RANCANG BANGUN SISTEM PEMANTAUAN KECEPATAN KENDARAAN PADA SUATU KAWASAN MENGGUNAKAN MIKROKONTROLER ESP32

TUGAS AKHIR

Untuk Memenuhi Persyaratan Mencapai Pendidikan

Diploma III (D3)

Disusun Oleh:

SHEVA AISHA PUTRI

40040518060002

PROGRAM STUDI DIII INSTUMENTASI DAN ELEKTRONIKA

SEKOLAH VOKASI

UNIVERSITAS DIPONEGORO

SEMARANG

2022

HALAMAN PENGESAHAN
TUGAS AKHIR

RANCANG BANGUN SISTEM PEMANTAUAN KECEPATAN KENDARAAN PADA SUATU KAWASAN MENGGUNAKAN MIKROKONTROLER ESP32

Disusun Oleh:

SHEVA AISHA PUTRI
40040518060002

Telah diujikan dan dinyatakan lulus oleh Tim Penguji
Pada Tanggal : 29 Maret 2022

Dosen Pembimbing

Jatmiko Endro Suseno, S.Si., M.Si., Ph.D.
NIP. 197211211998021001

Tim Penguji,

Penguji I	Penguji II
Dr. Eng. Agus Setyawan, S.Si., M.Si.	**Fajar Arianto, S.Si., M.Si.**
NIP. 197308251999031002	**NIP. 198608012021041001**

Tugas Akhir ini telah diterima sebagai salah satu persyaratan untuk memperoleh gelar Ahli Madya (A.Md)

Semarang, 25 Maret 2022

Mengetahui,
Ketua Program Studi

Dr. Priyono, M.Si.
NIP. 196703111993031005

HALAMAN PERSETUJUAN

Judul Tugas Akhir	: Rancang Bangun Sistem Pemantauan Kecepatan Kendaraan Pada Suatu Kawasan Menggunakan Mikrokontroler ESP32
Nama	: Sheva Aisha Putri
NIM	: 40040518060002

Tugas akhir ini telah selesai dan layak untuk mengikuti ujian tugas akhir di Program Studi D3 Instrumentasi dan Elektronika, Departemen Teknologi Industri, Sekolah Vokasi, Universitas Diponegoro.

Semarang, Maret 2022

Menyetujui,

Dosen Pembimbing Tugas Akhir

Jatmiko Endro Suseno, S.Si., M.Si., Ph.D.

NIP. 197211211998021001

MOTTO HIDUP DAN PERSEMBAHAN

MOTTO HIDUP

The only time you should ever look back, is to see how far you've come.

PERSEMBAHAN

Tugas Akhir ini saya persembahkan kepada:

- Diri saya sendiri yang telah teguh berjuang dan tidak pernah menyerah.
- Orang tua, adik saya tercinta Afifah Noviza dan keluarga besar yang selalu mendukung dan mendoakan dengan baik sampai bisa seperti sekarang ini.
- Alm. Prof. Dr. Suryono, S.Si, M.Si. dan Bapak Jatmiko Endro Suseno, S.Si., M.Si., Ph.D. sebagai dosen pembimbing.
- Seluruh dosen Instrumentasi dan Elektronika.
- Nurul Dzakiyyah, Jihan Rizki Hanifah, M Ibnu Haqi, Guntur Hidayat, Dzakia Ulfa, Humairah Andra, Muhammad Aqzel Maulana, Monica Suryatama Deby, Aditia Yodi Saputra, Haris Van Ananda, dan Evan Kurniawan yang telah menemani, mendukung dan memberi semangat dari sejak masa sekolah hingga saat ini.
- Dinda Widhi Aqiilah, Nadia Ananda Miranti, dan Fitri Rahmadhani selaku teman yang selalu mendukung saya.
- Keluarga Instrumentasi dan Elektronika 2018 yang telah membantu saya sejak awal masa kuliah hingga akhir.
- Almamaterku.

KATA PENGANTAR

Puji syukur senantiasa penulis panjatkan atas kehadirat Allah SWT karena atas berkat dan rahmat-Nya penulis dapat menyusun dan menyelesaikan tugas akhir ini dengan baik. Tugas akhir dengan judul **"Rancang Bangun Sistem Pemantauan Kecepatan Kendaraan pada Suatu Kawasan Menggunakan Mikrokontroler ESP32"** ini diajukan guna memenuhi persyaratan kelulusan pendidikan tingkat Diploma 3 pada Program Studi DIII Instrumentasi dan Elektronika, Sekolah Vokasi, Universitas Diponegoro. Hasil penulisan ini diharapkan dapat memberikan manfaat utama dalam pengembangan teknologi dibidang instrumentasi.

Terselesaikannya tugas akhir ini tidak lepas dari dukungan dan bantuan berbagai pihak. Pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Bapak Dr. Priyono, M.Si. selaku Ketua Program Studi DIII Instrumentasi dan Elektronika, Sekolah Vokasi, Universitas Diponegoro.
2. Bapak Prof. Dr. Suryono, S.Si., M. Si. (Alm.) dan Bapak Jatmiko Endro Suseno, S.Si., M.Si., Ph.D. selaku dosen pembimbing tugas akhir yang telah meluangkan waktunya untuk memberikan bimbingan, pengarahan, dan petunjuk dalam penyusunan laporan ini.
3. Bapak Drs. K. Sofjan Firdausi, M.Sc. sebagai koordinator tugas akhir.
4. Dosen Program Studi DIII Instrumentasi dan Elektronika yang telah memberikan banyak ilmu serta pengetahuan kepada penulis selama masa perkuliahan.
5. Kedua orang tua beserta keluarga besar yang telah memberikan semangat serta doa.
6. Semua pihak yang tidak mungkin dapat disebutkan satu persatu yang telah membantu sehingga terselesaikannya laporan tugas akhir ini.

Penulis menyadari bahwa penulisan laporan tugas akhir ini masih jauh dari kata sempurna dikarenakan keterbatasan ilmu, pengalaman, dan kemampuan. Oleh karena itu kritik dan saran yang membangun dari pembaca akan sangat berguna bagi penulis. Semoga laporan tugas akhir ini dapat bermanfaat bagi pembaca.

Semarang, Maret 2022

Penulis

Sheva Aisha Putri

NIM. 40040518060002

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Sistem pengukuran dan pemantauan kecepatan kendaraan ini merupakan sistem yang memanfaatkan pancaran sinyal. Mikrokontroller ESP32 sebagai *client* akan memancarkan data secara terus menerus dan kemudian akan ditangkap oleh *router*. Sinyal yang ditangkap oleh *router* kemudian akan diteruskan pada komputer *server* menggunakan protokol MQTT dan kemudian akan ditampilkan pada sebuah *web localhost*. Sistem ini merupakan sistem nikabel yang dapat digunakan pada jarak yang cukup jauh. Uji coba pada sistem ini dilakukan pada konektivitas *server* dan *client* serta pengukuran kecepatan. *Server* dan *client* dapat terkoneksi pada jarak maksimum 108 meter, terhitung 54 meter dari kanan dan kiri *router*. Pengukuran nilai kecepatan juga berhasil dilakukan dengan rata-rata nilai kesalahan sebesar 9.42%.

Kata Kunci: Mikrokontroler ESP32, *router*, sinyal, MQTT, kecepatan.

ABSTRACT

This vehicle speed measurement and monitoring system is a system that utilizes signal beams. The ESP32 microcontroller as a client will emit data continuously and capture it by the router. The signal captured by the router will then be passed on to the computer as the server using the MQTT protocol and displayed on a localhost web. This system is a non-wired system that can use at a considerable distance. Trials on this system are conducted on server and client connectivity and speed measurements. The server and client can be connected at a maximum distance of 108 meters, counting 54 meters from the right and left of the router. Speed value measurement was also successfully carried out with an average error value of 9.42%.

Keywords: ESP32 Microcontroller, router, signal, MQTT, velocity.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Salah satu faktor penyebab terjadinya kecelakaan berlalulintas adalah tidak sesuainya kecepatan kendaraan yang dikendarai dengan peraturan yang ditetapkan. Dalam hal ini adalah kecepatan kendaraan yang terlalu tinggi. Semakin tinggi kecepatan kendaraan yang dikendarai, maka semakin tinggi pula resiko kecelakaan yang akan terjadi. Berdasarkan data kepolisian, tingkat jumlah kecelakaan pada tahun 2016 yang dikarenakan banyaknya jumlah pelanggar kecepatan kendaraan mencapai 105.000 jiwa. Sesuai dengan data *Nationaal Highway Traffic Savety Administration*, lebih dari 30% kecelakaan lalu lintas terjadi berkaitan dengan faktor kecepatan yang menyebabkan angka kematian dan *social cost* yang tinggi (NHTSA, 2005).

Jika dilihat dari kinerja lalu lintas, kecepatan lalu lintas tinggi berarti mobilitas yang tinggi. Bila dilihat dari aspek keselamatan lalu lintas, kecepatan yang tinggi berarti resiko kecelakaan yang tinggi pula (Mauliza, dkk., 2019). Untuk menekan angka kecelakaan akibat pengendara yang sering melanggar lalu lintas melebihi kecepatan aman, maka diperlukan pengawasan terhadap pengendara untuk memberikan peringatan. Maka dibutuhkan alat yang dapat mengukur dan mengawasi kecepatan kendaraan yang melintas, agar para pelanggar lalu lintas dapat ditindaklanjuti oleh pihak yang berwenang.

Telah banyak dilakukan penelitian menggunakan metode dan sensor yang berbeda dalam mengukur kecepatan kendaraan bermotor. Beberapa diantaranya sistem pengukur kecepatan kendaraan menggunakan sensor magnetik, sensor infra merah, sensor ultrasonik, bahkan berbasis pengolahan video. Namun, terdapat kekurangan dari masing masing metode percobaan. Pada pengukur kecepatan menggunakan sensor magnetik dan infra merah, pengujian hanya dapat dilakukan pada jarak yang terbatas. Lalu pada pengukur menggunakan sensor ultrasonik, selain jarak yang terbatas percobaan juga dipengaruhi oleh kecepatan angin.

Sedangkan pada sistem pengukur kecepatan berbasis pengolahan video, sistem dapat bekerja dengan sangat baik pada siang hari dengan rentang cahaya 600-1900 lux. Namun pada saat malam hari dengan rentang cahaya yang hanya mencapai 5 lux, sistem tidak dapat bekerja dengan baik.

Dari permasalahan latar belakang tersebut, maka peneliti merancang alat yang dapat digunakan, dan diharapkan dapat meminimalisir kekurangan dalam mengukur dan mengawasi kecepatan kendaraan guna menekan angka kecelakaan yang dapat terjadi menggunakan Mikrokontroller ESP32 dan *router wifi.*

1.2. Tujuan

Merancang dan merealisasikan pemantauan kecepatan kendaraan pada jalan raya menggunakan Mikrokontroller ESP32 yang kemudian akan ditampilkan dalam *web localhost*, mengoneksikan *client* dengan *access point* serta *server* pada jarak maksimum jangkauan *router*, dan meminimalisir nilai eror pada alat.

1.3. Manfaat

Manfaat yang diharapkan dari penelitian ini antara lain:

1. Memberi peringatan dini kepada pengguna jalan agar tidak melanggar aturan tentang batas kecepatan kendaraan.
2. Mengurangi angka kecelakaan pada pengguna jalan yang disebabkan oleh pelanggaran terhadap batas kecepatan kendaraan.

BAB II
LANDASAN TEORI

2.1. Kecepatan

Kecepatan merupakan kemampuan bergerak secara berturut-turut untuk menempuh suatu jarak dalam satu satuan waktu (Sinaulan dkk., 2015). Pada suatu jarak tempuh yang sama, saat waktu tempuh semakin singkat maka nilai kecepatan pun semakin tinggi. Kecepatan termasuk besaran vektor, yang mana memiliki nilai arah. Hal ini menunjukkan seberapa cepat suatu benda berpindah. Besar dari vektor ini disebut dengan kelajuan dan dinyatakan dalam satuan meter per sekon (m/s atau ms^{-1}) . Kecepatan biasa digunakan untuk merujuk kecepatan sesaat yang dapat didefinisikan secara matematis sebagai berikut:

$$V = \lim_{\Delta t \to 0} \frac{r(t-\Delta t)-r(t)}{\Delta t} = \frac{dr}{dt} \quad \text{.....(2.1)}$$

dengan V adalah kecepatan sesaat dalam satuan meter per sekon (m/s) dan r(t) adalah perpindahan fungsi waktu dalam satuan sekon (s).

Selain kecepatan sesaat, dikenal juga besaran kecepatan rata rata ($\bar{v}$) yang didefinisikan dalam rentang waktu (Δt) yang tidak mendekati nol.

$$\bar{v} = \frac{\Delta r}{\Delta t} \quad \text{.....(2.2)}$$

Kecepatan rata-rata adalah panjang lintasan total yang ditempuh tiap satuan waktu keseluruhan dimana limit kecepatan rata-rata ketika selang waktu mendekati nol. Sedangkan kecepatan sesaat adalah kecepatan benda tiap saat tertentu.

Gerak lurus beraturan diartikan sebagai gerak suatu benda dengan kecepatan tetap. Kecepatan tetap disini berarti baik besar ataupun arahnya tetap, sehingga dapat disebut juga dengan kelajuan. Dengan demikian, gerak lurus beraturan dapat didefinisikan sebagai gerak suatu benda pada lintasan lurus dengan kelajuan dan nilai percepatan yang sama dengan nol. Gerak lurus beraturan berbentuk linear dengan nilai kecepatan yang didapat dari hasil bagi jarak dengan waktu yang ditempuh yang secara matematis dapat dirumuskan sebagai berikut:

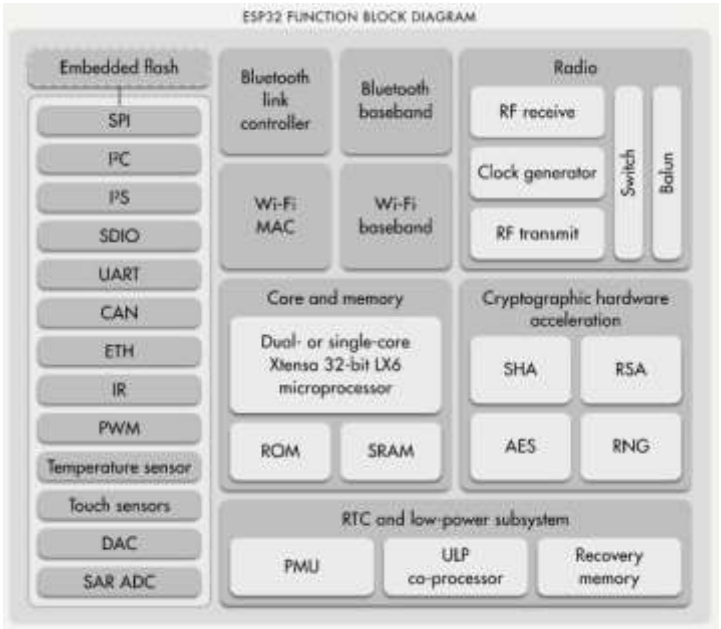
$$V = \frac{s}{t} \text{.....(2.3)}$$

dengan V adalah kecepatan dalam satuan meter per sekon (m/s), s adalah jarak yang ditempuh dalam satuan meter (m), dan t adalah watu tempuh dalam satuan sekon (s).

2.2. Mikrokontroler ESP32

ESP32 merupakan nama dari sebuah mikrokontroller yang dirancang oleh sebuah perusahaan asal Shanghai, Espressif Systems. ESP32 dapat memancarkan jaringan WiFi mandiri sebagai penghubung dari mikrokontroller yang telah ada ke jaringan WiFi. ESP32 menggunakan prosesor *dual core* yang berjalan di instruksi Xtensa LX16 (Kusumah dan Restu, 2019).

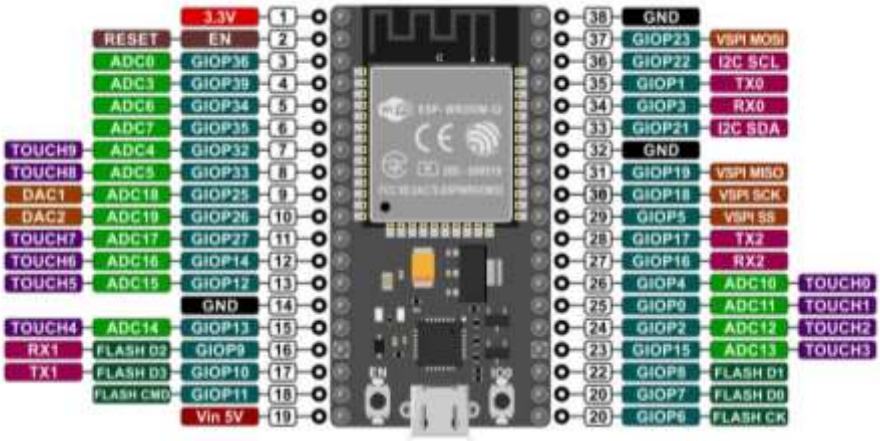
ESP32 dilengkapi dengan WiFi 2.4 GHz dan Bluetooth dengan daya yang rendah. Berdasarkan blok diagram pada gambar 2.1, chip ESP32 terdiri dari microprosessor Xtensa 32 bit, *cryptographic hardware acceleration*, DAC, ADC, CAN, SPI, dan I2C. Blok diagram chip ESP32 dapat dilihat pada gambar 2.1.



Gambar 2.1 Blok Diagram Chip ESP32 (Jatmiko dan Salita, 2019)

Fitur-fitur yang dimiliki oleh ESP32 dapat dijabarkan sebagai berikut; 18 buah kanal ADC (*Analog Digital Converter*), tiga antarmuka SPI, tiga antarmuka UART, dua antarmuka I2C, 16 kanal *output* PWM, dua kanal DAC (*Digital Analog Converter*), dua antarmuka I2S, dan 10 GPID sensor kapasitif.

Mikrokontroller ini juga dapat terhubung dengan internet, yang kemudian dapat menjadi pusat sistem *Internet of Things*. Mikrokontroller ini memiliki tegangan 3,3 V dalam operasinya, sehingga untuk perangkat input dan output yang beroperasi pada tegangan 5 V harus dilakukan konversi logika. ESP32 memiliki 12 bit masukan ADC berjumlah 12 masukan (Zulqarnain dkk.,2020). Fungsi dari masing-masing pin dapat dilihat pada Gambar 2.2 berikut.

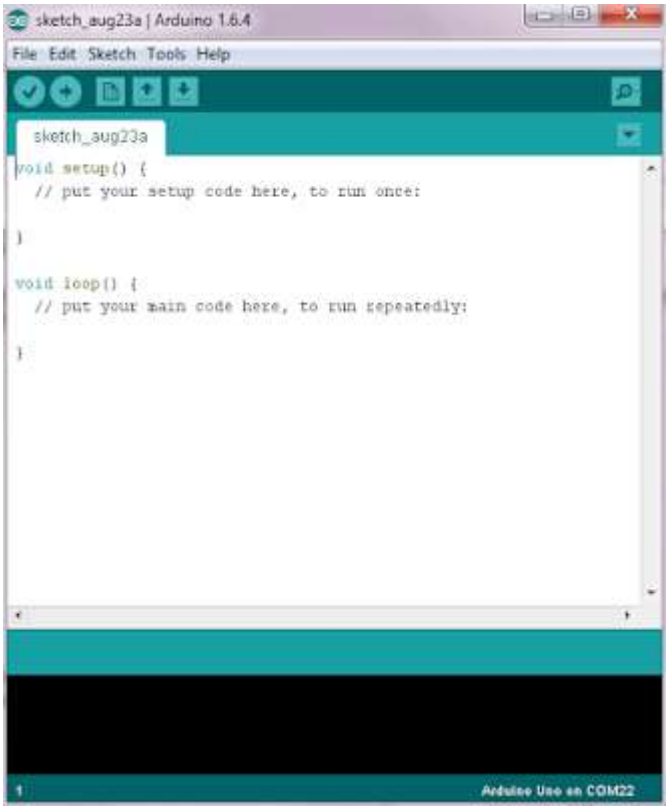


Gambar 2.2 Konfigurasi Pin ESP32 (Zulqarnain dkk., 2020)

Dari beberapa pin yang terlihat pada gambar, beberapa pin hanya dapat digunakan sebagai masukan. Pin-pin berikut tidak memiliki *pull up* ataupun *pull down* resistor internal, yaitu pin GPIO34, GPIO35, GPIO36, dan GPIO39. Mikrokontroller ini juga memiliki sensor sentuh kapasitif dimana sensor ini dapat merasakan benda bermuatan listrik. Apabila pin-pin tersebut disentuh maka perangkat akan mendeteksi induksi (Budjianto dan Winardi, 2021).

2.3. Arduino IDE

Software Arduino *Integrated Development Environment* (IDE) adalah suatu program khusus pada komputer agar dapat membuat sebuah rancangan program untuk Arduino (Aisyah, 2019). Software ini terdiri atas *editor program, verify, uploader, new, open, save,* serta *serial monitor*. Aplikasi ini dapat digunakan setelah instalasi pada komputer terlebih dahulu. Tampilan software Arduino IDE dapat dilihat pada Gambar 2.3.



Gambar 2.3 Tampilan Software Arduino IDE (kelasrobot.com, 2018)

Editor program berfungsi sebagai sebuah jendela yang memungkinkan pengguna menulis dan mengubah program dalam bahasa *processing*. *Verify* berfungsi untuk mengecek kembali kode *sketch* yang *error* sebelum melakukan *upload* ke *board* Arduino. *Uploader* yaitu sebuah modul yang memuat kode biner dari komputer ke dalam memori *board* Arduino. *New* digunakan untuk membuat *sketch* program baru. *Open* digunakan untuk membuka daftar *sketch* program yang telah dibuat sebelumnya pada *sketchbook* Arduino. *Save* digunakan ketika akan menyimpan sketch program yang telah dibuat ke *sketchbook* Arduino. Lalu *serial monitor* digunakan untuk menampilkan data serial dari *board* Arduino (Aisyah, 2019).

2.4. *Router Wi-Fi*

Router adalah perangkat yang melewatkan paket IP dari suatu jaringan menuju jaringan yang lain dengan menggunakan metode *addressing* dan protokol tertentu. *Router* yang terhubung dalam jaringan tersambung dalam suatu algoritma *routing* untuk menentukan jalur yang paling baik yang dilalui paket IP (Sofana, 2012). Proses *routing* dilakukan secara *hop by hop*. IP hanya *routing*

menyediakan *IP Address* dari *router* berikutnya yang lebih dekat dengan *host* tujuan (Herlambang dan Azis, 2008).

Fungsi utama *router* adalah membaca alamat logika atau *IP Address source* dan tujuan untuk menentukan *routing* dari suatu LAN ke LAN lainnya. Router bekerja menyimpan *routing table* untuk menentukan rute terbaik antara LAN ke WAN. *Router* juga berperan sebagai perangkat *layer* ketiga dalam *Open Systems Interconnection* (OSI) *Layer* (Samsumar dan Sofian, 2018). Bentuk fisik dari *Router* WiFi TP Link TL WR 845W dapat dilihat pada gambar 2.4.

Gambar 2.4 *Router* WiFi TP Link (tp-link.com, 2017)

Router WiFi TP Link TL WR 245W memiliki empat buah *port switch*, dan sudah menggunakan teknologi MIMO (*Multi Input Multi Output*) yang mana secara bersamaan dapat bekerja melalui tiga buah antena dalam mengirim (Tx) dan menerima (Rx) untuk mengatasi interferensi dan degradasi ketika melalui jarak yang jauh ataupun ketika melewati penghalang fisik. *Router* ini menggunakan *interface* 4 x 10/100 Mbps LAN *ports* serta 1 x 1/100 Mbps WAN *port*, dengan tombol *reset*, WiFi/WPS, dan *power on/off*. *Router* ini juga menggunakan tiga buah 5 dBi *fixed Omni Directional Antenna*, dengan *power supply* eksternal 9 V DC/0.6 A, dan *wireless standards* IEEE 802.11N, IEEE 802.11g, dan IEEE 802.11b.

2.5. Node.JS

Node.js adalah sistem perangkat lunak yang digunakan untuk pengembangan aplikasi web. Aplikasi ini ditulis dengan bahasa JavaScript, menggunakan basis *event* dan *asynchrounous* I/O. Berbeda dengan kebanyakan bahasa JavaScript yang digunakan pada peramban, Node.js digunakan sebagai sebuah aplikasi *server* (Iqbal dkk., 2012). Node.js dapat berjalan sebagai *server* karena dukungan mesin V8 Google dan beberapa modul *built-in* yang terintegrasi, seperti modul http, modul sistem *file*, modul keamanan dan beberapa modul lainnya (Fajrin, 2017).

Menurut Lutvianto (2019) beberapa kelebihan yang dimiliki node.js antara lain:

1. Node.js menggunakan bahasa pemrograman JavaScript yang dianggap sebagai bahasa pemrograman paling populer.
2. Node.js mampu menangani ribuan koneksi bersamaan dengan penggunaan *resource* minimum pada setiap prosesnya.
3. Node.js diandalkan dalam pembuatan aplikasi *realtime*.
4. Node.js adalah project *open source*, memudahkan siapapun dalam melihat struktur kode dan pengembangannya.
5. Penggunaan JavaScript disisi *server* dan *client* meminimalisir ketidakcocokan antara dua sisi lingkungan pemrograman
6. *Database* NoSQL seperti MongoDB dan CouchDB mendukung langsung JavaScript sehingga *interfacing* dengan *database* ini jauh lebih mudah.
7. Node.js menggunakan V8 yang mengikuti perkembangan standar ECMAScript, sehingga tidak ada kekhawatiran bahwa *browser* tidak mendukung fitur-fitur node.js.

2.6. Message Queue Telemetry Transport (MQTT)

Protokol Message Queue Telemetry Transport (MQTT) adalah protokol pesan yang cukup sederhana, menggunakan arsitektur *publish/subscribe* yang dirancang secara terbuka dan mudah untuk diimplementasikan (Budioko, 2016). MQTT mampu menangani ribuan *client* pada jarak yang jauh dengan satu server.

MQTT meminimalisir *bandwith* jaringan dan kebutuhan sumber daya perangkat saat mencoba menjamin pengiriman. Pendekatan ini menjadikan protokol MQTT cocok untuk menghubungkan mesin ke mesin, dan menjadikan MQTT aspek penting dalam konsep *Internet of Things* (Budioko, 2016).

Protokol MQTT merupakan protokol ringan yang digunakan diatas protokol TCP/IP. Protokol ini memiliki ukuran paket data *low overhead* yang kecil (minimal 2 *gigabyte*) dan konsumsi catudaya yang kecil (Saputra dkk., 2017). *Server* serta *broker* MQTT menggunakan *software* Mosquitto yang dapat dijalankan pada sistem operasi Windows maupun Linux. Mosquitto merupakan *broker* pesan yang bersifat *open source* (EPL/EDL Berlisensi) yang mengimplementasikan protokol MQTT versi 3.1 dan 3.1.1. MQTT menyediakan metode yang ringan untuk bisa mempublikasikan pesan sesuai dengan topik yang diinginkan. Hal ini membuat Mosquitto cocok dalam pengaplikasian *Internet of Things* dengan daya rendah ataupun perangkat *mobile* seperti ponsel, sistem tertanam, atau mikrokontroller seperti Arduino. *Server* MQTT secara *default* menggunakan *port* 1833 walaupun dalam penggunaannya *server* MQTT dapat menggunakan beberapa *port* lain dengan fungsi yang berbeda.

BAB III
PERANCANGAN DAN REALISASI SISTEM

3.1. Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan pada bulan Maret 2021 – Desember 2021 di Kost Aji Tembalang, Gedung Pascasarjana Lt.5 ddan Workshop Instrumentasi dan Elektronika, Departemen Fisika, Fakultas Sains dan Matematika, Universitas Diponegoro.

3.2. Alat dan Bahan

Pada penelitian ini digunakan beberapa alat dan bahan untuk pembuatan dan realisasi sistem ditunjukan pada tabel 3.1 dan 3.2.

Tabel 3.1 Alat Penelitian

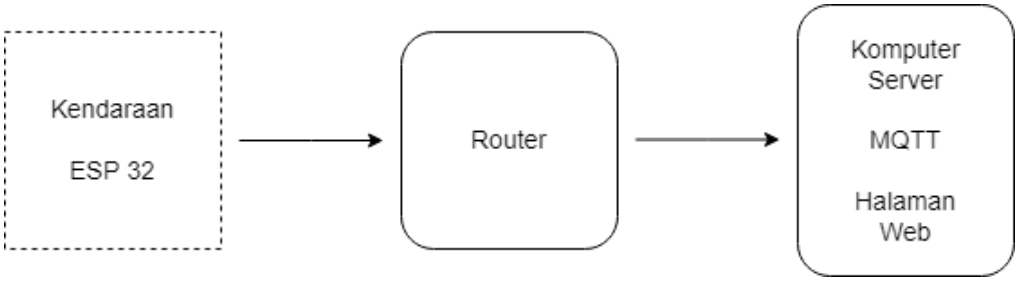
Alat dan Bahan	Fungsi
Mikrokontroler ESP32	Sebagai *chip* identitas kendaraan atau *client*.
Router	Sebagai *access point* dan router digunakan sebagai variabel jarak pada kecepatan kendaran.
Laptop	Sebagai *server* melakukan desain dan pemrograman sistem
Baterai	Sebagai sumber daya untuk *client* ESP32
Switch	Sebagai alat penghubung atau pemutus sumber daya
LED	Sebagai indikator ESP telah menyala

Tabel 3.2 Bahan Penelitian

Bahan	Fungsi
Arduino IDE	*Software* yang digunakan untuk memrogram ESP32 sebagai *client*
Node.js	Sebagai aplikasi yang digunakan untuk pengembangan aplikasi web
MQTT	Sebagai protokol yang digunakan untuk menjalankan sistem.

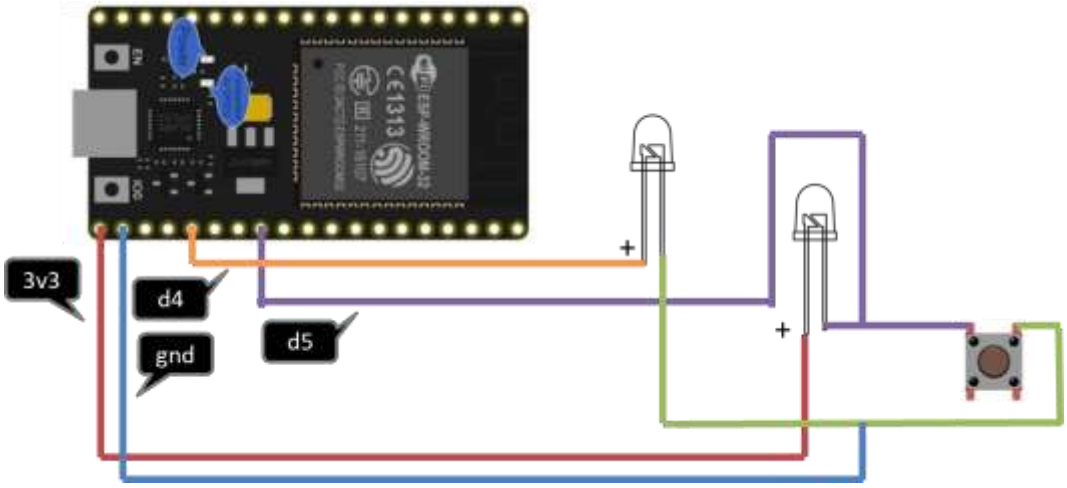
3.3 Diagram Blok dan Cara Kerja Sistem

Sistem pengukur kecepatan kendaraan ini bekerja dengan sistem nirkabel, atau yang biasa disebut dengan *Internet of Things* (IoT). Pada sistem ini Mikrokontroler ESP32 bekerja sebagai *client* yang digunakan sebagai alat pengidentifikasi kendaraan dan memancarkan data. Bila ditemukan *access point*, maka *client* akan terhubung ke *access point* yang selanjutnya diteruskan ke *server* menggunakan protokol MQTT untuk direkam kemudian ditampilkan pada sebuah halaman *web* yang telah dibuat. Diagram blok sistem dari alat ukur kecepatan kendaaraan menggunakan RFID Mikrokontroler ESP32 ditujukan oleh gambar 3.1.



Gambar 3.1 Diagram Blok Sistem

Sedangkan skema rangkaian yang digunakan ditunjukkan oleh gambar 3.2.

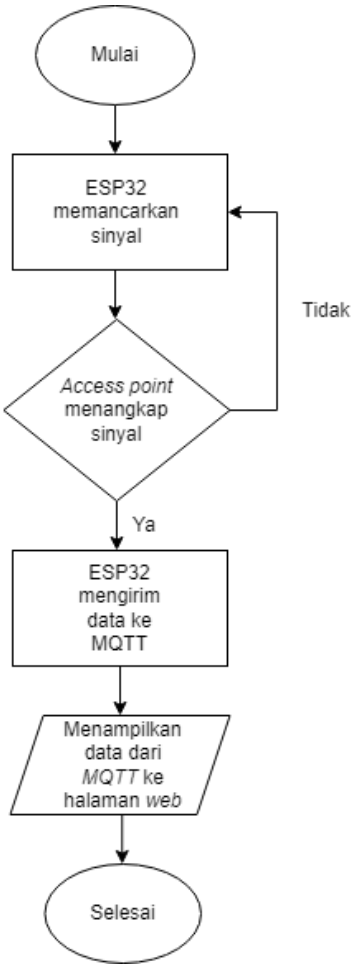


Gambar 3.2 Skema Rangkaian pada Client

Pada rangkaian ini pin 3V3 dihubungkan dengan kaki positif dari LED A, dan kaki negatif LED A dihubungkan pada *switch* serta pin D5. Lalu**,** PIN D4 dihubungkan pada kaki positif LED B. Sedangkan kaki negatif LED B dihubungan pada GND serta *switch*. LED A digunakan sebagai indikator bahwa ESP32 siap digunakan, sedangkan LED B digunakan sebagai lampu peringatan jika kecepatan kendaraan melebihi batas kecepatan yang telah digunakan.

3.4 Diagram Alir Sistem Kerja Alat

Alur kerja dari sistem alat ini secara keseluruhan dijabarkan secara berurut pada *flowchart* berikut. Proses kerja sistem alat ukur kecepatan kendaraan ini ditunjukkan pada gambar 3.3.

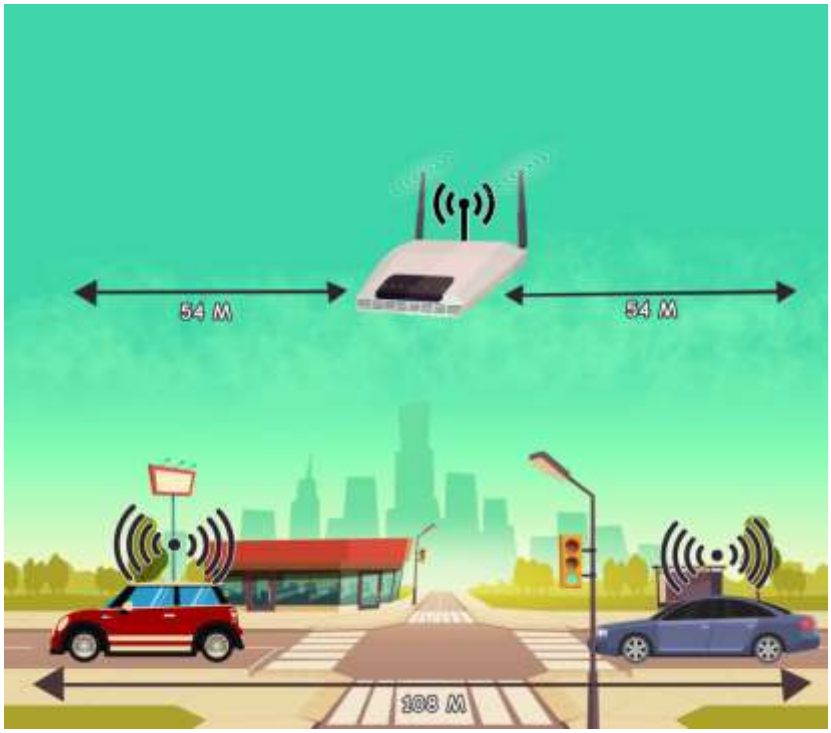


Gambar 3.3 Diagram Alir Sistem Kerja Alat

Sistem ini bekerja menggunakan sebuah Mikrokontroler ESP32 yang diberi tegangan dari sebuah baterai sebesar 3,7 V. Mikrokontroler ESP32 ini digunakan sebagai sebuah *client* yang memancarkan sinyal secara terus menerus karena memiliki sebuah modul *Wi-Fi* didalamnya. Sinyal ini akan terkoneksi dengan *server* apabila terhubung dengan sebuah *access point* melalui protokol MQTT untuk menghasilkan sebuah kecepatan terukur. Lalu, kecepatan terukur akan ditampilkan pada halaman web.

3.5 Sketsa Penggunaan Alat

Alat ini digunakan dengan membawa rangkaian Mikrokontroller ESP32 pada kendaraan. Sketsa penggunaan alat ini dapat ditunjukkan oleh gambar 3.4.



Gambar 3.4 Sketsa Penggunaan Alat

Saat kendaraan membawa rangkaian Mikrokontroller ESP32 kedalam sebuah *access point*, *server* akan menghitung waktu saat kendaraan terhubung dengan *access point* hingga kendaraan tidak terhubung lagi dengan *access point*. Pada percobaan ini, jangkauan yang dimiliki oleh *router* sebagai *access point* adalah 108 meter. Dimana 108 meter didapat dari 54 m dari kiri *router*, hingga 54 meter arah kanan *router*.

BAB IV
PENGUJIAN ALAT

Setelah rancang bangun alat dibuat, dilakukan uji coba agar dapat mengetahui apakah rancang bangun alat terealisasi dan bekerja sesuai fungsinya atau tidak. Pengujian ini meliputi pengujian keseluruhan sistem alat untuk mengukur kecepatan kendaraan. Selanjutnya dilakukan analisis hasil pengujian alat yang sudah dilakukan untuk mengetahui perbandingan hasil pengukuran kecepatan dengan kecepatan sesungguhnya.

4.1. Konektivitas *Server-Client*

Pada pengujian ini dilakukan uji konektivitas *server* dan *client*. Dimana dilakukan uji konektivitas dengan jarak tertentu antara *server* dan *client*. Pengujian ini dilakukan dengan meletakkan sebuah *router* tepat di tepi jalan dengan *client* yang akan bergerak ke arah kiri dan kanan pada jarak tertentu hingga *client* tidak terkoneksi lagi dengan *server*. Proses pengujian ini dipantau dengan menggunakan *command prompt* dari Node.Js. Hasil pengujian konektivitas *Server-Client* dapat dilihat pada tabel 4.1.

Tabel 4.1 Hasil Pengujian Konektivitas *Server-Client*

Jarak (m)	Ke Kanan	Ke Kiri
0 – 10	Terkoneksi	Terkoneksi
11 – 20	Terkoneksi	Terkoneksi
21 – 30	Terkoneksi	Terkoneksi
31 – 40	Terkoneksi	Terkoneksi
41 – 50	Terkoneksi	Terkoneksi
51 – 54	Terkoneksi	Terkoneksi
54 – 60	Tidak Terkoneksi	Tidak Terkoneksi

Dari data yang dihasilkan dapat disimpulkan bahwa dari jarak 0 hingga 54 meter ke kanan ataupun ke kiri *client* dapat terkoneksi dengan *access point* atau *server*. Akan tetapi, pada jarak lebih dari 54 meter, *client* tidak dapat terhubung

dengan *server*. Dapat disimpulkan *client* dapat terhubung dengan *server* dalam jarak 54 meter ke kanan dan 54 meter ke kiri, sehingga jarak *client* dapat terhubung dengan *server* yaitu dalam jangkauan maksimum 108 meter.

4.2. Kecepatan Kendaraan

Pada percobaan ini dilakukan enam kali percobaan dengan jarak yang tetap senilai 100 meter. Pengujian dilakukan dengan meletakkan rangkaian Mikrokontroller ESP32 pada sebuah sepeda motor untuk membandingkan nilai kecepatan terukur dengan nilai kecepatan sebenarnya. Untuk mendapatkan nilai kecepatan terukur kendaraan pada rancang bangun alat dapat dihitung menggunakan rumus matematis pada persamaan 2.3 pada bab dua.

Hasil perhitungan kecepatan kendaraan dalam enam kali percobaan pada jarak tetap 100 meter ditunjukkan oleh tabel 4.2.

Tabel 4.2 Hasil Pengukuran Kecepatan Kendaraan

Jarak (km)	Waktu Tempuh (s)	Kecepatan (km/h)	Status
0,1	9,892	36.495	AMAN
0,1	10,557	34,196	AMAN
0,1	11,743	30.742	AMAN
0,1	13,221	27,305	AMAN
0,1	10,186	35,441	AMAN
0,1	12,438	29,024	AMAN

Selisih nilai kecepatan terukur dengan nilai kecepatan sebenarnya akan dihitung guna mengukur persentase kesalahan sistem alat ukur kecepatan kendaraan ini. Tabel data persentase kesalahan akan ditunjukkan oleh tabel 4.3.

Tabel 4.3 Data Persentase Nilai Eror

Kecepatan Terukur (km/h)	Kecepatan Sebenarnya (km/h)	Selisih (km/h)	Persentase Kesalahan
36,495	33	3,495	9,57%
34,196	31	3,196	9,34%
30,742	28	2,742	8,91%
27,305	24	3,305	12,1%
35,441	32	3,441	9,7%
29,024	27	2,024	6,9%
	Rata-rata Persentase Kesalahan		9,42%

Berdasarkan kedua tabel diatas dapat dilihat hasil pengukuran kecepatan, serta perbedaan antara pengukuran menggunakan alat dan speedometer pada sepeda motor. Hasil kecepatan yang diperoleh bergantung pada waktu yang ditempuh. Pada waktu 9,892 s kecepatan yang terukur senilai 36,495 km/h, dan pada waktu 13,221 s kecepatan yang terukur senilai 27,305 km/h. Dari hasil pengujian dapat diketahui bahwa semua percobaan berstatus aman karena tidak melebihi batas kecepatan maksimum, baik untuk batas kecepatan maksimum pada jalan antar kota pada 80 km/jam maupun jalan dalam kota pada 60 km/jam. Persentase kesalahan pada hasil pengukuran kecepatan dengan menggunakan alat sebesar 9,42%. Hal ini menunjukkan pengukuran kecepatan kendaraan menggunakan metode ini dapat dilakukan namun membutuhkan pengembangan sistem agar hasil yang ditunjukkan dapat lebih presisi.

BAB V
PENUTUP

5.1. Kesimpulan

Dari penelitian yang telah dilakukan tentang sistem alat ukur kecepatan kendaraan pada jalan raya menggunakan *Router Wi-Fi* dan Mikrokontroller ESP32, dapat disimpulkan:

1. Telah berhasil dirancang dan direalisasikan sebuah sistem untuk mengukur kecepatan kendaraan dengan menggunkan *Router Wi-Fi* dan Mikrokontroler ESP32 yang selanjutnya data tersebut dapat ditampilkan pada sebuah halaman *web*.
2. Pada sistem alat ukur kecepatan kendaraan ini dapat mengoneksikan *client* dengan *access point* dan *server* dalam jarak maksimum 108 meter.
3. Rata-rata nilai persentase kesalahan pada alat ini adalah 9,42%, hal ini menunjukkan metode pengukuran kecepatan ini dapat digunakan namun perlu pengembangan lebih lanjut.

5.2. Saran

Saran dari hasil penelitian yang telah dilakukan didapatkan adalah sebagai berikut:

1. Jika ingin memperluas daerah pemantauan dapat menggunakan *router* Wi-Fi dengan jangkauan lebih jauh.
2. Perlu pengecekan kembali agar meminimalisir *error* yang dapat terjadi
3. Perlu pengembangan metode lebih lanjut agar alat dapat mengukur kecepatan dengan lebih presisi.

DAFTAR PUSTAKA

Aisyah, Siti. 2019. *Perancangan Sistem Monitoring dan Notifikasi Kejadian Upwelling Berbasis Internet of Things (IoT) Menggunakan Protokol MQTT*. Skripsi. Tangerang: Sekolah Tinggi Meteorologi Klimatologi dan Geo*f*isika.

Budijanto, Arif dan Winardi, S. 2021. *Interfacing ESP32*. Surabaya: Scopindo Media Pustaka.

Budioko, T. 2016. *Sistem Monitoring Suhu Jarak Jauh Berbasis Internet Of Things Menggunakan Protokol MQTT*. Seminar Nasional Riset Teknologi Informasi-SRITI, Vol. 8, 353-358.

Fajrin, R. 2017. *Pengembangan Sistem Informasi Geografis Berbasis Node. Js Untuk Pemetaan Mesin Dan Tracking Engineer dengan Pemanfaatan Geolocation pada PT IBM Indonesia*. Jurnal Komputer Terapan, Vol. *3*(1), 33-40.

Herlambang, M. Linto dan Azis Catur. 2008. *Panduan Lengkap Menguasai Router*. Yogyakarta: Andi Offset

Iqbal, M., M. Husni, dan H. Studiawan. 2012. *Implementasi Klien SIP Berbasis Web Menggunakan HTML5 dan Node. js*. Jurnal Teknik ITS, Vol.1(1), A242-A245.

Jatmiko, D., dan Salita Prini. 2019. *Implementasi dan Uji Kinerja Algoritma Background Subtraction pada ESP32*. Komputika, Vol. 8(2), 59-65.

Kusumah, Hendra dan Restu A. Pradana. 2019. *Penerapan Trainer Interfacing Mikrokontroler dan Internet Of Things Berbasis ESP32 pada Mata Kuliah Interfacing*. Journal Cerita, Vol. 5(2), 120-134.

Lutvianto, K. 2019. *Perancangan Sistem Single Sign On dengan Metode Otentikasi Oauth 2.0*. Doctoral dissertation. Yogyakarta: STMIK AKAKOM.

Mauliza, Rizki Intan, Tania Bonita Sabrina, dan Wahyu Maulana. 2019. *Pelanggaran Kecepatan Kendaraan pada Ruas Jalan Tol Cipularang*. Jurnal Teknil Sipil, Vol.5(1), 39.

NHTSA. 2005. *National Highway Traffic Safety Administration*. Washington, D.C: United States Congress.

Samsumar, L. D., dan Sofian Hadi. 2018. *Pengembangan Jaringan Komputer Nirkabel (Wifi) Menggunakan Mikrotik Router (Studi Kasus pada SMA PGRI Aikmel).* METHODIKA: Jurnal Teknik Informatika dan Sistem Informasi, Vol.4(1), 1-9.

Saputra, G. Y., dkk. .2017. *Penerapan Protokol MQTT pada Teknologi WAN (Studi Kasus Sistem Parkir Univeristas Brawijaya)*. Malang: Universitas Brawijaya.

Sinaulan, Olivia M., Yaulie DY Rindengan, dan Brave A. 2015. *Perancangan Alat Ukur Kecepatan Kendaraan Menggunakan AT Mega 16*. Jurnal Teknik Elektro dan Komputer, Vol.4(3), 60-70.

Sofana, I. 2012. *Cisco CCNA dan Jaringan Komputer*. Bandung: Informatika.

Zulqarnain, A., Dini, H. S., dan Azis, H. 2020. *Rancang Bangun Prototype Suplai Daya Termoelektrik Pada Kompor Biomassa Dengan Menggunakan Mikrokontroler ESP32 Berbasis Internet Of Things*. Doctoral dissertation. Jakarta: Institut Teknologi PLN.

LAMPIRAN

Lampiran 1: Program Arduino IDE

```
#include "WiFi.h"
#define LED 2
#define BUTTON 5
#define LED2 4

int buttonState = 0;
bool ledState = false;

//user Defined Variables
const char* ssid = "NIA ANISTI";
const char* password = "NIAANISTI";
const uint16_t port = 5000;
IPAddress host(192,168,1,1);

void setup() {
  Serial.begin(115200);
  pinMode(LED,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(BUTTON,INPUT);
  digitalWrite(LED2,HIGH);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    digitalWrite(LED,HIGH);
    delay(500);
    digitalWrite(LED,LOW);
  }

  digitalWrite(LED,HIGH);
}

void loop() {
  buttonState = digitalRead(BUTTON);

  if(WiFi.status() != WL_CONNECTED){
    WiFi.begin(ssid, password);
    delay(500);
    digitalWrite(LED,LOW);
    delay(500);
    digitalWrite(LED,HIGH);
  }else{
    delay(500);
    digitalWrite(LED,HIGH);
  }
  if (ledState == true){
```

```
    digitalWrite(LED2,HIGH);
    WiFiClient client;
    if (client.connect(host, port)) {
        while (true){
          client.print("40");
          delay(500);
        }
    }
    Serial.println("Connected to server successful!");
  }else{
    digitalWrite(LED2,LOW);
  }

  if (buttonState == HIGH) {
    delay(500);
    Serial.println("Button LOW");
  }else{
    ledState = true;
    delay(500);
    Serial.println("Button HIGH");
  }
}
```

Lampiran 2: Program Server MQTT

```
const Net = require('net');
const Excel = require('exceljs');
const express = require('express');
const app = express();
const path = require('path');

// The port on which the server is listening.
const port = 5000;
let speeds;


const readline = require('readline');
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});


var workbook = new Excel.Workbook();
// workbook.xlsx.readFile('result.xlsx')
//     .then(function() {
//         //writeExcel('28/1/2022', 4.20, 42.85)
//     });

function readExcel() {
    var worksheet = workbook.getWorksheet(1);
    worksheet.eachRow({
        includeEmpty: true
    }, function(row, rowNumber) {
        console.log("Row " + rowNumber + " = " +
JSON.stringify(row.values));
    });
}

function writeExcel(date, time, speed, expected) {
    var worksheet = workbook.getWorksheet(1);
    var lastRow = worksheet.lastRow.number;
    var getRowInsert = worksheet.getRow(++(lastRow));
    getRowInsert.getCell('A').value = date;
    getRowInsert.getCell('B').value = time;
```

```
    getRowInsert.getCell('C').value = speed;
    getRowInsert.getCell('D').value = expected;
    getRowInsert.commit();
    return workbook.xlsx.writeFile('result.xlsx');
}

const router = express.Router();

app.set("view engine", "pug");
app.set("views", path.join(__dirname, "views"));

router.get("/", (req, res) => {
    res.render("index", {
        speed: 0
    });
});

router.get("/2", (req, res) => {
    res.render("index", {
        speed: Math.floor(speeds)
    });
});




var expectSpeed;
rl.question('Enter Expected Speed : ', function(name) {
    expectSpeed = name;
});

const server = new Net.Server();

server.listen(port, function() {
    console.log(`Server listening for connection requests on
socket localhost:${port}`);
});

server.on('connection', function(socket) {
    var startTime = Date.now();
    console.log('A new connection has been established.');

    socket.write('Hello, client.');
```

```
    // The server can also receive data from the client by
reading from its socket.
    var i = 1;
    socket.on('data', function(chunk) {
        console.log(`Data received: ${chunk.toString()}`);
        i = 1;
        if (chunk.toString().length == 0) {
            var endTime = Date.now() - startTime - 2;
            console.log(`no data received client must be
disconnected`);
            console.log(endTime / 1000);
        }
    }, 500);

    setInterval(function() {
        i++
        if (i == 60) {
            var endTime = Date.now() - startTime - 2;
            console.log(`no data received client must be
disconnected`);
            speeds = 50 / (endTime / 1000) * 3.6;
            console.log(speeds);
            socket.end();
            socket.destroy();
            workbook.xlsx.readFile('result.xlsx')
                .then(function() {
                    var todayDate = new
Date().toJSON().slice(0, 10).replace(/-/g, '/');
                    writeExcel(todayDate, endTime / 1000,
speeds, expectSpeed);
                }).catch(error => console.log("Error
Occurred : " + error));
            return;
        }
    }, 40)

    console.log("program ends");

    // When the client requests to end the TCP connection
with the server, the server
    // ends the connection.
    socket.on('end', function() {
        var endTime = Date.now() - startTime - 2;
        console.log('Closing connection with the client');
        speeds = 50 / (endTime / 1000) * 3.6;
```

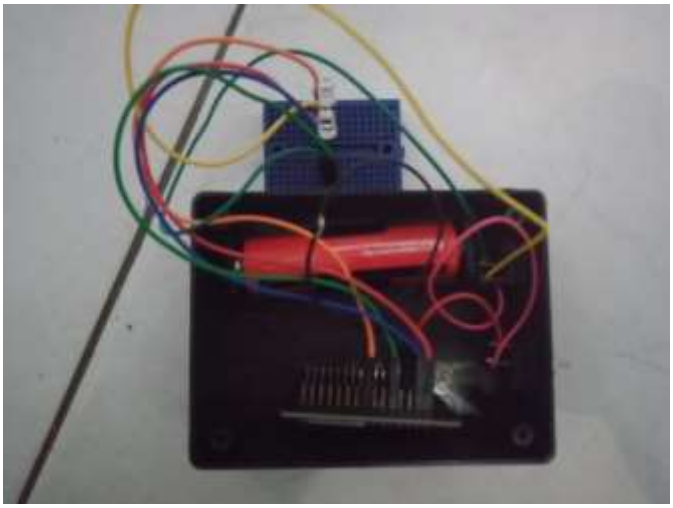
```
        console.log(speeds);
        socket.end();
        socket.destroy();
        workbook.xlsx.readFile('result.xlsx')
            .then(function() {
                writeExcel(new Date().toJSON().slice(0,
10).replace(/-/g, '/'), endTime, speeds, expectSpeed);
            });
    });

    // Don't forget to catch error, for your own sake.
    socket.on('error', function(err) {
        console.log(`Error: ${err}`);
    });
});

//add the router


app.use('/', router);
app.listen(8080);
```

Lampiran 3: Dokumentasi Alat



Lampiran 4: Datasheet Mikrokontroler ES

1 Overview

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The ESP32 series of chips includes ESP32-D0WD-V3, ESP32-U4WDH, ESP32-S0WD, ESP32-D0WDQ6-V3 (NRND), ESP32-D0WD (NRND), and ESP32-D0WDQ6 (NRND), among which, ESP32-D0WD-V3, ESP32-D0WDQ6-V3 (NRND), and ESP32-U4WDH are based on ECO V3 wafer.

For details on part numbers and ordering information, please refer to Section 7.

For details on ECO V3 instructions, please refer to *ESP32 ECO V3 User Guide*.

1.1 Featured Solutions

1.1.1 Ultra-Low-Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

> **Note:**
> For more information, refer to Section 3.7 *RTC and Low-Power Management*.

1.1.2 Complete Integration Solution

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

1.2 Wi-Fi Key Features

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps
- WMM
- TX/RX A-MPDU, RX A-MSDU
- Immediate Block ACK

P32

- Defragmentation
- Automatic Beacon monitoring (hardware TSF)
- 4 x virtual Wi-Fi interfaces
- Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes
  Note that when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.
- Antenna diversity

Note:
For more information, please refer to Section 3.5 Wi-Fi.

1.3 Bluetooth Key Features

- Compliant with Bluetooth v4.2 BR/EDR and Bluetooth LE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +9 dBm transmitting power
- NZIF receiver with -94 dBm Bluetooth LE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR Bluetooth LE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic Bluetooth and Bluetooth LE
- Simultaneous advertising and scanning

1.4 MCU and Advanced Features

1.4.1 CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s)
- CoreMark® score:
  - 1 core at 240 MHz: 504.85 CoreMark; 2.10 CoreMark/MHz
  - 2 cores at 240 MHz: 994.26 CoreMark; 4.14 CoreMark/MHz
- 448 KB ROM
- 520 KB SRAM

- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

1.4.2 Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/Bluetooth functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

1.4.3 Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I2S
- 2 × I2C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- TWAI®, compatible with ISO 11898-1 (CAN Specification 2.0)
- RMT (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

1.4.4 Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:

- AES
- Hash (SHA-2)
- RSA
- ECC
- Random Number Generator (RNG)

1.5 Applications (A Non-exhaustive List)

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition
- Image Recognition
- Mesh Network
- Home Automation
  - Light control
  - Smart plugs
  - Smart door locks
- Smart Building
  - Smart lighting
  - Energy monitoring
- Industrial Automation
  - Industrial wireless control
  - Industrial robotics
- Smart Agriculture
  - Smart greenhouses
  - Smart irrigation
  - Agriculture robotics
- Audio Applications
  - Internet music players
  - Live streaming devices
  - Internet radio players
  - Audio headsets
- Health Care Applications
  - Health monitoring
  - Baby monitors
- Wi-Fi-enabled Toys
  - Remote control toys
  - Proximity sensing toys
  - Educational toys
- Wearable Electronics
  - Smart watches
  - Smart bracelets
- Retail & Catering Applications
  - POS machines
  - Service robots

1.6 Block Diagram

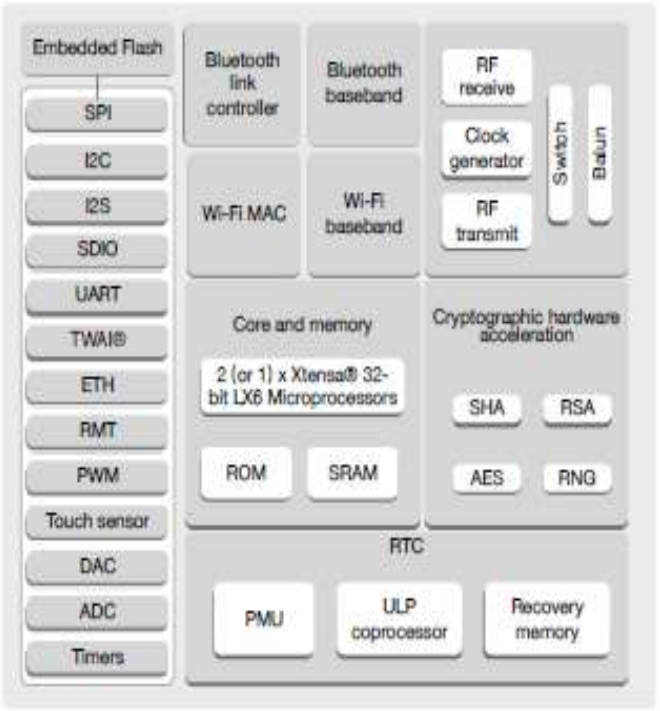


Figure 1: Functional Block Diagram

Note:
Products in the ESP32 series differ from each other in terms of their support for embedded flash and the number of CPUs they have. For details, please refer to Section 7 *Part Number and Ordering Information*.

2 Pin Definitions

2.1 Pin Layout

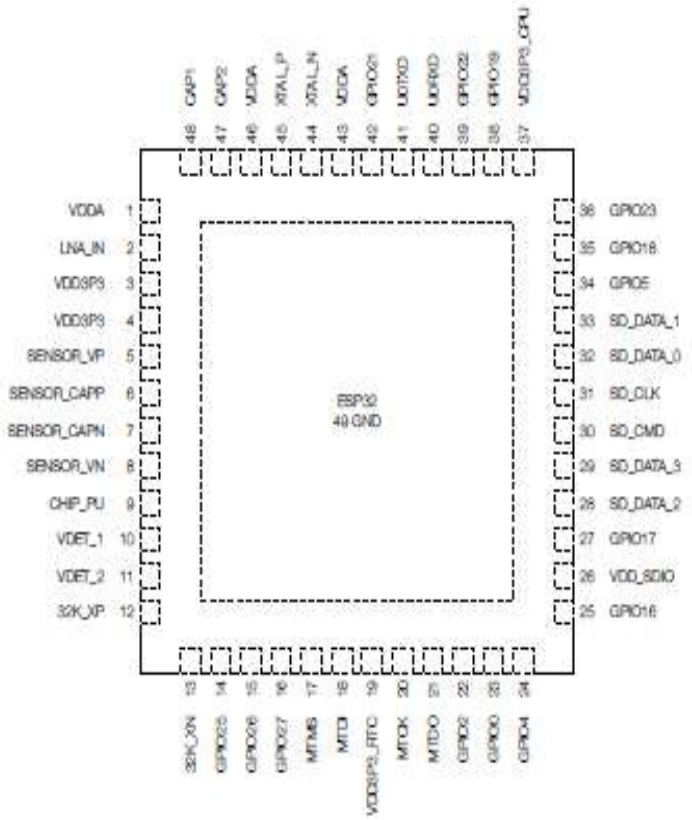


Figure 2: ESP32 Pin Layout (QFN 6*6, Top View)