajikan dalam bentuk daftar berpoin untuk mempermudah identifikasi setiap item secara mandiri.

DAFTAR PUSTAKA

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., et al. (2019). Guidelines for human-ai interaction. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., and Tamburri, D. A. (2017). Devops: Introducing infrastructure-as-code. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 497–498. IEEE.

Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Sebastopol, CA.

Burns, B., Beda, J., and Hightower, K. (2018). *Kubernetes: Up and Running*. O'Reilly Media, Sebastopol, CA, 2nd edition.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J. (2016). Borg, omega, and kubernetes. volume 14, pages 70–93. ACM.

Cloud Native Computing Foundation (2024). Kubernetes documentation. `https://kubernetes.io/docs/`.

Corkill, D. D. (1991). Blackboard systems. *AI Expert*, 6(9):40–47.

Dang, Y., Lin, Q., and Huang, P. (2019). Aiops: Real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5. IEEE.

Dorri, A., Kanhere, S. S., and Jurdak, R. (2018). Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593.

Google (2024). Agent-to-agent (a2a) protocol specification. `https://google.github.io/A2A/`.

Google Cloud (2024). Google agent development kit (adk) documentation. `https://cloud.google.com/vertex-ai/docs/generative-ai/agents/adk`.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38.

LangChain Inc. (2024). Langchain: Building applications with llms through composability. `https://python.langchain.com/docs/`.

LF Projects, L. (2025). Specification - model context protocol. `https://modelcontextprotocol.io/specification/2025-11-25`.

Lyu, Y., Li, H., Sheng, Q. Z., et al. (2021). Towards a unified framework for aiops. *IEEE Access*, 9:98288–98303.

Morris, K. (2020). Infrastructure as code: Dynamic systems for the cloud age.

Notaro, P., Cardoso, J., and Gerndt, M. (2021). A survey of aiops methods for failure management. *ACM Transactions on Intelligent Systems and Technology*, 12(6):1–45.

Picoreti, R., do Carmo, A. B., de Queiroz, F. M., Garcia, A. S. A. C., Vassallo, R. F., and Simeonidou, D. (2018). Multilevel observability in cloud orchestration. In *2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing*, pages 776–784. IEEE.

Shneiderman, B. (2020). Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human-Computer Interaction*, 36(6):495–504.

Soldani, J., Tamburri, D. A., and Van Den Heuvel, W.-J. (2022). The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146:215–232.

Sridharan, C. (2018). *Distributed Systems Observability: A Guide to Building Robust Systems*. O'Reilly Media, Sebastopol, CA.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, UK, 2nd edition.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., et al. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

LAMPIRAN A

KODE PROGRAM

Lampiran A.1. Program Pembacaan Sensor Ultrasonic

Lampiran A.2. Program Keseluruhan Proyek Akhir

LAMPIRAN B

GAMBAR-GAMBAR

Lampiran B.1. Foto Aktivitas Kegiatan Proyek Akhir

Lampiran B.2. Foto Produk Proyek Akhir