



REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

SaR-PaM (Self-Balancing Robot with Path Memorization)

GROUP 15

Bryan Herdianto	2306210885
Daffa Bagus Dhiananto	2306250756
Nabiel Harits Utomo	2306267044
Tjokorde Gde Agung Abel Putra	2206059736

PREFACE

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa karena dengan rahmat dan karunia-Nya, laporan proyek akhir berjudul **SaR-PaM (Self-Balancing Robot with Path Memorization)** ini dapat diselesaikan dengan baik. Proyek ini merupakan bagian dari rangkaian praktikum Real Time System and Internet of Things dan bertujuan untuk merancang sebuah robot self-balancing berbasis ESP32 yang mampu menjaga kestabilan menggunakan sensor IMU dan kendali PID, sekaligus dapat merekam serta mengulangi jalur pergerakan melalui mekanisme path memorization. Selain itu, sistem juga menyediakan dua jenis metode komunikasi, yaitu kontrol melalui WebSocket (WiFi) untuk jangkauan lebih luas.

Penyelesaian proyek ini tidak terlepas dari dukungan dan bantuan berbagai pihak. Kami mengucapkan terima kasih kepada seluruh asisten laboratorium Digital FTUI yang telah memberikan arahan, bimbingan, serta evaluasi selama proses praktikum dan pengembangan proyek ini. Kami juga berterima kasih kepada rekan-rekan kelompok yang telah bekerja sama secara aktif dalam merancang, mengimplementasikan, menguji, dan mendokumentasikan sistem hingga proyek ini dapat terselesaikan.

Kami menyadari bahwa laporan ini masih memiliki kekurangan dan keterbatasan. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun demi peningkatan kualitas pada proyek maupun dokumentasi di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan menjadi referensi bagi pengembangan teknologi di bidang robotika dan Internet of Things.

Depok, December 07, 2025

Group 15

TABLE OF CONTENTS

CHAPTER 1.....	4
INTRODUCTION.....	4
1.1 PROBLEM STATEMENT.....	4
1.3 ACCEPTANCE CRITERIA.....	5
1.4 ROLES AND RESPONSIBILITIES.....	6
1.5 TIMELINE AND MILESTONES.....	6
CHAPTER 2.....	8
IMPLEMENTATION.....	8
2.1 HARDWARE DESIGN AND SCHEMATIC.....	8
2.2 SOFTWARE DEVELOPMENT.....	9
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	12
CHAPTER 3.....	15
TESTING AND EVALUATION.....	15
3.1 TESTING.....	15
3.2 RESULT.....	15
3.3 EVALUATION.....	17
CHAPTER 4.....	19
CONCLUSION.....	19

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Pengembangan robot self-balancing menghadapi beberapa tantangan yang berkaitan dengan kestabilan dan kemampuan navigasi. Robot dua roda secara alami tidak stabil dan membutuhkan sistem kendali yang cepat dan akurat agar dapat mempertahankan keseimbangannya. Tanpa pengolahan data sensor yang baik dan algoritma kontrol yang tepat, robot akan mudah jatuh, kehilangan arah, atau gagal merespons perubahan lingkungan.

Selain itu, kebanyakan robot self-balancing hanya mampu bergerak secara manual dikontrol oleh alat dan tidak memiliki kemampuan untuk mengingat atau mengulangi jalur yang telah dilalui. Hal ini membatasi fungsionalitas robot dalam skenario yang memerlukan navigasi berulang, misalnya eksplorasi ruangan atau aplikasi semi-otonom. Di sisi lain, kebutuhan fleksibilitas kontrol juga menjadi perhatian. WebSocket memberikan jangkauan yang lebih luas namun membutuhkan mekanisme komunikasi yang lebih stabil dan real-time.

Dengan demikian, permasalahan yang ingin diselesaikan melalui proyek ini adalah bagaimana merancang sebuah robot self-balancing berbasis ESP32 yang:

- Mampu menjaga kestabilan menggunakan pembacaan sensor IMU dan kendali PID,
- Dapat merekam dan mengulangi jalur pergerakan,
- Mendukung kontrol jarak jauh dan dekat melalui WebSocket (WiFi),
- Serta mampu beroperasi secara real-time melalui pengelolaan multitasking menggunakan FreeRTOS.

1.2 PROPOSED SOLUTION

Solusi yang diusulkan untuk menyelesaikan permasalahan yang telah diidentifikasi adalah merancang sebuah self-balancing robot berbasis ESP32 yang mengintegrasikan sensor IMU, kendali PID, mekanisme penyimpanan jalur, serta

metode komunikasi (WiFi) di dalam satu sistem yang dikelola menggunakan FreeRTOS.

Robot ini menggunakan sensor IMU untuk membaca orientasi dan sudut kemiringan secara real-time. Data tersebut kemudian diproses oleh algoritma PID guna mengatur kecepatan dan arah putaran kedua motor sehingga robot dapat mempertahankan keseimbangannya. Untuk mendukung navigasi, robot dilengkapi kemampuan path memorization, yaitu mekanisme pencatatan pergerakan motor yang memungkinkan robot mengingat dan mengulangi jalur yang pernah dilalui.

Sistem komunikasi WebSocket (WiFi) digunakan untuk kontrol jarak jauh ataupun dekat. Modul ini berjalan bersamaan dalam arsitektur FreeRTOS, di mana masing-masing fungsi penting, pembacaan sensor, kontrol motor, komunikasi, dan path processing, dijalankan dalam task terpisah sehingga operasi robot tetap deterministik dan tidak saling mengganggu.

Dengan pendekatan tersebut, solusi ini tidak hanya memungkinkan robot untuk menjaga keseimbangan, tetapi juga memberikan fleksibilitas mode operasi dan kemampuan navigasi yang lebih cerdas, sekaligus memenuhi kebutuhan proyek untuk menerapkan konsep RTOS dan IoT secara terintegrasi.

1.3 ACCEPTANCE CRITERIA

Kriteria penerimaan dari proyek ini adalah sebagai berikut:

1. Sistem harus mampu menjaga keseimbangan robot secara stabil menggunakan sensor IMU dan kontrol PID.
2. Sistem harus mampu beroperasi dalam tiga mode: mode stabil, mode penyimpanan jalur, mode kontrol WebSocket.
3. Robot harus mampu merekam jalur pergerakan dan mengulang kembali jalur tersebut secara konsisten.
4. Sistem harus mampu menerima perintah kontrol melalui WebSocket menggunakan koneksi WiFi dengan respons yang baik.
5. Arsitektur berbasis FreeRTOS harus memastikan bahwa task pembacaan IMU, komputasi PID, dan pengendalian motor berjalan tepat waktu tanpa kegagalan.

6. Sistem harus memiliki input melalui sensor atau komunikasi, serta output melalui aktuator berupa motor driver.
7. Proyek harus menggunakan minimal enam modul praktikum sesuai dengan ketentuan yang diberikan.
8. Sistem harus dapat didemonstrasikan secara penuh menggunakan rangkaian fisik pada saat presentasi.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Programmer	Membuat rangkaian elektrik, membuat kerangka 3D, membuat kode, finalisasi laporan, membuat PPT, dan mengambil foto dokumentasi	Bryan Herdianto
Project Manager	Membuat laporan, membuat PPT, dan membuat dokumentasi proyek	Daffa Bagus Dhiananto
Project Manager	Membuat laporan, membuat PPT, menambahkan README, dan membuat dokumentasi proyek	Nabiel Harits Utomo
Project Manager	Membuat dokumentasi proyek	Tjokorde Gde Agung Abel Putra

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

No	Milestone	November							December						
		24	25	26	27	28	29	30	1	2	3	4	5	6	7
1	Hardware Design Completion														
1.1	Finalisasi design 3D														
1.2	Pemasangan komponen-komponen														
1.3	Pembuatan wiring semua komponen														
2	Software Development														
2.1	Pembuatan kode balancing robot														
2.2	Pembuatan kode WebSocket														
3	Hardware & Software Integration														
3.1	Tuning parameter-parameter PID														
3.2	Kontrol motor dari WebSocket														
4	Final Assembly & Testing														
4.1	Uji semua modul														
4.2	Pembuatan dokumentasi														

Table 2. Timeline Gantt Chart

Milestone	Description	Status
Hardware Design Completion	Finalisasi desain rangka, pemasangan motor driver, IMU, ESP32, baterai, serta wiring seluruh komponen hardware.	Completed
Software Development	Implementasi FreeRTOS tasks, pembacaan IMU, kontrol PID, kontrol motor, path memorization, serta integrasi komunikasi WebSocket.	Completed
Hardware & Software Integration	Sinkronisasi pembacaan IMU, PID balancing, dan kontrol motor dengan input dari WebSocket, serta pengujian integrasi awal seluruh sistem.	Completed
Final Assembly & Testing	Final tuning sistem balancing, debugging, uji operasional semua modul, dan penyusunan dokumentasi akhir.	Completed

Table 3. Timeline and Milestones

CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Hardware design pada sistem SaR-PaM (Self-Balancing Robot with Path Memorization) mencakup perancangan keseluruhan rangkaian elektronik yang digunakan untuk menjaga kestabilan robot, mengontrol motor, membaca sensor, serta menyediakan koneksi nirkabel. Desain hardware berfokus pada integrasi sensor IMU, driver motor, mikrokontroler, modul komunikasi, serta sistem catu daya. Berikut adalah list barang yang digunakan untuk proyek ini:

No	Nama Item	Jumlah
1	MPU6050 (6-Axis Accelerometer & Gyroscope)	1
2	L298N Motor Driver	1
3	Baterai Li-ion 3,7V	2
4	Case Battery Holder 2 Slot	1
5	DC Gearbox	2
6	Breadboard SYB-170 Mini	2
7	Wheel	2
8	Kabel Jumper Male-to-Male	20
9	ESP32	1
10	Metal Bracket	2
11	Mur 5mm	16
12	Tongkat Besi 5mm	1
13	Board Aluminium 30x12cm	1

Table 4. List Barang

Robot ini menggunakan sensor IMU (Inertial Measurement Unit) sebagai sumber data utama untuk pengendalian keseimbangan. IMU mengukur percepatan dan kecepatan sudut, kemudian data ini diolah oleh mikrokontroler untuk menjaga

posisi robot menggunakan algoritma PID. Dua motor DC dengan gearbox dihubungkan ke motor driver yang mampu memberikan arus cukup besar untuk menggerakkan robot secara stabil. Selain itu, SaR-PaM dilengkapi modul WiFi untuk komunikasi berbasis WebSocket. Integrasi modul ini memungkinkan robot untuk kendali jarak jauh dan pengiriman data secara real-time melalui WebSocket. Power dari sistem menggunakan baterai Li-ion yang terhubung untuk menjaga stabilitas tegangan ke motor dan mikrokontroler ESP32. Schematic dari projek ini dapat dilihat pada **Appendix A**.



Fig 1. Robot 3D Design

2.2 SOFTWARE DEVELOPMENT

Software untuk sistem SaR-PaM dikembangkan menggunakan Arduino IDE dengan dukungan library ESP32. Arsitektur perangkat lunak disusun menggunakan FreeRTOS, yang telah terintegrasi pada ESP32, sehingga setiap proses utama dapat dijalankan secara paralel. Pendekatan multitasking ini memastikan robot tetap responsif saat melakukan balancing, menerima perintah pengguna, dan menyimpan jalur pergerakan secara bersamaan. Terdapat beberapa modul utama dalam pengembangan software:

- **IMU Processing & PID Balancing Module**

Modul ini bertanggung jawab untuk:

- Membaca data gyro dan accelerometer dari IMU.
- Menghitung sudut kemiringan.
- Menjalankan algoritma PID untuk menjaga keseimbangan robot.

Proses ini berjalan dalam task real-time dengan software timer sehingga balancing tidak terganggu walaupun modul lain sedang aktif.

- **Path Memorization Module**

Modul ini melakukan fungsi-fungsi:

- Mencatat urutan instruksi (seperti FORWARD, LEFT, atau STOP) yang diterima dari WebSocket ke dalam variabel bertipe vector.
- Menggunakan software timer untuk mengeksekusi kembali urutan perintah yang tersimpan dengan interval waktu tertentu.

- **WebSocket Control Module**

Modul ini mengelola koneksi WiFi untuk remote-control jarak jauh ataupun dekat. Fungsinya meliputi:

- Menerima instruksi dari user melalui WebSocket.
- Menghubungkan ESP32 ke Access Point menggunakan SSID dan Password yang telah ditentukan.

- **Motor Control Module**

Modul ini menghasilkan sinyal PWM untuk mengatur:

- Kecepatan motor kiri dan kanan
- Arah rotasi

PWM dijalankan melalui software timer sehingga nilai output selalu stabil.

- **Website Remote Controller**

Modul ini merupakan interface yang dapat digunakan oleh user agar bisa mengontrol robot. Berikut fitur-fiturnya:

- Mengirimkan paket berformat JSON ke ESP32 menggunakan protokol WebSocket untuk menjamin pengiriman data dengan latensi rendah.
- Memantau status koneksi jaringan secara real-time dan memberikan indikator visual.
- Memproses user interaction dari klik mouse pada button forward, reverse, left, dan right, serta record dan play button.

Berikut adalah flowchart untuk program:

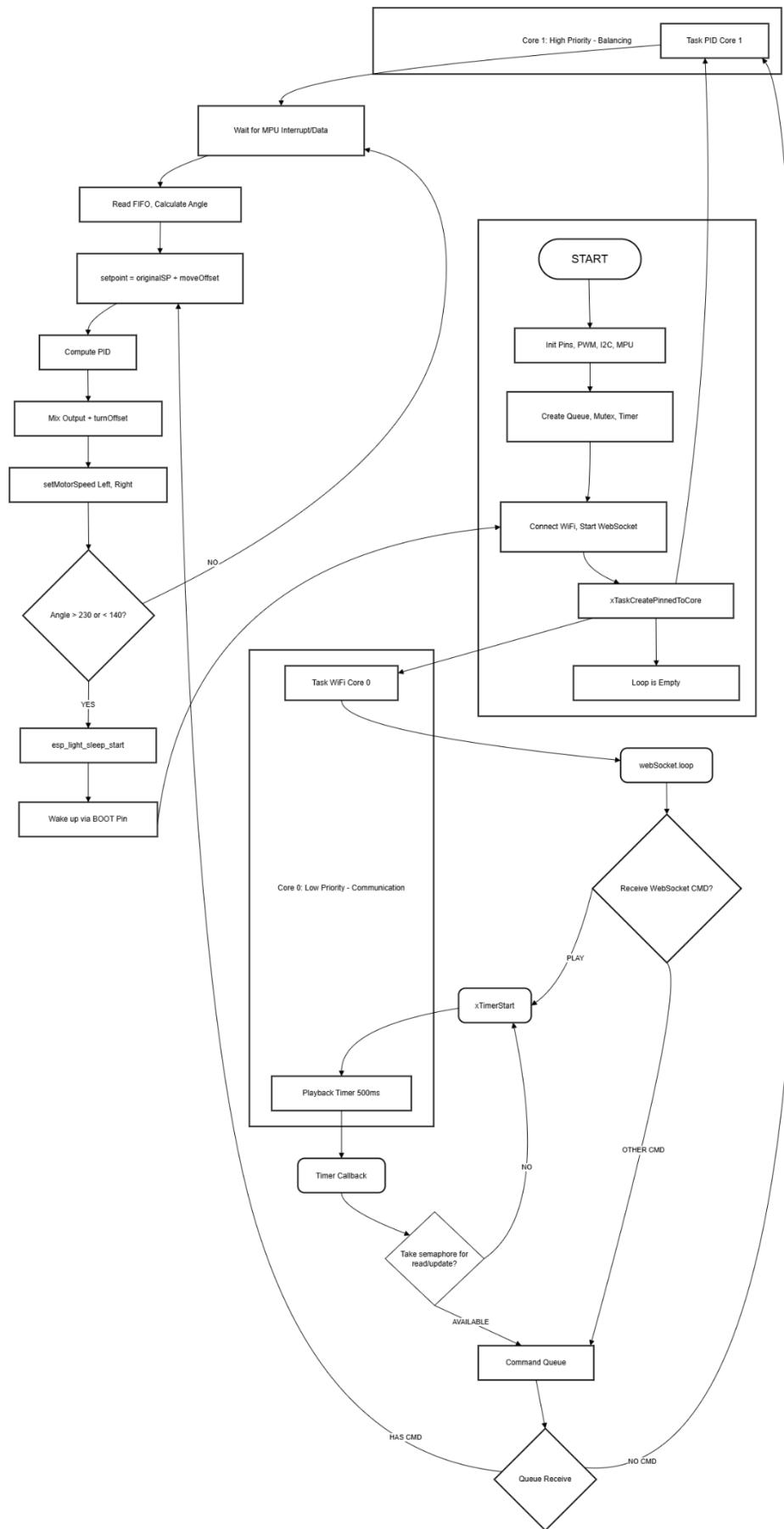


Fig 2. Flowchart Program

Dari flowchart di atas, sistem dimulai dengan inisialisasi hardware (pin, motor, MPU6050) dan objek RTOS (Queue, Mutex, Timer) sebelum membagi proses ke dalam dua core yang berjalan paralel. Pada Core 0 (prioritas rendah), Task WiFi bertugas menerima perintah dari user melalui WebSocket. Jika perintah tersebut berupa instruksi gerak, data dikirimkan ke Core 1 melalui Command Queue. Jika fitur Playback diaktifkan, Software Timer akan berjalan setiap 500ms untuk membaca record (yang dilindungi oleh Mutex agar aman) dan mengirimkan perintah tersebut ke queue yang sama. Sekaligus memastikan bahwa logika komunikasi tidak pernah memblokir proses balancing untuk robot.

Sementara itu, Core 1 (prioritas tinggi) menjalankan loop PID utama untuk menjaga keseimbangan robot. Pada setiap siklusnya, task ini mengecek apakah ada instruksi baru di dalam Queue untuk memperbarui setpoint gerak, lalu menunggu sinyal Interupsi dari sensor MPU6050 untuk memastikan data sudut yang dibaca selalu baru dan akurat. Setelah sudut dihitung dan output PID diterapkan ke motor, sistem melakukan pengecekan apakah robot terdeteksi jatuh selama lebih dari 10 detik. Jika iya, maka sistem akan mematikan motor dan masuk ke mode Light Sleep untuk menghemat daya hingga dibangunkan kembali secara manual.

Semua kode itu dibagi dalam beberapa file agar tetap modular. Berikut struktur file yang berisi kode utama untuk mengontrol robot self-balancing:

- **main.ino (file yang akan diupload ke ESP32)**
- **MotionControl.cpp**
- **MotionControl.h**
- **MotorControl.cpp**
- **MotorControl.h**
- **Network.cpp**
- **Network.h**
- **Shared.h**

2.3 HARDWARE AND SOFTWARE INTEGRATION

Langkah pertama yang harus dilakukan adalah kalibrasi sensor IMU dengan menggunakan kode util/mpu6050_calibration.ino. Proses kalibrasi ini bertujuan untuk menemukan nilai offset sensor yang akurat, yaitu: XGyroOffset, YGyroOffset,

ZGyroOffset, dan ZAccelOffset. Berikut gambar tampilan Arduino IDE setelah menemukan nilai offset yang tepat:



The screenshot shows the Arduino Serial Monitor window. At the top, it says "Message (Enter to send message to 'ESP32 Dev Module' on 'COM14')". Below that, there are two sets of sensor readings. The first set is labeled "Sensor readings INCLUDING offsets:" and the second is "Compare with IDEAL sensor readings:". Both sets show values for X, Y, and Z axes: X=0, Y=0, Z=16389 for the first and Z=16384 for the second. A dashed line separates these from the next section. Below the dashed line, it says "Your MPU-6050 offsets:" followed by C++ code for setting offsets: mpu.setXAccelOffset(-1109); mpu.setYAccelOffset(1920); mpu.setZAccelOffset(968); mpu.setXGyroOffset(-2); mpu.setYGyroOffset(74); mpu.setZGyroOffset(7); At the bottom, there is a note: "You can copy and paste the above offsets directly into your sketch. :)"

Fig 3. IMU Calibration

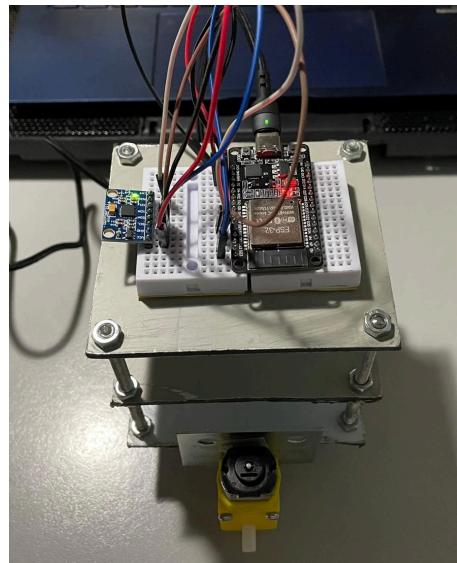


Fig 4. Robot Posture during IMU Calibration

Setelah kalibrasi IMU, barulah dilakukan Tuning Kontrol PID untuk menyesuaikan variabel Kp, Ki, dan Kd guna mengoptimalkan respons dan kestabilan robot saat bergerak atau menanggapi gangguan. Tahap tuning kontrol PID ini yang merupakan proses berdurasi lama karena kita perlu mengganti nilai pada kode secara perlahan dan mengupload pada ESP32 setiap kali. Akhirnya, nanti ketemu nilai berikut:

```
double Kp = 25.0;
double Kd = 1.2;
double Ki = 80.0;
```

Integrasi hardware dan software pada SaR-PaM dilakukan melalui sinkronisasi antara pembacaan sensor, pengendalian motor, dan komunikasi nirkabel secara simultan. Data dari IMU dibaca oleh task sensor, lalu dikirim melalui queue ke task

PID untuk menjaga keseimbangan. Task motor kemudian menerima output PID untuk menentukan kecepatan dan arah motor. Saat robot berjalan, data posisi dan kecepatan direkam untuk fitur path memorization, sehingga robot mampu mengulang jalur yang pernah dilewati. Pada mode kontrol jarak jauh, komunikasi WebSocket digunakan untuk mengirim perintah dan menerima data secara real-time tanpa mengganggu stabilitas robot. Dengan desain hardware yang terintegrasi dan sistem software multitasking yang terstruktur, SaR-PaM dapat menjalankan tiga mode operasinya secara efektif:

- Mode Stabil, menjaga robot tetap tegak.
- Mode Mengingat Jalur, menyimpan dan mengulang lintasan.
- Kontrol WebSocket, kendali berbasis WiFi.

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Pengujian dilakukan untuk memastikan bahwa seluruh fungsi sistem SaR-PaM berjalan sesuai dengan spesifikasi, terutama pada kemampuan robot untuk bergerak stabil, menerima perintah melalui WiFi, dan manajemen energi. Beberapa pengujian yang dilakukan meliputi:

- Pengujian Stabilitas Gerak Robot Dua Roda
 - Mengamati performa balancing saat robot bergerak maju dan mundur.
 - Menguji respon terhadap perubahan kecepatan mendadak ataupun kecil.
- Pengujian Path Memorization
 - Mampu membuat gerakan yang sudah direkam sebelumnya tanpa membuat robot jatuh.
 - Robot tanpa melakukan gerakan yang sama, tanpa ada perbedaan.
- Pengujian Kontrol WiFi
 - Mengirimkan perintah gerak (forward, backward, turn left, turn right) melalui WebSocket.
 - Menguji konsistensi perintah dalam jangkauan WiFi berbeda.
- Pengujian Power Management
 - Dapat sleep ketika memenuhi kondisi tertentu.

3.2 RESULT

Berikut adalah result dari pengujian projek kami.

No	Pengujian	Parameter	Status
1	Stabilitas robot roda dua	Kestabilan PID dan balancing tilt agar robot tidak jatuh	PASS
2	Kontrol WebSocket	Delay respon yang cepat dan	PASS

		akurasi perintah yang sesuai	
3	Power management	Pengurangan konsumsi daya setelah modul dimatikan	PASS
4	Path memorization	Jalur yang di save dapat dijalankan kembali oleh robot	PASS

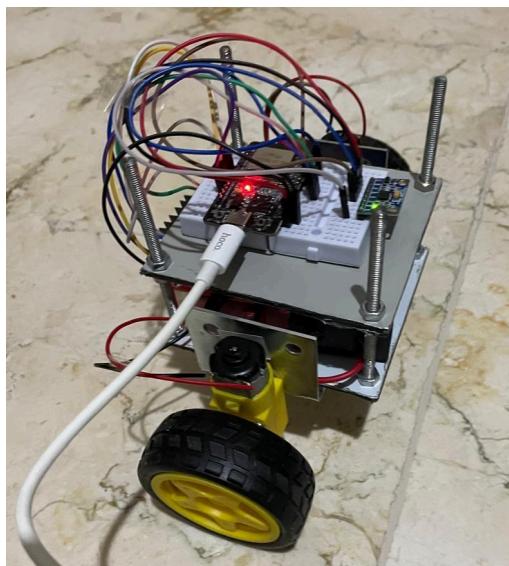


Fig 5. Testing Stabilitas Robot

STATUS: CONNECTED

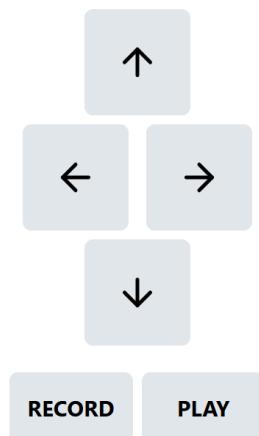


Fig 6. Tampilan Website Connected

STATUS: NOT CONNECTED

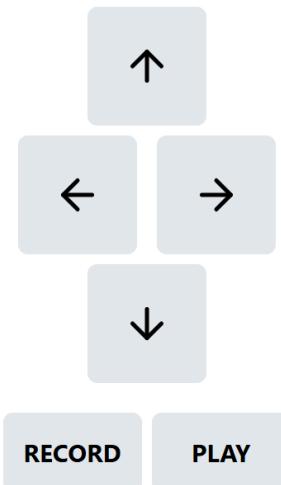


Fig 7. Tampilan Website Not Connected

```
E (11071) err: esp_err_t = -1
Entering Sleep...
Woke up!
E (16676) err: esp_err_t = -1
Entering Sleep...
```

Fig 8. Testing Light Sleep Mode

Gambar-gambar di atas menunjukkan proses testing SaR-PaM pada berbagai mode kontrol. Dalam foto tersebut terlihat robot sedang diuji untuk bergerak maju dan berbelok sambil menjaga stabilitas menggunakan balancing system. Pengujian juga melibatkan verifikasi apakah robot tetap stabil saat perintah dikirim melalui WebSocket. Demo video dari self-balancing robot dapat dilihat di **Appendix B**.

3.3 EVALUATION

Berdasarkan hasil testing, evaluasi utama yang dapat diberikan adalah:

- **Robot stabil meskipun menggunakan dua roda**

Sistem balancing berbasis IMU dan PID mampu menjaga robot tetap tegak saat bergerak. Robot dapat mempertahankan stabilitas meskipun terdapat perubahan kecepatan maupun manuver belok.

- **Robot dapat dikontrol melalui WiFi**

Kontrol melalui WebSocket berjalan dengan baik. Delay relatif kecil dan robot merespons perintah secara real-time. Mode ini efektif untuk kontrol jarak jauh karena jangkauan WiFi lebih luas.

- **Robot dapat melakukan sleep**

Sistem secara real-time memantau sudut kemiringan robot melalui sensor MPU6050. Jika sudut kemiringan melewati batas tertentu, sistem menganggap robot telah terjatuh. Jika robot dibiarkan dalam kondisi terjatuh selama durasi tertentu, ESP32 secara otomatis masuk mode Light Sleep. Dalam mode ini, CPU dihentikan sementara dan koneksi WiFi diputus untuk menghemat baterai, tetapi RAM tetap aktif. Robot dapat dibangunkan kembali melalui interupsi eksternal (menekan tombol BOOT).

CHAPTER 4

CONCLUSION

Proyek pengembangan platform kontrol robot berbasis ESP32 ini berhasil membangun sebuah sistem robot self-balancing dua roda yang dikendalikan melalui komunikasi WiFi, dengan fokus pada efisiensi energi dan stabilitas gerak. Keberhasilan utama terletak pada kemampuan robot untuk mencapai stabilitas gerak yang konsisten meskipun menggunakan konfigurasi dua roda sederhana. Implementasi pengendalian motor yang cermat, mengombinasikan PWM (Pulse Width Modulation) dengan tuning kecepatan yang tepat, memungkinkan robot mempertahankan arah gerak yang konsisten. Hasilnya, robot menunjukkan kemampuan manuver yang stabil dan terarah, baik saat bergerak maju, mundur, maupun ketika melakukan manuver belok.

Aspek komunikasi dan fungsionalitas jarak jauh sistem ini juga mencapai hasil yang memuaskan. Robot berhasil mendukung kontrol jarak jauh melalui koneksi WiFi berbasis WebSocket, yang menjamin komunikasi real-time dan latency rendah. Kemampuan robot untuk secara andal menerima perintah dari frontend menegaskan efektivitas sistem kontrol nirkabel yang dibangun. Implementasi multitasking menggunakan sistem operasi ringan FreeRTOS terbukti sangat krusial, karena mampu menjaga responsivitas robot tetap tinggi dan stabil, meskipun beberapa modul kritis (seperti PID balancing dan komunikasi WiFi) berjalan secara paralel.

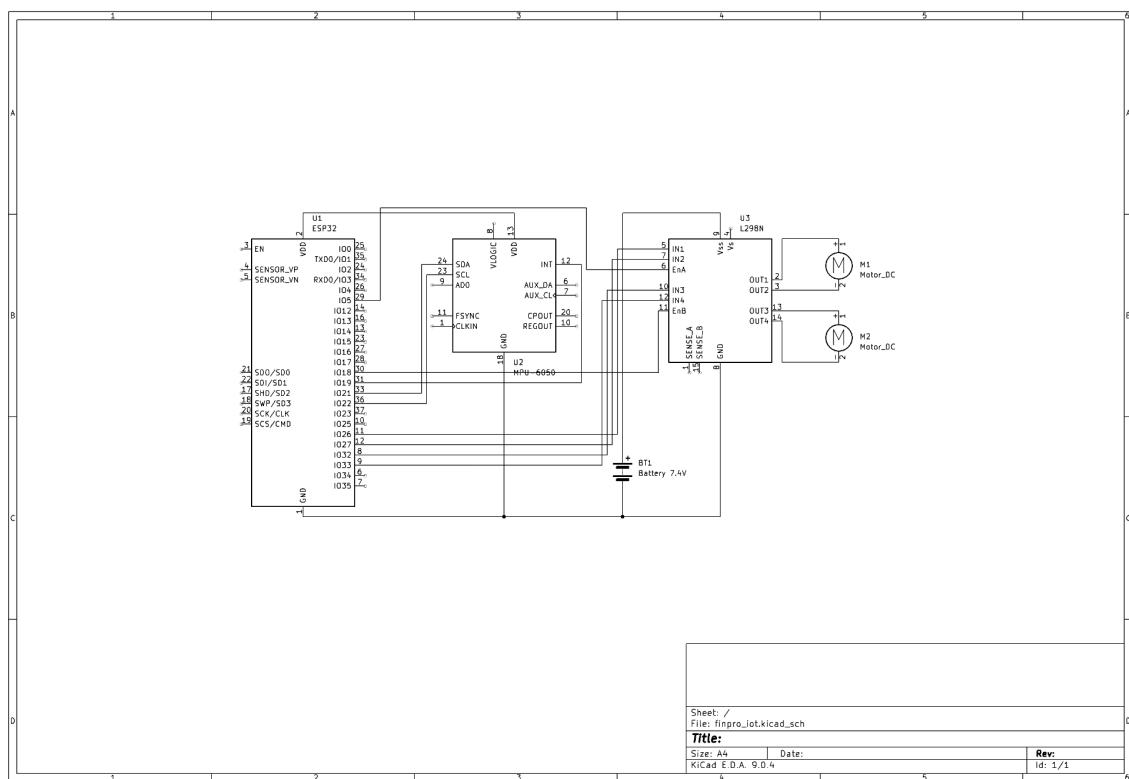
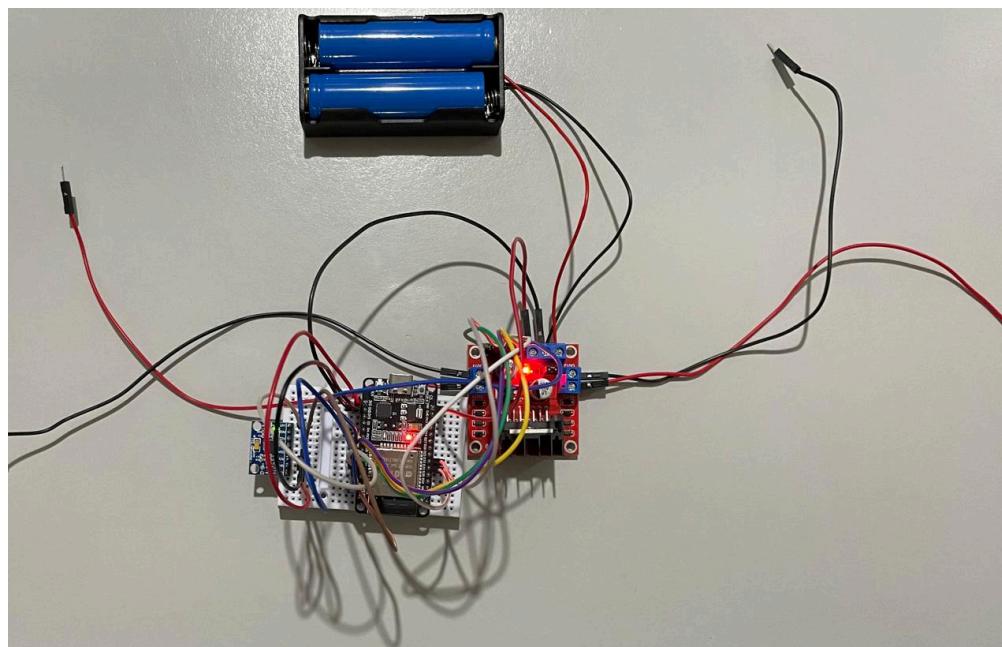
Secara keseluruhan, sistem yang telah dikembangkan ini dinyatakan berhasil karena telah memenuhi seluruh acceptance criteria yang ditetapkan. Untuk pengembangan lebih lanjut, beberapa rekomendasi teknis telah diidentifikasi guna meningkatkan performa dan kepraktisan robot. Peningkatan tersebut mencakup penggantian motor menjadi stepper motor untuk torsi yang lebih baik, integrasi modul charger baterai TP4056 dan baterai polymaire untuk manajemen daya yang efisien, penambahan switch on/off, serta perancangan PCB (Printed Circuit Board) khusus untuk kerapian instalasi dan distribusi beban (weight distribution) yang lebih merata, yang akan semakin mengoptimalkan kestabilan robot.

REFERENCES

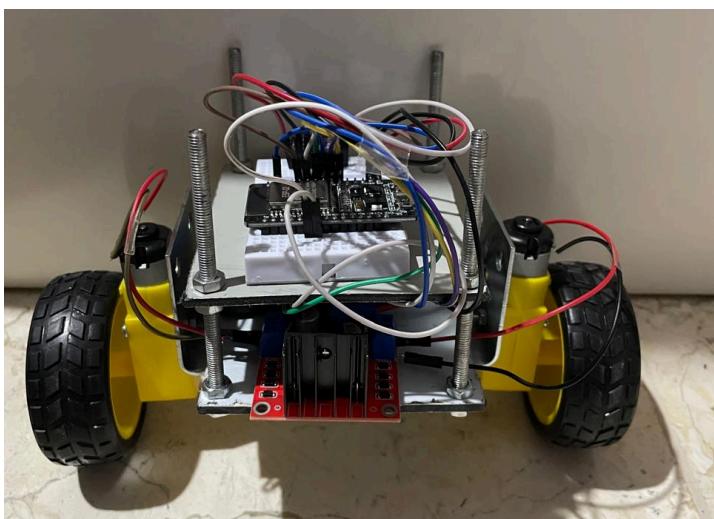
- [1] "Module 1 - Introduction to SMP with RTOS," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-1-introduction-to-smp-with-rtos> (Accessed: Dec. 7, 2025).
- [2] "Module 2 - Task Management," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-2-task-management> (Accessed: Dec. 7, 2025).
- [3] "Module 3 - Memory Management & Queue," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-3-memory-management-queue> (Accessed: Dec. 7, 2025).
- [4] "Module 4 - Deadlock & Synchronization," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-4-deadlock-synchronization> (Accessed: Dec. 7, 2025).
- [5] "Module 5 - Software Timer," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-5-software-timer> (Accessed: Dec. 7, 2025).
- [6] "Module 7 - MQTT, HTTP, WIFI," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-7-mqtt-http-wifi> (Accessed: Dec. 7, 2025).
- [7] "Module 8 - Power Management," Digilab DTE. [Online]. Available: <https://learn.digilabdte.com/books/internet-of-things/chapter/module-8-power-management> (Accessed: Dec. 7, 2025).
- [8] Instructables, "DIY ESP32 Wifi Self Balancing Robot - B-Robot ESP32 Arduino Programming," Instructables, Sep. 19, 2021. [Online]. Available: <https://www.instructables.com/DIY-ESP32-Wifi-Self-Balancing-Robot-B-Robot-ESP32/> (Accessed: Dec. 7, 2025).

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation



LINK YT: <https://youtube.com/shorts/Nn4jgWTpkQI>