

**RANCANG BANGUN SERVER IoT MENGGUNAKAN *CLOUD*
COMPUTING DAN PROTOKOL MQTT UNTUK
KOMUNIKASI PADA PERANGKAT BIKEBIKEAJA**

LAPORAN TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana
teknik dari Institut Teknologi Sumatera



Diajukan Oleh:

Ahmad Fajar
118130040

**PROGRAM STUDI TEKNIK ELEKTRO
JURUSAN TEKNOLOGI PRODUKSI DAN INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN**

PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama : Ahmad Fajar

NIM : 118130040

Dengan ini menyatakan bahwa laporan dengan judul:

**RANCANG BANGUN SERVER IoT MENGGUNAKAN *CLOUD*
COMPUTING DAN PROTOKOL MQTT UNTUK KOMUNIKASI PADA
PERANGKAT BIKEBIKEAJA**

Merupakan laporan tugas akhir yang saya buat terbebas dari unsur plagiasi. Adapun pendapat dari sumber lain telah dikutip melalui penulisan referensi yang sesuai dan material yang digunakan sudah memperoleh izin untuk ditampilkan di laporan tugas akhir ini.

Surat pernyataan ini dibuat dengan sebenar-benarnya dan jika dikemudian hari diketahui keliru, saya bersedia menerima sanksi sesuai aturan yang berlaku.

Lampung Selatan, 23 Mei 2023

Ahmad Fajar
118130040

PENGESAHAN

TUGAS AKHIR

RANCANG BANGUN SERVER IoT MENGGUNAKAN *CLOUD* *COMPUTING* DAN PROTOKOL MQTT UNTUK KOMUNIKASI PADA PERANGKAT BIKEBIKEAJA

disusun Oleh:

Ahmad Fajar
118130040

Telah dipertahankan didepan Tim Penguji Tugas Akhir Program Studi Teknik
Elektro Sub – Jurusan Teknik Elektro, Informatika dan Sistem Fisis Jurusan
Teknologi Produksi dan Industri Institut Teknologi Sumatera pada tanggal:

(23 Mei 2023)

disetujui pada tanggal: Desember 2023

Dosen Pembimbing Tugas Akhir II

Dosen Pembimbing Tugas Akhir II

Purwono Prasetyawan, S.T., M.T.
NRK. 1985 0525 2021 1300

Syamsyarief Baqaruzi, S.T., M.T
NIP. 19900907 201903 1 015

MOTTO

Hduik Baraka, Mati Bariman

PERSEMBAHAN

Alhamdulillah, puji dan syukur saya panjatkan kehadiran Allah SWT, karena berkat limpahan rahmat dan karunianya sehingga saya dapat mempersembahkan laporan penelitian Tugas Akhir dengan sebaik-baiknya kepada:

1. Kedua orang tua yang selalu memberikan support terbaik
2. Seluruh keluarga besar ibu dan bapak
3. Dosen pembimbing yang senantiasa memberikan solusi dan saran
4. Teman-teman seperjuangan kampus ITERA
5. Dosen teknik elektro di Institut Teknologi Sumatera

RANCANG BANGUN SERVER IoT MENGGUNAKAN CLOUD COMPUTING DAN PROTOKOL MQTT UNTUK KOMUNIKASI PADA PERANGKAT BIKEBIKEAJA

Ahmad Fajar (118130040)

Purwono Prasetyawan, S.T., M.T.

Syamsyarief Baqaruzi, S.T., M.T.

INTISARI

Pertumbuhan teknologi telah melaju pesat dalam beberapa tahun terakhir, dan ini telah mendorong peningkatan penggunaan internet di berbagai bidang. Salah satu bidang yang mengalami peningkatan penggunaan internet adalah *Internet of Things* (IoT). *Internet of Things* telah diterapkan dalam berbagai hal, seperti mobil pintar, motor pintar, rumah pintar, dan sepeda pintar yang umumnya terhubung ke internet. Dengan memanfaatkan teknologi *Internet of Things*, kita dapat menggunakan server sebagai tempat penyimpanan data virtual. Server tersebut terhubung langsung melalui internet dan menerima berbagai data dari perangkat yang terhubung. Data tersebut mencakup informasi, sensor, waktu, lokasi geografis, dan output yang sedang dikendalikan. Namun, data tersebut juga rentan terhadap berbagai risiko yang dapat menyebabkan penyalahgunaan perangkat IoT, seperti akses kontrol yang tidak sah, pencurian data, atau kerusakan sistem server yang tidak dapat dipulihkan. Oleh karena itu, diperlukan sistem manajemen server yang ideal, aman, dan mendefense yang mungkin dieksploitasi oleh pihak yang tidak bertanggung jawab. Penggunaan protokol komunikasi MQTT memungkinkan pengoperasian server IoT yang terhubung dengan perangkat BikeBikeAja yang cepat sekaligus aman.

Kata kunci: *Internet Of Things*, Server , MQTT, Data Komunikasi, BikeBikeAja.

RANCANG BANGUN SERVER IoT MENGGUNAKAN CLOUD COMPUTING DAN PROTOKOL MQTT UNTUK KOMUNIKASI PADA PERANGKAT BIKEBIKEAJA

Ahmad Fajar (118130040)

Purwono Prasetyawan, S.T., M.T.

Syamsyarief Baqaruzi, S.T., M.T

ABSTRACT

Technological advancement has been progressing rapidly in recent years, leading to an increased usage of the internet in various fields. One of the areas experiencing a surge in internet usage is the Internet of Things (IoT). The Internet of Things has been implemented in various things, such as smart cars, smart motorcycles, smart homes, and smart bikes, which are commonly connected to the internet. By utilizing IoT technology, we can employ servers as virtual data storage. These servers are directly connected to the internet and receive diverse data from connected devices. The data includes information, sensors, time, geographic location, and controlled outputs. However, this data is also susceptible to various risks that can result in the misuse of IoT devices, such as unauthorized access control, data theft, or irreparable server system damage. Therefore, an ideal and secure server management system is required to defend against potential exploitation by malicious parties. The use of MQTT communication protocol enables the operation of a fast and secure IoT server connected to the BikeBikeAja device.

Keywords: Internet of Things, Server, MQTT, Communication Data, Just Bike Bikes.

KATA PENGANTAR

Assalamua'laikum Warohmatullahi Wabarokatuh

Alhamdulillah dengan menyebut nama Allah yang Maha Pengasih lagi Maha Penyayang, puji syukur penulis lantunkan kepada Allah SWT karena berkat limpah rahmat serta karunia-Nya sehingga saya dapat menyelesaikan laporan Tugas Akhir dengan sebaik-baiknya. Shalawat dan salam senantiasa diucapkan kepada Nabi besar Muhammad SAW yang telah menjadikan teladan bagi umat manusia.

Pada dokumen tugas akhir yang menjadi salah satu syarat untuk memperoleh gelar sarjana teknik dari Institut Teknologi Sumatera, penulis mengucapkan syukur Alhamdulillah karena dapat menyelesaikan laporan ini dengan judul "Rancang Bangun Aplikasi Sistem Keamanan Sepeda Listrik Berbasis Android".

Kami ucapkan terima kasih kepada pihak-pihak yang mendukung serta terlibat di dalam kegiatan untuk menyelesaikan dokumen tugas akhir ini dan telah memberikan semangat serta motivasinya. Untuk itu, penulis sangat berterima kasih kepada:

1. Allah SWT, yang telah memberikan kesehatan serta kemudahan sehingga penulis dapat menyelesaikan tugas ini,
2. Kepada kedua orang tua yang telah memberikan dukungan dan kontribusi besar bagi kami,
3. Bapak Syamsyarief Baqaruzi, S.T,M.T selaku ketua prodi Teknik Elektro,
4. Bapak Prof. **Dr. Ir. Deny Juanda Puradimaja**, DEA, selaku ketua Jurusan Teknologi Produksi dan Industri (JTPI),
5. Bapak Purwono Prasetyawan, S.T., M.T. dan Syamsyarief Baqaruzi, S.T., M.T selaku dosen pembimbing,
6. Teman-teman yang telah berkontribusi dan turut serta membagikan ilmu yang bermanfaat dalam perancangan tugas akhir ini.
7. Kepada Fia Andani yang selalu memberikan semangat dan **dukungan kepada**

Penulis mengharapkan semoga informasi dan materi yang ada dalam laporan tugas akhir ini bisa menambah ilmu pengetahuan, wawasan serta kemajuan bagi

pembaca dan rekan-rekan mahasiswa lainnya. Kami berupaya sebaik mungkin agar laporan tugas akhir ini mudah dipahami dan membawa dampak positif bagi pembaca dan meningkatkan kepedulian lingkungan hidup. Tidak ada yang sempurna di dunia ini selain Allah SWT yang Maha Sempurna.

Oleh karena itu, kritik serta saran yang membangun agar menjadi masukan agar karya tulis selanjutnya dari semua pihak sangat dibutuhkan. Demikian laporan tugas akhir ini kami buat, jika ada kesalahan kata dalam penulisan materi yang disampaikan, kami memohon maaf sebesar-besarnya. Semoga laporan tugas akhir ini bermanfaat, baik untuk penulis serta bagi para pembaca. Atas perhatiannya penulis ucapkan terima kasih.

Wassalamu'alaikum Warohmatullahi Wabarokatuh

Lampung Selatan, 23 Mei 2023

Ahmad Fajar

DAFTAR ISI

LAPORAN TUGAS AKHIR.....	1
PERNYATAAN	ii
PENGESAHAN	iii
MOTTO.....	iv
PERSEMBAHAN	v
INTISARI.....	vi
ABSTRACT	vii
KATA PENGANTAR	viii
DAFTAR ISI.....	x
DAFTAR TABEL.....	xii
DAFTAR GAMBAR.....	xiii
DAFTAR LAMPIRAN	xiv
DAFTAR SINGKATAN	xv
BAB I PENDAHULUAN	12
1.1 Latar Belakang.....	12
1.2 Rumusan Masalah	13
1.3 Batasan Masalah.....	14
1.4 Tujuan Pengembangan	14
BAB II KAJIAN PUSTAKA	15
2.1 Kajian Teori.....	15
2.1.1 Mikrokontroler	15
2.1.2 GSM SIM 7600 G	16
2.1.3 MQTT.....	18
2.1.4 Cloud Computing	19
2.1.5 IoT	21
2.2 Kajian Pengetahuan Pendukung yang Relevan	22
BAB III PERANCANGAN	24
3.1 Model dan Prosedur Pengembangan	24
3.1.1 Metodologi Perencanaan.....	24
3.1.2 Penggunaan Rumus	26
3.1.3 Timeline Pengerjaan.....	28
3.1.4 Spesifikasi Produk.....	28
3.1.5 Prosedur Pengujian.....	28
3.2 Spesifikasi Produk	Error! Bookmark not defined.
3.2.1 Spesifikasi Hardware.....	Error! Bookmark not defined.

3.2.2	Spesifikasi Software	Error! Bookmark not defined.
3.3	Instrumen dan Teknik Analisis	32
3.4	Perancangan Produk	32
BAB IV HASIL DAN ANALISIS		34
4.1	Hasil Implementasi dan Produk	34
4.1.1	Implementasi Server	34
4.1.2	Implementasi MQTT Broker	35
4.2	Hasil Pengujian	38
4.2.1	Pengujian Konektifitas IoT	38
4.2.2	Pengujian Publisher dan subscriber	45
4.2.3	Pengujian Keamanan Server dan MQTT	47
BAB V SIMPULAN		51
5.1	Kesimpulan	51
5.2	Saran	51
DAFTAR PUSTAKA		53
Lampiran		56

DAFTAR TABEL

Tabel 2. 1 Spesifikasi ESP32 Wrover E	16
Tabel 2. 2 Spesifikasi GSM SIM 7600 G-H	17
Tabel 2. 3 Spesifikasi MQTT Broker	18
Tabel 3. 1 Metode Wartefall	25
Tabel 4. 1 Ping test	39
Tabel 4. 2 Tabel pengiriman topik Rental Publisher dan Subscriber	47

DAFTAR GAMBAR

Gambar 2. 1 ESP32 Wrover E	16
Gambar 2. 2 SIM7600G-H	17
Gambar 2. 3 Sistematis MQTT	18
Gambar 2. 4 Cloud Computing	20
Gambar 3. 1 Arsitektur Utama BikeBikeAja	24
Gambar 3. 2 Metode Wartefall	25
Gambar 4. 1 Cloud Computing Ubuntu Server 20.04 LTS	34
Gambar 4. 2 Versi MQTT Broker Mosquitto	35
Gambar 4. 3 Service MQTT	36
Gambar 4. 4 Topik MQTT	37
Gambar 4. 5 Implementasi Publisher dan Subscriber	38
Gambar 4. 6 Pengujian Ping	39
Gambar 4. 7 Pesan yang diterima di server	41
Gambar 4. 8 Konektifitas perangkat Bike Bike Aja	41
Gambar 4. 9 MQTT Fail	42
Gambar 4. 10 Service MQTT Bermasalah	42
Gambar 4. 11 Maintenance Service MQTT	43
Gambar 4. 12 Status Mosquitto	43
Gambar 4. 13 Perangkat Bike Bike Terkoneksi	44
Gambar 4. 14 Pesan dari MQTT	44
Gambar 4. 15 MQTT Publisher rental 1	45
Gambar 4. 16 MQTT Subscriber	45
Gambar 4. 17 MQTT Publisher rental 0	46
Gambar 4. 18 MQTT Subscriber	46
Gambar 4. 19 Proses Sistem Penyewaan	46
Gambar 4. 20 Hasil Scanning Server	48
Gambar 4. 21 Implementasi Autentikasi MQTT	49

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

Singkatan	Kepanjangan
IoT	<i>Internet of Things</i>
MQTT	Message Queue Telemetry Transport
DHCP	Dynamic Host Configuration Protocol
HTTP	Hypertext Transfer Protocol
DNS	Domain Name System
FTP	File Transfer Protocol
SPI	Serial Peripheral Interface
ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
UART	Universal Asynchronous Receiver-Transmitter
MHz	Megahertz
GPIO	General Purpose Input/Output
KB	Kilobyte
RAM	Random Access Memory
MB	Megabyte
GSM	Global System for Mobile Communications
SIM	Subscriber Identity Module
GPRS	General Packet Radio Service
EDGE	Enhanced Data rates for GSM Evolution
WCDMA	Wideband Code Division Multiple Access
HSDPA	High-Speed Downlink Packet Access
HSPA	High-Speed Packet Access
FDD-LTE	Frequency Division Duplex Long Term Evolution
TDD-LTE	Time Division Duplex Long Term Evolution
GPS	Global Positioning System
GLONASS	Global Navigation Satellite System
BeiDou	BeiDou Navigation Satellite System
IBM	International Business Machines Corporation
OSI	Open Systems Interconnection
TCP/IP	Transmission Control Protocol/Internet Protocol

Singkatan	Kepanjangan
OASIS	Organization for the Advancement of Structured Information Standards
M2M	Machine-to-Machine
WSN	Wireless Sensor Network
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
LTS	Long-Term Support
SSD	Solid-State Drive
TB	Terabyte
Gbps	Gigabits per second
Bps	Bytes per second
Kbps	Kilobits per second
Gbps	Gigabits per second
SSL	Secure Sockets Layer
TLS	Transport Layer Security
QoS	Quality of Service
VPS	Virtual Private Server
SWAP	Swap Space
ICMP	Internet Control Message Protocol
Ms	Milliseconds
Nmap	Network Mapper

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang semakin pesat di era modern teknologi informasi dan komputer ini mengakibatkan persaingan dalam kemajuan iptek serta memberikan daya efektifitas dan efisiensi yang sudah terbukti mampu mempercepat kinerja di berbagai aspek kehidupan[1]. Dalam pemanfaatan teknologi informasi dan komputer saat ini khususnya di bidang IoT dapat mempermudah kita dalam mengontrol serta memonitoring perangkat elektronik dari jarak jauh dengan menggunakan internet sebagai media penghubungnya. Sehingga dengan pemanfaatan IoT tersebut para pengguna dapat mengoptimalkan serta mengelola informasi dari perangkat elektronik dalam waktu yang sesingkat mungkin. Tidak hanya pemanfaatan teknologi IoT ini perlu dioptimalkan, ada beberapa aspek-aspek penting lainnya agar pemanfaatan IoT dapat berjalan dengan maksimal diantaranya pengelolaan server yang baik akan memberikan jaminan kehandalan serta konektivitas dari IoT tersebut[2].

Server merupakan sebuah sistem yang menyediakan jenis layanan tertentu dalam sebuah jaringan komputer. Selain itu server dapat menjadi penyimpanan data virtual yang didapatkan dari berbagai layanan yang bekerja. Sebuah server biasanya didukung dengan prosesor yang bersifat scalable dan RAM, juga dilengkapi dengan sistem operasi khusus untuk menjalankannya. Di dalam sistem operasi server, umumnya berbagai macam *Service* yang dijalankan seperti *DHCP Service*, *HTTP Service*, *DNS Service*, *FTP Service* dan *MQTT Service*[3]. Ketika IoT dan server sudah berkaitan satu sama lain, maka kita dapat menghubungkan ke berbagai perangkat elektronik yang mudah dikontrol dan dimonitoring seperti mobil, motor, sepeda dan bahkan rumah. Seperti yang dikatakan sebelumnya IoT dapat memberikan mempercepat kinerja dan dapat memberikan keuntungan di berbagai sektor dan aspek kehidupan.

Salah satu sektor yang memberikan keuntungan besar ialah bidang pariwisata. Pada pariwisata sendiri banyak hal yang bisa diterapkan dan implementasi

khususnya yang terkait dengan IoT seperti kendaraan yang bisa dipakai dimana saja dengan akses yang mudah sekaligus aman digunakan. Dengan contoh kendaraan tersebut akan menghasilkan sesuatu ide bisnis sewa menyewa dan dapat diterapkan di sektor pariwisata. Kendaraan yang cocok untuk diterapkan di dalam sektor pariwisata yang berbentuk transaksi sewa menyewa ialah sepeda. Sepeda adalah salah satu moda transportasi yang ramah lingkungan yang bisa menjadi solusi dalam akses dimana saja dan aman digunakan. Pada dasarnya bersepeda dapat diartikan oleh wisatawan sebagai bagian integral dari ekskursi atau liburan, cara yang menguntungkan demi meningkatkan kualitas waktu liburan[4]. Dengan adanya sepeda, para pengguna lebih mudah bepergian menuju ke lokasi – lokasi tertentu yang menarik di suatu area pariwisata. Oleh karena itu perlu adanya pemantauan aktivitas ke para pengguna agar tidak terjadi sesuatu yang tidak diinginkan selama bepergian ke suatu lokasi tertentu seperti pencurian kendaraan, terjadinya kecelakaan dan resiko yang berkaitan dengan kendaraan pada umumnya.

BikeBikeAja adalah produk yang dikembangkan untuk meningkatkan keamanan sepeda dan memfasilitasi layanan penyewaan antara penyedia dan pengguna. Produk ini dirancang dengan memanfaatkan teknologi terkini, termasuk penggunaan IoT, server sebagai penyimpanan data virtual, dan protokol untuk mengakses server. Oleh karena itu, penelitian ini berfokus pada pengembangan sistem rental BikeBikeAja, dengan penekanan pada perancangan dan implementasi server IoT menggunakan *Cloud Computing* dan protokol MQTT untuk komunikasi pada perangkat BikeBikeAja. Hasil yang diperoleh dari penelitian ini mencakup analisis data, kecepatan akses ke server, dan desain server yang ideal untuk perangkat IoT pada produk BikeBikeAja..

1.2 Rumusan Masalah

Adapun rumusan masalah dari penelitian ini yaitu:

1. Bagaimana rancangan server yang ideal untuk perangkat IoT BikeBikeAja dengan menggunakan protokol MQTT?
2. Bagaimana cara mengakses protokol MQTT pada server produk BikeBikeAja?
3. Bagaimana membangun sistem keamanan untuk server IoT?

1.3 Batasan Masalah

Sistem BikeBikeAja merupakan sistem penguci sepeda pintar yang menyediakan fitur keamanan yang terkoneksi dengan jaringan internet dengan sistem IoT yang berfungsi untuk meningkatkan keamanan sepeda dan memonitoring posisi sepeda ketika digunakan. Adapun ruang lingkup dari penelitian ini membahas konektivitas server IoT menggunakan protokol MQTT.

1.4 Tujuan Pengembangan

Pengembangan dari pembahasan pada penelitian ini bertujuan untuk:

1. Mengetahui jenis server apa yang ideal digunakan pada perangkat IoT BikeBikeAja .
2. Memastikan aksesibilitas dan stabilisasi serta keamanan data yang tersimpan di server agar mudah diakses dan aman ketika digunakan oleh aplikasi.

BAB II

KAJIAN PUSTAKA

2.1 Kajian Teori

Pada penelitian yang akan dilakukan terdapat informasi untuk menjelaskan teori yang digunakan pada setiap komponen sebagai referensi dalam pengembangan sistem agar dapat meningkatkan keberhasilan dalam pengimplementasian produk. Ada beberapa teori yang akan dikaji pada pembahasan ini seperti Mikrokontroler IoT, Server, Protokol MQTT Keamanan IoT.

2.1.1 Mikrokontroler

Mikrokontroler ESP32 merupakan sebuah mikrokontroler *System on Chip* (SoC) yang terintegrasi dengan WiFi 802.11 b/g/n, Bluetooth versi 4.2, dan berbagai peripheral lainnya. Mikrokontroler ini memiliki kemampuan Wi-Fi dan Bluetooth, serta dilengkapi dengan berbagai fitur seperti ADC (*analog-to-digital converter*), DAC (*digital-to-analog converter*), PWM (*pulse width modulation*), I2C, SPI, dan UART. ESP32 juga memiliki dua inti prosesor dengan kecepatan hingga 240 MHz dan dapat dioperasikan pada rentang tegangan 2.2V hingga 3.6V. Chip ESP32 cukup lengkap karena memiliki prosesor, penyimpanan, dan akses pada GPIO (General Purpose Input Output). Selain itu, ESP32 juga dilengkapi dengan memori internal dan eksternal yang besar, termasuk 512KB RAM dan 4MB flash. Mikrokontroler ini dapat dioperasikan pada suhu yang sangat ekstrem, yaitu dari -40°C hingga 125°C. Dalam pengembangan aplikasi *Internet of Things* (IoT), ESP32 sering digunakan sebagai platform karena kemampuannya dalam terkoneksi dengan jaringan Wi-Fi dan Bluetooth. Selain itu, mikrokontroler ini juga bisa dikonfigurasi menjadi *Access Point* atau *Station*[5].



Gambar 2. 1 ESP32 Wrover E[6]

Tabel 2. 1 Spesifikasi ESP32 Wrover E

Komponen	Deskripsi
CPU	Dual-core Tensilica LX6 microprocessor
Kecepatan CPU	240 MHz
Kapasitas Memori	8 MB (external)
Kapasitas Flash	4 MB
Wi-Fi	802.11 b/g/n
Bluetooth	Bluetooth v4.2 BR/EDR and BLE specification
Antena Wi-Fi	PCB antenna
Antena Bluetooth	External antenna
GPIO Pins	38
Konsumsi Daya	2.5 μ A dalam mode deep-sleep, 20 mA pada saat active Wi-Fi
Operating Voltage	2.7V-3.6V
Temperature	-40°C to +85°C
Package	38-pin QFN package
Dimensi	5.5 mm \times 5.5 mm

2.1.2 GSM SIM 7600 G

GSM SIM7600 merupakan modul komunikasi seluler yang mendukung jaringan 4G LTE. Modul ini dirancang oleh SIMCom dan memiliki ukuran yang relatif kecil, sehingga cocok untuk diintegrasikan ke dalam sistem elektronik yang membutuhkan konektivitas seluler. Modul SIM7600 mendukung berbagai

frekuensi seluler, termasuk GSM, GPRS, EDGE, WCDMA, HSDPA, HSPA+, FDD-LTE, dan TDD-LTE. Modul ini juga mendukung GPS, GLONASS, BeiDou, dan QZSS untuk menambah kemampuan lokasi[7].

Modul SIM7600 memiliki antarmuka yang mudah digunakan, yaitu melalui antarmuka serial (UART) yang dapat dihubungkan dengan mikrokontroler atau komputer. Modul ini dapat dikonfigurasi dan dikontrol dengan perintah AT Command, sehingga memudahkan dalam integrasi ke dalam sistem yang sudah ada.



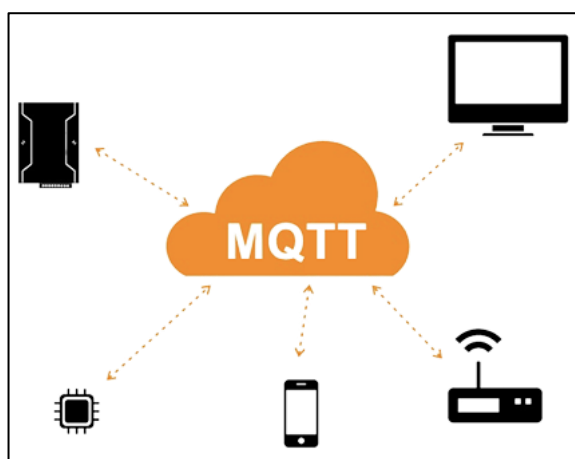
Gambar 2. 2 SIM7600G-H

Tabel 2. 2 Spesifikasi GSM SIM 7600 G-H

Spesifikasi	Deskripsi
Model	SIM7600E
Antarmuka	Mini-PCIE
Teknologi	GSM/GPRS/EDGE, UMTS/HSPA, LTE Cat4
Jaringan	2G/3G/4G
Frekuensi	GSM/GPRS/EDGE: 850/900/1800/1900MHz
	UMTS/HSPA: 850/900/1900/2100MHz
	LTE FDD: B1/B3/B5/B7/B8/B20
	LTE TDD: B38/B40/B41
Kecepatan	Hingga 150 Mbps (unduh) dan 50 Mbps (unggah) pada LTE Cat4
Dimensi	30 x 50 x 4,9 mm
Suhu Operasi	-40°C hingga +85°C
Tegangan	3,3 VDC
Konsumsi Daya	Maks. 2,5 W saat terhubung dengan jaringan 4G
Antena	Antena PCB Internal

2.1.3 MQTT

Perusahaan IBM menemukan sebuah protokol perpesanan yang dikenal sebagai MQTT (Message Queue Transportasi Telemetri) yang cocok untuk IoT. Dalam Model OSI berbasis TCP/IP, ini adalah lapisan aplikasi protokol memiliki overhead yang sangat ringan seperti yang dimilikinya ukuran tajuk tetap 2 byte. Ini berbasis OASIS standar. MQTT mengikuti arsitektur asimetris. Karena ini adalah protokol yang ringan, jadi terlepas dari IoT juga cocok untuk M2M (mesin ke mesin) & WSN (Jaringan Sensor Nirkabel)[9]. Penerbit dan pelanggan dapat dianggap sebagai MQTT klien, sedangkan Broker adalah server MQTT.



Gambar 2. 3 Sistematis MQTT[8]

Tabel 2. 3 Spesifikasi MQTT Broker

Spesifikasi MQTT	Deskripsi
Nama	Mosquitto
Versi	2.0.12
Protokol	MQTT versi 3.1 dan 3.1.1
Platform	Berjalan pada sistem operasi Linux, macOS, dan Windows
Bahasa	Ditulis dalam bahasa C
Mode	Dapat berjalan sebagai daemon, layanan, atau aplikasi yang berjalan pada terminal
Keamanan	Dapat diintegrasikan dengan Transport Layer Security (TLS) dan otentikasi pengguna dengan username dan password atau sertifikat
Skalabilitas	Dapat menangani ribuan koneksi dan pesan secara simultan
Konfigurasi	Konfigurasi file yang mudah dibaca dan dapat diatur melalui perintah command-line

Spesifikasi MQTT	Deskripsi
Integrasi	Mendukung integrasi dengan sistem manajemen pesan (message queueing) seperti Apache Kafka, Apache Storm, dan Apache NiFi
Lisensi	Open-source dengan lisensi Eclipse Public License v2.0

2.1.4 Cloud Computing

Menurut Barrie Sosinsky [9], cloud computing mengacu pada aplikasi dan layanan yang beroperasi pada jaringan terdistribusi dengan menggunakan sumber daya yang di-virtualisasi dan diakses melalui protokol internet serta standar jaringan yang umum. Dalam cloud computing, sumber daya tersebut bersifat virtual dan tidak terbatas, sementara rincian fisik dari sistem di mana perangkat lunak berjalan diabstraksikan dari pengguna. Cloud computing memiliki potensi untuk mengubah kehidupan kita di berbagai cara dalam beberapa tahun mendatang.

Model Layanan atau Model Layanan Jasa menggambarkan jenis layanan yang disediakan oleh penyedia. Beberapa model layanan yang terkenal adalah: *Software as a Service* (SaaS), *Platform as a Service* (PaaS), dan *Infrastructure as a Service* (IaaS) yang dikenal sebagai model SPI. Model layanan ini saling terkait dan menentukan apa yang harus dikelola oleh penyedia layanan dan apa yang menjadi tanggung jawab klien. Pertumbuhan internet dan perkembangan perusahaan jasa besar telah memungkinkan skala besar sistem cloud computing.

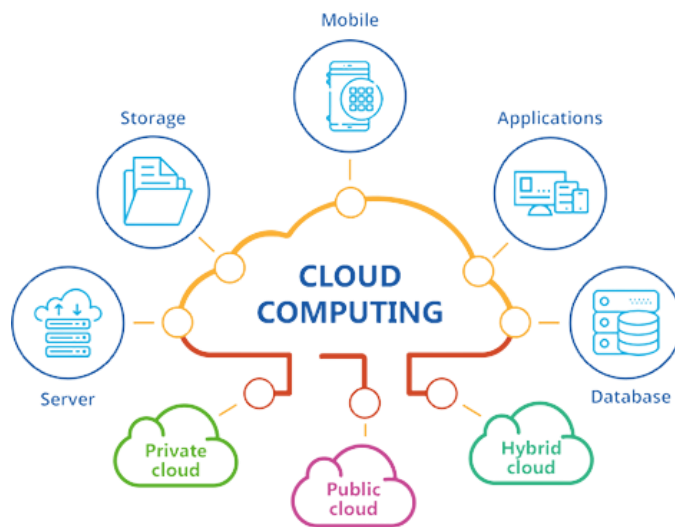
1. *Software as a Service* (SaaS) adalah kemampuan untuk menggunakan aplikasi yang dijalankan pada infrastruktur cloud tanpa perlu mengelola infrastruktur tersebut. Aplikasi dapat diakses melalui berbagai perangkat dan antarmuka pengguna seperti web browser. Konsumen tidak perlu mengelola jaringan, server, sistem operasi, penyimpanan, atau kemampuan aplikasi individu, kecuali dalam beberapa konfigurasi khusus.
2. *Platform as a Service* (PaaS) adalah kemampuan untuk mengembangkan dan menjalankan aplikasi pada infrastruktur cloud dengan menggunakan bahasa pemrograman, perpustakaan, layanan, dan alat yang didukung oleh penyedia. Konsumen tidak perlu mengelola infrastruktur *cloud* seperti

jaringan, server, sistem operasi, namun memiliki kontrol melalui aplikasi yang dikembangkan dan konfigurasi hosting aplikasi.

3. *Infrastructure as a Service (IaaS)* adalah kemampuan untuk mengontrol proses, penyimpanan, jaringan, dan sumber daya komputasi di mana konsumen dapat mengembangkan dan menjalankan perangkat lunak sesuai kebutuhan, termasuk sistem operasi dan aplikasi. Konsumen tidak perlu mengelola infrastruktur cloud yang mendasarinya, namun memiliki kontrol terbatas melalui sistem operasi, penyimpanan, aplikasi, dan komponen jaringan seperti firewall host.[10]

Fungsi *Cloud Computing* secara umum dilakukan oleh sebuah komputer adalah

- a. menyimpan aplikasi dan database yang di butuhkan oleh komputer yang terhubung.
- b. menyediakan fitur keamanan computer.
- c. melindungi semua komputer yang terhubung menggunakan firewall.
- d. menyediakan IP Address untuk mesin komputer terhubung.



Gambar 2. 4 Cloud Computing[11]

Tabel 2. 4 Spesifikasi Server

Spesifikasi	Detail
Sistem Operasi	Ubuntu 20.04 LTS
CPU	1 core
RAM	2 GB

Spesifikasi	Detail
Penyimpanan	20 GB SSD
Bandwidth	2 TB
Koneksi Jaringan	1 Gbps

2.1.5 IoT

Internet of Things merupakan suatu konsep dimana suatu perangkat dapat mempunyai kemampuan dalam hal komunikasi jaringan internet, seperti proses pentransferan data tanpa adanya proses komunikasi yang dilakukan antar manusia (manusia ke manusia) maupun antar manusia ke perangkat sistem seperti komputer atau sebuah mikrokontroler. Dengan teknologi *Internet of Things* ini proses kerja sebuah sistem dapat dilakukan semangkin luas, jarak jangkauannya juga semangkin luas, proses pengolahan data dan analisis data terhadap sebuah sistem juga semangkin bagus.[2]



Gambar 2. 2 Ruang Lingkup IoT

Didalam IoT sudah terbentuk sistem embedded yang bertujuan memperluas pemanfaatan dari konektivitas internet yang tersambung secara terus menerus sehingga dapat berkemampuan seperti berbagi data, remote control dan peralatan elektronik yang sensor dan terhubung dengan jaringan. Keterkaitan objek yang terkoneksi dengan internet sebagai dasar pengembangan semua layanan atau benda fisik diintegrasikan ke dalam jaringan informasi secara berkaitan, yang di mana

benda-benda fisik tersebut berperan secara aktif dalam proses bisnis. Tersedia layanan pintar yang saling terkoneksi, mencari dan mengubah status mereka sesuai dengan setiap informasi yang dikaitkan, disamping memperhatikan masalah privasi dan keamanan [12].

Rancang bangun IoT para engineer harus memperhatikan ketiga aspek yaitu ukuran, ruang, dan waktu. Dengan menentukan aspek – aspek tersebut perkembangan IoT akan lebih mudah untuk diterapkan di kehidupan sehari-hari[13].

2.2 Kajian Pengetahuan Pendukung yang Relevan

Penelitian ini mengenai sistem kehandalan pada sepeda rental menggunakan pengunci yang diatur oleh sistem dan *smartlock* dengan menggunakan mikrokontroler sebagai pusat kendalinya selain itu juga terdapat sistem IoT untuk meningkatkan koneksi dan sepeda rental yang sudah terintegrasi dengan aplikasi sebagai antarmuka kepada pengguna. Rancangan sistem terdiri dari beberapa komponen yaitu mikrokontroler ESP32 Wroom E, Telemetri Sim7600 G sebagai konektivitas internet sekaligus sensor GPS sebagai navigasi perangkat yang sudah tertanam di dalam SIM7600. Kemudian juga terdapat buzzer sebagai alarm warning dan motor gear dc sebagai komponen penggerak pengunci pada *smartlock*. Dari penelitian yang sudah dilakukan penggunaan *smartlock* dan GPS tracker merupakan perangkat yang banyak digunakan sebagai perangkat keamanan kendaraan yang dapat dikendalikan dan dipantau dari jarak jauh secara *realtime*.

Derian Pratama¹, Nina Sariana (2019) melakukan penelitian rancang bangun sistem informasi penyewaan kendaraan berbasis web[14]. Penelitian ini memanfaatkan teknologi web sebagai proses sewa menyewa pada kendaraan. Dengan alur proses yang masih terbilang konvensional dan belum mengandalkan sistem IoT untuk mempercepat dalam proses sewa menyewa. Untuk itu pada penelitian ini dapat meningkatkan dari segi segala hal, seperti sewa penyewa, sistem IoT dan sistem server yang digunakan.

Davit Nurhannavi, Fajar Yumono , Putri Nur Rahayu (2021) melakukan rancang bangun alat keamanan sepeda motor berbasis IoT menggunakan nodemcu dan GPS[15]. Penelitian tersebut membuat sistem keamanan sepeda motor dengan

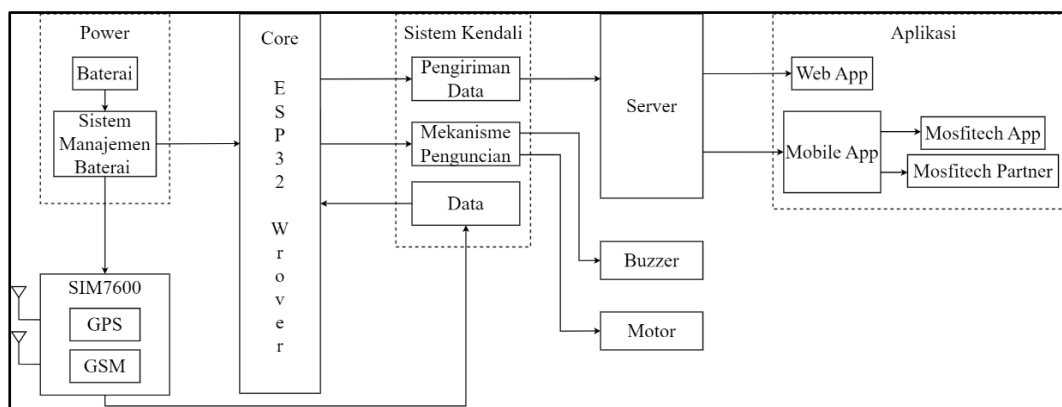
menggunakan nodemcu sebagai mikrokontrolernya. Selain itu sistem tersebut mempunyai relay dengan 4 chanel yang bertujuan untuk memutus dan menghubungkan arus pada motor kendaraan dengan hal tersebut motor dalam keadaan mati dan tidak bisa dipakai sementara. Untuk menghidupkan sepeda motor harus menggunakan aplikasi bylink yang harus terkoneksi dengan wifi. Dari perancangan tersebut memiliki beberapa kekurangan seperti harus menggunakan wifi sebagai komunikasinya yang dimana wifi tersebut dapat retas sehingga belum cukup aman dari segi komunikasinya. Kekurangan selanjutnya ialah untuk mengontrol sepeda motor menggunakan metode ON/OFF relay yang di jumper dengan kabel stopkontak kunci pada sepeda motor, sehingga hal itu belum cukup aman untuk diterapkan dalam keadaan darurat.

BAB III

PERANCANGAN

3.1 Model dan Prosedur Pengembangan

Perancangan sistem BikeBikeAja secara keseluruhan terbagi menjadi tiga sub sistem utama, yaitu sub sistem power supply, sub sistem kendali, sub sistem server dan dan subsistem aplikasi. Arsitektur utama BikeBikeAja ditunjukkan pada Gambar 3.1 di bawah ini.



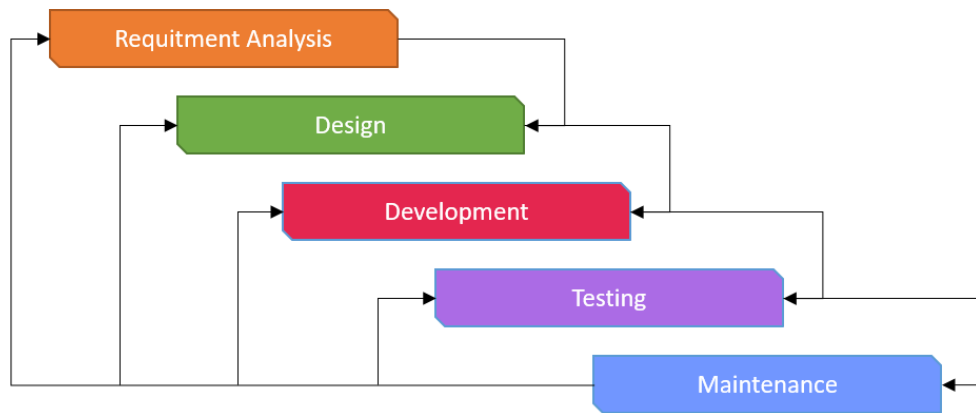
Gambar 3. 1 Arsitektur Utama BikeBikeAja

Mikrokontroler Core ESP32 Wrover sebagai pusat kontrol menerima berbagai informasi dari input dan output yang kemudian data dan informasi proses kerja dari mikrokontroler akan dikirimkan menggunakan modul Gsm Sim7600 G yang telah terkoneksi oleh server melalui protokol MQTT. Sehingga dari semua tersebut akan ditampilkan berupa interface aplikasi mitra dan aplikasi *user*. Perancangan protokol MQTT berguna untuk menjembatani antara Hardware, server dan aplikasi. dengan menggunakan protokol MQTT ini menjamin stabilitas koneksi IoT pada perangkat BikeBikeAja sekaligus keamanan data.

3.1.1 Metodologi Perencanaan

Model pengembangan yang digunakan pada perancangan Server IoT Bike Aja menggunakan protocol MQTT adalah model pengembangan waterfall. Alasan pemilihan model ini adalah karena menurut Sukamto dan Salahuddin, seperti yang dikutip pada, metode waterfall menawarkan alur hidup perangkat lunak berurutan yang dimulai dengan fase analisis, desain, pengkodean, pengujian, dan dukungan.

Metode waterfall terdiri dari lima tahap, yaitu requirement, design, implementation, verification, dan maintenance.



Gambar 3. 2 Metode Wartefall

Tabel 3. 1 Metode Wartefall

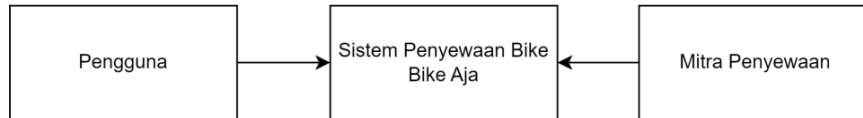
Requitment Analysis	Pada fase ini penulis akan melakukan analisis apa saja yang dibutuhkan dalam perancangan ini serta indentifikasi dan dokumentasi yang diperlukan
Design	Pada fase ini memulai mendesain se ideal mungkin yang sesuai dengan Requitment Analysis sehingga semua yang dibutuhkan akan terpenuhi.
Development	Pada fase ini mulai membuat, merakit ataupun memprogram dengan hasil desain dan memerhatikan kaidah-kaidah yang berlaku
Testing	Pada fase testing ini mulai menguji cobakan, mengimplementasikan dan menerapkan hasil dari fase fase sebelumnya.
Maintenance	Pada fase ini apakah dari hasil testing mengalami kendala atau mengalami kekurangan tertentu, sehingga dengan adanya tahap ini akan memperbaiki hasil dari perancangan.

3.1.2 Data Flow Diagram (DFD)

DFD menjelaskan mengenai hubungan antar subsistem satu dengan subsistem lainnya, DFD memiliki 3 tingkatan, mulai dari DFD konteks untuk tingkat rendah,

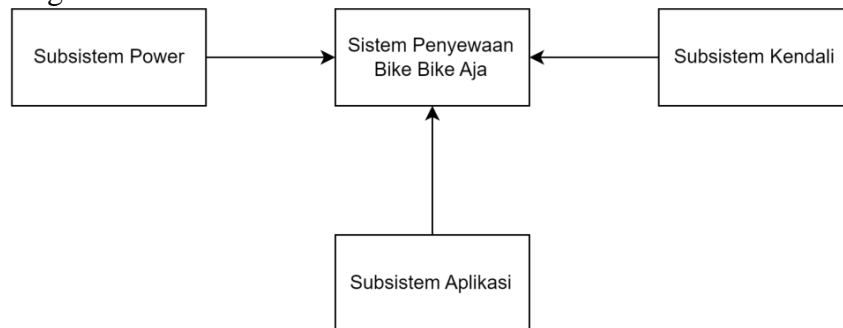
selanjutnya DFD tingkat 1 yaitu lanjutan lebih terperinci dari tingkat sebelumnya dan terakhir tingkat 2 yang menggambarkan perincian proses-proses pada tingkat 1.

1. DFD tingkat 0



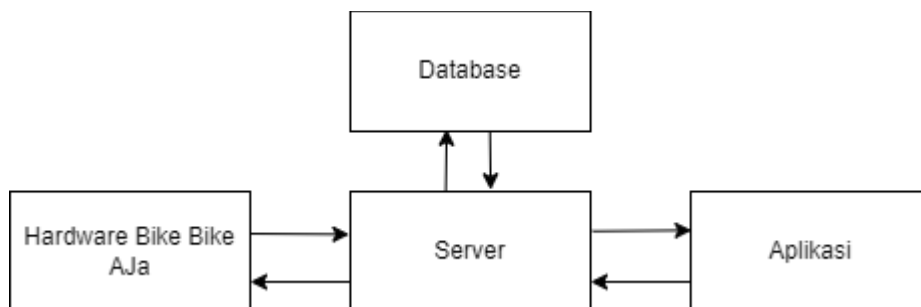
Gambar 3. 3 DFD Tingkat 0

2. DFD Tingkat 1



Gambar 3. 4 DFD Tingkat 1

3. DFD tingkat 2 (Subsistem Server)



Gambar 3. 5 DFD Tingkat 2 Subsistem Server

3.1.3 Penggunaan Rumus

Adapun rumus yang diperlukan pada perancangan ini diantaranya:

3.1.3.1 Pengukuran Throughput

Pengukuran Throughput adalah pengukuran yang dilakukan untuk mengukur jumlah data yang dapat ditransmisikan melalui suatu jaringan atau saluran komunikasi dalam waktu tertentu. Biasanya, pengukuran throughput dilakukan dalam satuan bit per detik (bps) atau kilobit per detik (Kbps), megabit per detik (Mbps), atau gigabit per detik (Gbps), tergantung pada kecepatan transfer data yang diinginkan. Pengukuran ini penting untuk mengetahui seberapa efektif suatu jaringan atau saluran komunikasi dalam mentransmisikan data dalam waktu tertentu[16].

$$Throughput = \frac{n}{time} \quad (1)$$

Keterangan :

n = Jumlah data yang berhasil terkirim

$time$ = Waktu yang dibutuhkan untuk mengirim

3.1.3.2 Pengukuran Delay

Pengukuran delay adalah ukuran waktu yang diperlukan untuk mengirimkan data dari pengirim ke penerima melalui jaringan komunikasi. Delay ini dapat terdiri dari beberapa jenis, seperti delay transmisi (waktu yang diperlukan untuk mengirimkan data melalui media transmisi), delay processing (waktu yang diperlukan untuk memproses data pada node jaringan), dan delay propagation (waktu yang diperlukan untuk sinyal radio atau cahaya dalam menjalar melalui media transmisi). Pengukuran delay umumnya dilakukan dalam satuan waktu seperti detik, milidetik, atau mikrodetik[16].

$$Delay = \frac{t1}{t2} \quad (2)$$

Keterangan:

$t1$ = Waktu Terkirim

$t2$ = Waktu Dikirim

3.1.3.3 Paket Loss

Packet loss adalah kondisi dimana satu atau beberapa paket data hilang selama proses pengiriman data melalui jaringan. Pengukuran packet loss merupakan

metode untuk mengetahui seberapa besar persentase paket data yang hilang atau tidak diterima oleh penerima dalam suatu jaringan[16].

$$Packet\ Loss\ Percentage = \frac{Number\ of\ Lost\ Packets}{Total\ Number\ of\ Packets\ Sent} \times 100\% \quad (3)$$

3.1.4 Timeline Pengerjaan

Berikut adalah timeline pengerjaan dari sub sistem Server.

Tabel 3. 1 Timeline Pengerjaan

Kegiatan dan waktu pelaksanaan	Januari				Februari				Maret				April			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Persiapan																
Pembuatan aplikasi																
Pembuatan server																
Perakitan hardware																
Pengujian																

3.1.5 Spesifikasi Produk

3.1.5.1 Spesifikasi Hardware

- Mikrokontroler ESP32 Wroomer sebagai komponen pemrosesan data input dan output dari sensor atau actuator yang ada pada sistem.
- Sim 7600 G sebagai koneksi jaringan ke *server* dengan sinyal GSM sekaligus komponen untuk memperoleh data koordinat lokasi dari satelit.

3.1.5.2 Spesifikasi Hardware

Pada spesifikasi software pada penelitian ini menggunakan *cloud computing* berupa server linux yang telah terinstal protokol MQTT sebagai protokol komunikasi antara hardware BikeBikeAja dan server. Jenis broker MQTT yang digunakan pada penelitian ini ialah Mosquitto. Broker Mosquitto memiliki fitur seperti dukungan untuk koneksi SSL/TLS, pengaturan hak akses, dan dukungan untuk protokol MQTT versi 5. Mosquitto juga memiliki antarmuka baris perintah untuk memudahkan penggunaan dan konfigurasi.

Fitur yang diimplementasikan pada perangkat BikeBikeAja menggunakan Mosquitto antara lain:

1. **Authentikasi:** Fitur ini memungkinkan pengguna untuk melakukan proses autentikasi atau verifikasi identitas sebelum dapat mengakses dan menggunakan layanan yang disediakan oleh perangkat BikeBikeAja.
2. **QoS (Quality of Service):** Fitur ini mengatur tingkat kualitas layanan yang diberikan oleh Mosquitto. QoS memungkinkan pengiriman pesan dengan prioritas tertentu dan menjamin pengiriman pesan yang handal dan andal.
3. **Publisher dan Subscriber:** Fitur ini memungkinkan perangkat BikeBikeAja untuk berperan sebagai publisher (pengirim) dan subscriber (penerima) pesan MQTT. Dengan fitur ini, perangkat dapat mengirim pesan ke topik tertentu dan juga menerima pesan dari topik yang sama atau berbeda.

Dengan adanya fitur-fitur ini, perangkat BikeBikeAja dapat berkomunikasi secara aman, handal, dan efisien melalui protokol MQTT yang diimplementasikan menggunakan Mosquitto.

3.1.6 Prosedur Pengujian

Pada Pengujian ini bertujuan memastikan bahwasanya Server yang digunakan pada perangkat BikeBikeAja dapat berjalan dengan yang semestinya dan tanpa ada kendala apapun. Berikut merupakan prosedur pengujian yang akan dilakukan pada penelitian ini.

3.1.6.1 Pengujian Konektifitas IoT

Pada Pengujian ini bertujuan mengirimkan data pada perangkat BikeBikeAja menuju server yang sudah terkoneksi internet menggunakan protokol MQTT sebagai jembatan aksesnya. Server akan melakukan proses timbal berupa respon waktu ataupun data losses.. Tabel 3.1.

Tabel 3. 2 Prosedur Pengujian Konektifitas IoT

Pengujian	Prosedur
	<ol style="list-style-type: none"> 1. Pastikan program sudah di <i>upload</i> ke perangkat <i>hardware</i> BikeBikeAja , hidupkan perangkat dengan power supply. 2. Buka sistem <i>server</i> BikeBikeAja menggunakan terminal ssh yang terdapat di <i>cloud computing</i> dan lakukan pengujian ping tes dns dari local.

Konektifitas IoT	3. Lakukan ping sebanyak 20 kali dari computer pengujian ke vps server 4. Analisis lihat seberapa besar paket yang dikirim sekaligus waktu pengiriman. 5. Cek paket Loss jika ada.
------------------	--

3.1.6.2 Pengujian Publisher - Subscriber

Pengujian publisher dan subscriber bertujuan memastikan fungsi yang baik dari sistem MQTT dalam pengiriman dan penerimaan pesan atau data. Pengujian publisher melibatkan pengiriman pesan atau data dari perangkat BikeBikeAja atau aplikasi ke broker MQTT di server. Sementara itu, pengujian subscriber melibatkan penerimaan pesan atau data dari broker MQTT ke perangkat IoT BikeBikeAja atau aplikasi yang terhubung.

Tabel 3. 3 Prosedur Pengujian Publisber dan Subscriber

Pengujian	Prosedur
-----------	----------

Publish - Subciber	<ol style="list-style-type: none"> 1. Melakukan persiapan perangkat keras dan perangkat lunak, termasuk perangkat BikeBikeAja dan broker MQTT yang terkoneksi dengan baik, serta membuat dua atau lebih client untuk publish dan subscribe. 2. Membuat topik untuk mengirimkan pesan dengan nama "topik/rental". 3. Melakukan subscribe pada topik yang sudah dibuat sebelumnya dan memastikan bahwa pesan yang dikirim dari client publish dapat diterima oleh client subscribe. 4. Melakukan publish dengan mengirimkan pesan dari client publish ke topik yang sudah dibuat sebelumnya dan memastikan bahwa pesan berhasil diterima oleh client subscribe. 5. Melakukan pengujian waktu pesan ke topik yang sudah dibuat sebelumnya dan memastikan bahwa pesan tetap tersimpan di broker dan dapat diterima oleh client subscribe bahkan setelah koneksi terputus.
--------------------	--

3.1.6.3 Pengujian Keamanan Server dan MQTT

Pengujian ini bertujuan untuk memperoleh keandalan sistem dengan performa yang baik pada fungsional aplikasi serta memastikan semua fitur dan interface telah berjalan sesuai dengan rancangan yang telah ditentukan. Pengujian ini menggunakan bantuan tools lain atau dengan istilah melakukan penetrasi testing pada disisi server dan clien. Selanjutnya akan dilakukan prosedur pengujian dengan langkah-langkah seperti pada tabel dibawah ini.

Tabel 3. 4 Pengujian Keamanan dan Stabilitas

Pengujian	Prosedur
-----------	----------

Kemanan server dan stabilitas MQTT	<ol style="list-style-type: none"> 1. Implementasikan skenario pengujian yang bisa dilakukan secara manual maupun menggunakan tool-tool keamanan seperti, nmap, dan lain-lain pada Server. 2. Identifikasi risiko keamanan pada implementasi MQTT, seperti pengiriman data yang tidak terenkripsi dan autentikasi oleh MQTT 3. Evaluasi hasil pengujian keamanan, dan lakukan analisis terhadap temuan-temuan keamanan yang ditemukan. Identifikasi penyebab terjadinya kelemahan pada keamanan dan tindak lanjuti temuan-temuan keamanan yang ditemukan. 4. Terakhir, hasil pengujian keamanan dan stabilitas pada implementasi MQTT perlu didokumentasikan dalam screenshot.
------------------------------------	--

3.2 Instrumen dan Teknik Analisis

Teknik analisis yang dilakukan pada penelitian untuk memastikan semua fitur yang ada pada aplikasi BikeBikeAja serta respon perangkat *hardware* dapat berfungsi sesuai dengan tujuan pembuatannya. Bagian pengujian yang diperhatikan dan dianalisis yaitu tampilan aplikasi yang sesuai dengan fungsional dan tujuannya, informasi data yang ditampilkan pada aplikasi telah sesuai dengan data yang ada pada perangkat *hardware* dengan membandingkan data pada perangkat *hardware* dan *server*, lalu pengujian delay pengiriman data dari aplikasi ke perangkat *hardware* dan sebaliknya yang dikirim menggunakan jaringan internet ke *server*. Penggunaan aplikasi juga dapat memonitoring lokasi perangkat *hardware* dari jarak jauh secara *realtime* ketika dibutuhkan untuk melacak posisi sepeda jika terjadi pencurian.

3.3 Perancangan Produk

Produk BikeBikeAja dirancang dengan 4 subsistem utama yaitu *power*,

kendali, server dan aplikasi. Subsistem kendali merupakan bagian dari *hardware* yang dipasangkan di sepeda serta mampu mengontrol sistem pada perangkat BikeBikeAja dan memberikan data lokasi ke *server*. Subsistem *power* berfungsi mensuplai untuk setiap penggunaan modul yang tertanam di dalam *mainboard* utama, serta mengestimasi cukupnya daya dalam penggunaan *hardware* untuk jangka waktu yang lama, Subsistem server dimana subsistem ini berfungsi menjembatani konektivitas dan laju aliran data antara subsistem kendali dan aplikasi menggunakan platform internet. Subsistem aplikasi dimana subsistem ini menjadi antarmuka bagi pengguna maupun mitra untuk melakukan penyewaan maupun menyewakan sepeda nya, subsistem ini memiliki peran untuk memvisualisasikan lokasi setiap sepeda yang disewakan serta mampu melakukan peminjaman sepeda dengan mudah

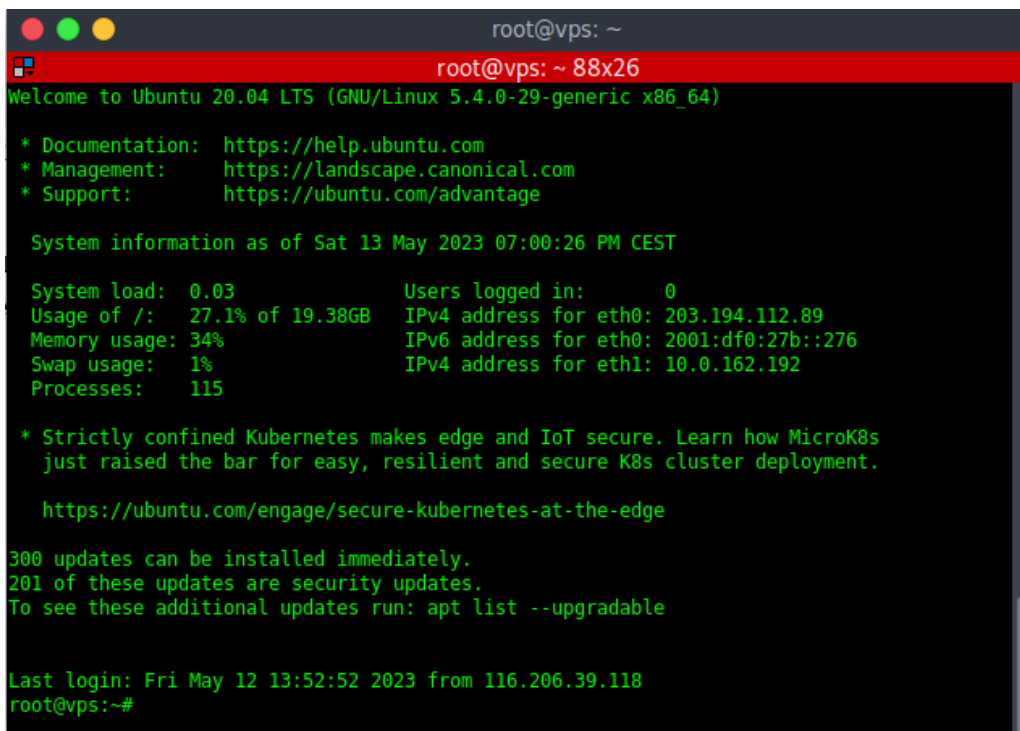
BAB IV HASIL DAN ANALISIS

4.1 Hasil Implementasi dan Produk

Hasil implementasi dan produk dari perancangan server IoT menggunakan protokol MQTT pada perangkat BikeBikeAja diantaranya implementasi server dan implementasi MQTT.

4.1.1 Implementasi Server

Pada implementasi server IoT pada perangkat BikeBikeAja digunakan *cloud computing* dengan sistem operasi Linux, khususnya Ubuntu Server 20.04 LTS. VPS Ubuntu Server ini dilengkapi dengan dukungan dan konfigurasi lain yang diperlukan, seperti *processor*, RAM, SWAP, serta konfigurasi jaringan publik untuk dapat diakses melalui internet, bukan hanya pada jaringan lokal. Berikut ini contoh VPS yang digunakan pada perangkat BikeBikeAja .



```
root@vps: ~
root@vps: ~ 88x26
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat 13 May 2023 07:00:26 PM CEST

System load:  0.03               Users logged in:      0
Usage of /:   27.1% of 19.38GB   IPv4 address for eth0: 203.194.112.89
Memory usage: 34%               IPv6 address for eth0: 2001:df0:27b::276
Swap usage:   1%                IPv4 address for eth1: 10.0.162.192
Processes:    115

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

300 updates can be installed immediately.
201 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Fri May 12 13:52:52 2023 from 116.206.39.118
root@vps:~#
```

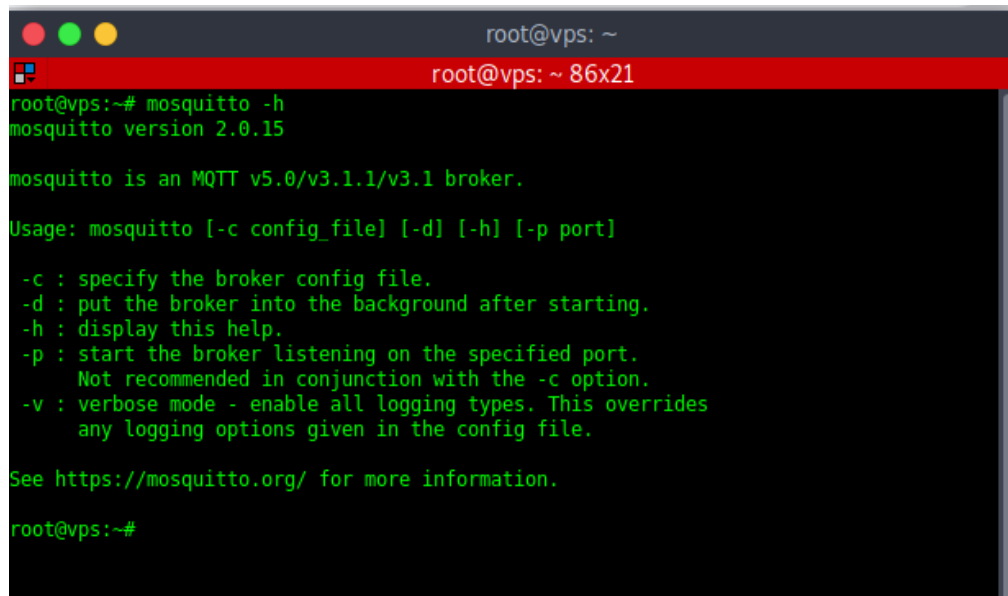
Gambar 4. 1 *Cloud Computing* Ubuntu Server 20.04 LTS

4.1.2 Implementasi MQTT Broker

Pada implementasi MQTT Broker sebagai protocol yang digunakan pada perangkat BikeBikeAja ini dirancang dengan tujuan untuk digunakan pada jaringan dengan bandwidth yang rendah dan koneksi yang tidak stabil seperti jaringan seluler. Sehingga dengan bandwidth yang lebih rendah saja, mqtt dapat mengirim serta menerima data pada perangkat BikeBikeAja. Berikut ini ada beberapa bagian pada MQTT.

4.1.2.1 Version MQTT

Pada penggunaan MQTT ini menggunakan dengan engine atau broker yang digunakan ialah Mosquitto. Mosquitto adalah sebuah broker atau engine yang berfungsi sebagai perantara atau mediator antara perangkat pengirim dan penerima dalam sistem IoT. Mosquitto juga dapat digunakan untuk mengatur tingkat keamanan pada sistem IoT yang menggunakan protokol MQTT, seperti autentikasi pengguna, enkripsi data, dan kontrol akses pada topik-topik tertentu.

A screenshot of a terminal window with a dark background and green text. The window title bar shows 'root@vps: ~' and 'root@vps: ~ 86x21'. The terminal content shows the command 'mosquitto -h' being executed, resulting in the output 'mosquitto version 2.0.15' and a detailed help message. The help message states 'mosquitto is an MQTT v5.0/v3.1.1/v3.1 broker.' and lists usage options: '-c : specify the broker config file.', '-d : put the broker into the background after starting.', '-h : display this help.', '-p : start the broker listening on the specified port. Not recommended in conjunction with the -c option.', and '-v : verbose mode - enable all logging types. This overrides any logging options given in the config file.' It also provides a link to 'https://mosquitto.org/' for more information. The prompt 'root@vps:~#' is visible at the bottom.

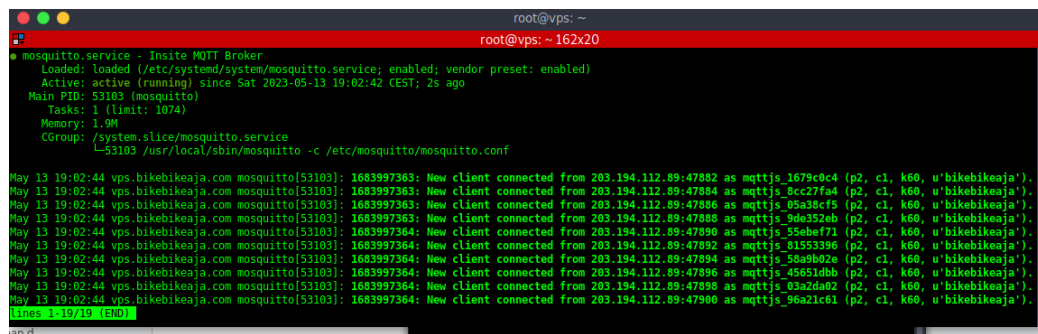
```
root@vps: ~  
root@vps: ~ 86x21  
root@vps:~# mosquitto -h  
mosquitto version 2.0.15  
  
mosquitto is an MQTT v5.0/v3.1.1/v3.1 broker.  
  
Usage: mosquitto [-c config_file] [-d] [-h] [-p port]  
  
-c : specify the broker config file.  
-d : put the broker into the background after starting.  
-h : display this help.  
-p : start the broker listening on the specified port.  
    Not recommended in conjunction with the -c option.  
-v : verbose mode - enable all logging types. This overrides  
    any logging options given in the config file.  
  
See https://mosquitto.org/ for more information.  
root@vps:~#
```

Gambar 4. 2 Versi MQTT Broker Mosquitto.

Versi yang digunakan pada Broker Mosquito ialah versi 2.0.15 dimana versi ini dianggap lebih stabil dan bagus dari segi kewanamanan vulnabiliti.

4.1.2.2 Service MQTT

Service MQTT pada implementasi perangkat BikeBikeAja bertujuan sebagai layanan atau proses berjalan pada broker MQTT yang bertanggung jawab untuk mengatur dan memfasilitasi pengiriman dan penerimaan pesan antara publisher dan subscriber pada sebuah topik. *Service* ini bekerja dengan menerima pesan dari publisher dan kemudian menyebarkan pesan tersebut ke semua subscriber yang terhubung ke topik yang sama. Selain itu, *Service* ini juga memantau koneksi client dan mengelola sesi client, memastikan bahwa koneksi dan sesi client tetap terjaga.



```
root@vps: ~
root@vps: ~ 162x20
mosquitto.service - Insite MQTT Broker
Loaded: loaded (/etc/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
Active: active (running) since Sat 2023-05-13 19:02:42 CEST; 2s ago
Main PID: 53103 (mosquitto)
Tasks: 1 (limit: 1074)
Memory: 1.9M
CGroup: /system.slice/mosquitto.service
└─53103 /usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47882 as mqttjs_1679c0c4 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47884 as mqttjs_8cc27fa4 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47886 as mqttjs_65a38cf5 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47888 as mqttjs_3da352eb (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47890 as mqttjs_35abef71 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47892 as mqttjs_81553396 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47894 as mqttjs_38a9b02e (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47896 as mqttjs_45651dbb (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47898 as mqttjs_03a2da02 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47900 as mqttjs_96a21c61 (p2, c1, k60, u'bikebikeaja').
root@vps: ~ 162x20
```

Gambar 4. 3 *Service* MQTT

4.1.2.3 Konektivitas MQTT

Konektivitas MQTT bertujuan untuk menyediakan protokol yang menghubungkan antara perangkat BikeBikeAja dengan server, sehingga perangkat dapat dikonfigurasi dan diatur menggunakan bahasa pemrograman C/Arduino agar terhubung dengan server. Berikut ini contoh potongan program agar terkoneksi dengan server.

```
boolean mqttConnect()
{
    Serial.print("Connecting to ");
    Serial.print(broker);

    // Connect to MQTT Broker
    // boolean status = mqtt.connect("Active");

    // Or, if you want to authenticate MQTT:
    boolean status = mqtt.connect("Active", mqtt_user, mqtt_pass);

    if (status == false) {
        Serial.println(" fail");
        return false;
    }
    Serial.println(" success");
    mqtt.publish(topicStatus, imei.c_str());
    mqtt.subscribe(topicRental, 1);
    mqtt.subscribe(topicWarning, 0);
    return mqtt.connected();
}
```

```

    if (!mqtt.connected()) {
      Serial.println("MQTT Sedang Koneksi");
      // Reconnect every 10 seconds
      uint32_t t = millis();
      if (t - lastReconnectAttempt > 10000L) {
        lastReconnectAttempt = t;
        if (mqttConnect()) {
          lastReconnectAttempt = 0;
        }
      }
    }
    delay(1000);
    return;
  }
}

```

4.1.2.4 Implementasi Publishser dan Subscriber MQTT

Implementasi publisher dan subscriber MQTT pada perangkat BikeBikeAja dilakukan untuk memastikan bahwa sistem MQTT dapat berfungsi dengan baik dalam mengirim dan menerima pesan atau data berupa pengontrolan dari aplikasi ke perangkat BikeBikeAja. Pada publisher, dilakukan uji coba pengiriman pesan atau data dari perangkat BikeBikeAja atau aplikasi ke broker MQTT yang ada di server, sedangkan pada subscriber, dilakukan uji coba penerimaan pesan atau data dari broker MQTT ke perangkat IoT BikeBikeAja atau aplikasi yang terhubung. Berikut contoh potongan program pada Arduino.

```

// MQTT details
const char *broker = "203.194.112.89";
int port = 1883 ;
const char *mqtt_user = "bikebikeaja"; Topik Publisher dan Subscriber
const char *mqtt_pass = "Bikebike4ja";
const char *topicStatus = "rental/Status";
const char *topicImei = "rental/UUID";
const char *topicRental = "rental/860371050882459";
const char *topicLocations = "rental/location";
const char *topicBattray = "rental/battray";
const char *topicWarning = "rental/warning/860371050882459";
const char *topicTime = "rental/time";

```

Gambar 4. 4 Topik MQTT

```

void mqttCallback(char *topic, byte *payload, unsigned int len)
{
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("]: ");
    Serial.write(payload, len);
    Serial.println();
    // memcpy(receivedPayload, payload, len);
    // receivedPayload[len] = '\0';

    // // Only proceed if incoming message's topic matches

    if (String(topic) == topicRental) {
        if ((char)payload[0] == '1') {
            Serial.println("Sistem Sewa Aktif");
            mqtt.publish(topicStatus, "Sistem Sewa Aktif");
            digitalWrite(Dinamo, HIGH);
            digitalWrite(Buzzer, HIGH);
            delay(50);
            digitalWrite(Buzzer, LOW);
            delay(50);
            digitalWrite(Buzzer, HIGH);
            delay(50);
            digitalWrite(Buzzer, LOW);
            delay(50);
            delay(903);
            digitalWrite(Dinamo, LOW);
            digitalWrite(Buzzer, LOW);
        } else if ((char)payload[0] == '0') {
            digitalWrite(Dinamo, LOW);
            Serial.println("Dinamo turned off");
            mqtt.publish(topicStatus, "Waktu Habis");
            Buzzer_Blink();
        }
    }

    else if (String(topic) == topicWarning) {
        if ((char)payload[0] == '1') {
            Serial.println("Sistem Sewa Aktif");
            mqtt.publish(topicStatus, "Diluar Area");
            digitalWrite(Buzzer, HIGH);
            delay(100);
            digitalWrite(Buzzer, LOW);
            delay(100);
            digitalWrite(Buzzer, HIGH);
            delay(100);
            digitalWrite(Buzzer, LOW);
            delay(200);
        } else if ((char)payload[0] == '0') {
            digitalWrite(Buzzer, LOW);
            Serial.println("Dinamo turned off");
            mqtt.publish(topicStatus, "Didalan Area");
        }
    }
}

```

Gambar 4. 5 Implementasi Publisher dan Subscriber

4.2 Hasil Pengujian

4.2.1 Pengujian Konektifitas IoT

Pada pengujian konektifitas IoT perangkat BikeBikeAja bertujuan untuk memastikan bahwa konektifitas IoT berjalan lancar dengan dilakukan dua metode.

Metode yang pertama dilakukan sebuah ping terhadap IP host server, dimana ping tersebut bertujuan menguji kekuatan jaringan yang ada di server sehingga dapat memastikan bahwa jaringan berjalan cukup stabil dan diharapkan tidak terdapat paket loss. Berikut hasil pengujian ping

```

$ping 203.194.112.89
PING 203.194.112.89 (203.194.112.89) 56(84) bytes of data.
64 bytes from 203.194.112.89: icmp_seq=1 ttl=54 time=45.7 ms
64 bytes from 203.194.112.89: icmp_seq=2 ttl=54 time=43.8 ms
64 bytes from 203.194.112.89: icmp_seq=3 ttl=54 time=40.1 ms
64 bytes from 203.194.112.89: icmp_seq=4 ttl=54 time=50.0 ms
64 bytes from 203.194.112.89: icmp_seq=5 ttl=54 time=32.9 ms
64 bytes from 203.194.112.89: icmp_seq=6 ttl=54 time=32.2 ms
64 bytes from 203.194.112.89: icmp_seq=7 ttl=54 time=30.0 ms
64 bytes from 203.194.112.89: icmp_seq=8 ttl=54 time=39.1 ms
64 bytes from 203.194.112.89: icmp_seq=9 ttl=54 time=39.7 ms
64 bytes from 203.194.112.89: icmp_seq=10 ttl=54 time=45.7 ms
64 bytes from 203.194.112.89: icmp_seq=11 ttl=54 time=98.6 ms
64 bytes from 203.194.112.89: icmp_seq=12 ttl=54 time=33.5 ms
64 bytes from 203.194.112.89: icmp_seq=13 ttl=54 time=40.8 ms
64 bytes from 203.194.112.89: icmp_seq=14 ttl=54 time=48.8 ms
64 bytes from 203.194.112.89: icmp_seq=15 ttl=54 time=48.6 ms
64 bytes from 203.194.112.89: icmp_seq=16 ttl=54 time=48.0 ms
64 bytes from 203.194.112.89: icmp_seq=17 ttl=54 time=35.8 ms
64 bytes from 203.194.112.89: icmp_seq=18 ttl=54 time=44.8 ms
64 bytes from 203.194.112.89: icmp_seq=19 ttl=54 time=42.9 ms
64 bytes from 203.194.112.89: icmp_seq=20 ttl=54 time=47.7 ms
^C
--- 203.194.112.89 ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19026ms
rtt min/avg/max/mdev = 29.988/44.441/98.631/13.791 ms

```

Gambar 4. 6 Pengujian Ping

Dapat dilihat pada Gambar 4.1, pengujian Ping dilakukan dengan mengirimkan paket transmitter pada host IP 203.194.112.89 dengan ICMP range 1 sampai 20. Tabel di atas menunjukkan hasil pengujian, yang terdiri dari kolom icmp_seq, ttl, dan time.

Tabel 4. 1 Ping test

icmp_seq	bytes	time
1	64	45.7
2	64	43.8
3	64	40.1
4	64	50
5	64	32.9
6	64	32.2
7	64	30
8	64	39.1
9	64	39.7

icmp_seq	bytes	time
10	64	45.7
11	64	98.6
12	64	33.5
13	64	40.8
14	64	48.8
15	64	48.6
16	64	48
17	64	35.8
18	64	44.8
19	64	42.9
20	64	47.7

Data tersebut menunjukkan kecepatan jaringan yang ada di dalam server dengan rata-rata kecepatan 44.441 ms, kecepatan minimal 29.988 ms, dan kecepatan maksimal 98.631 ms. Untuk mengetahui nilai *throughput* yang didapatkan kita bisa menggunakan rumus :

$$Throughput = \frac{n}{t}$$

Keterangan :

n = Jumlah data yang berhasil terkirim

t = Waktu yang dibutuhkan untuk mengirim

Maka dapat dimasukan nilai jumlah dan rata waktu yang dibutuhkan untuk mengirimkan sebagai berikut:

$$Throughput = \frac{20}{44.441 \text{ ms}}$$

$$= 0.4500 \text{ bytes/ms atau } 450.0450 \text{ bytes/s}$$

Maka trougput yang didapatkan ialah 450.0450 bps (bit per detik)

Untuk mengetahui nilai presetasi paket loss dengan menggunakan rumus sebagai berikut :

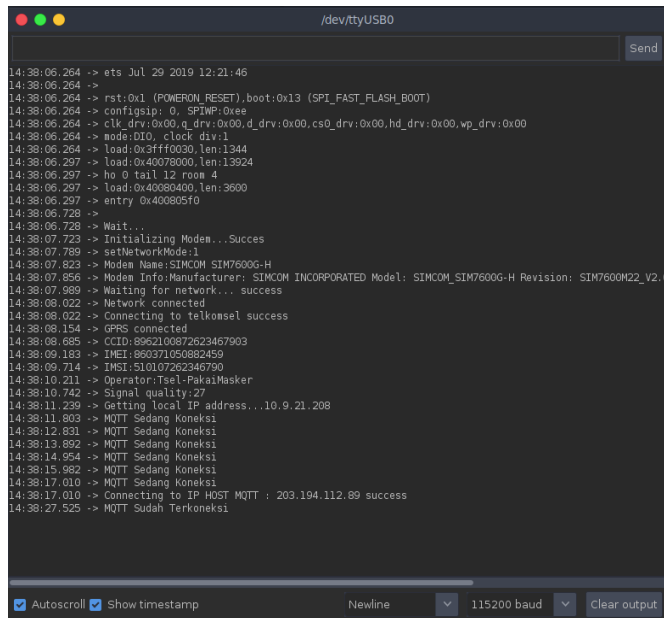
$$Packet Loss Percentage = \frac{Number of Lost Packets}{Total Number of Packets Sent} \times 100\%$$

$$Packet Loss Percentage = \frac{0}{20} \times 100\%$$

$$Packet Loss Percentage = 0\%$$

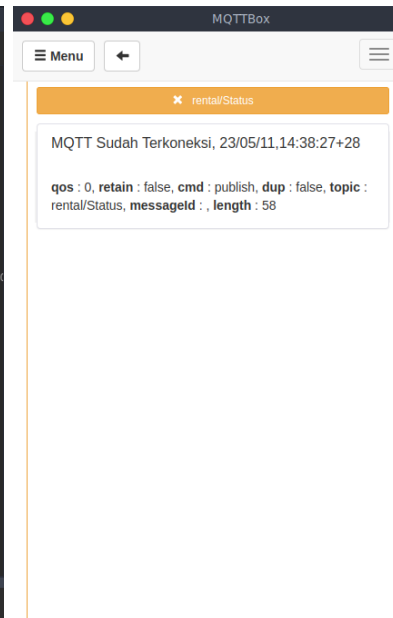
Hasil tersebut menunjukkan bahwa kecepatan jaringan rata-rata stabil dan tidak terdapat paket loss atau paket yang terbuang.

Untuk pengujian konektivitas berikutnya ialah konektivitas dari perangkat BikeBikeAja ke Server.



```
/dev/ttyUSB0
14:38:06.264 -> ets Jul 29 2019 12:21:46
14:38:06.264 ->
14:38:06.264 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
14:38:06.264 -> config:0: 0, SPIWP:0xee
14:38:06.264 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
14:38:06.264 -> mode:DIO, clock div:1
14:38:06.264 -> load:0x3ffff000, len:1344
14:38:06.297 -> load:0x40078000, len:13924
14:38:06.297 -> h0 0 tail 12 row 4
14:38:06.297 -> load:0x40080400, len:3600
14:38:06.297 -> entry 0x400805f0
14:38:06.728 ->
14:38:06.728 -> Wait...
14:38:07.723 -> Initializing Modem...Success
14:38:07.789 -> setNetworkMode:1
14:38:07.823 -> Modem Name:SIMCOM SIM7600G-H
14:38:07.856 -> Modem Info:Manufacturer: SIMCOM INCORPORATED Model: SIMCOM_SIM7600G-H Revision: SIM7600M22_V2.0
14:38:07.989 -> Waiting for network... success
14:38:08.022 -> Network connected
14:38:08.022 -> Connecting to telkomsel success
14:38:08.154 -> GPRS connected
14:38:08.685 -> CCID:8962100872623467903
14:38:09.183 -> IMEI:860371050882450
14:38:09.714 -> IMEI:810107262346790
14:38:10.211 -> Operator:Tsel-PakaiMasker
14:38:10.742 -> Signal quality:27
14:38:11.239 -> Getting local IP address...10.9.21.208
14:38:11.803 -> MQTT Sedang Koneksi
14:38:12.801 -> MQTT Sedang Koneksi
14:38:13.892 -> MQTT Sedang Koneksi
14:38:14.954 -> MQTT Sedang Koneksi
14:38:15.982 -> MQTT Sedang Koneksi
14:38:17.010 -> MQTT Sedang Koneksi
14:38:17.010 -> Connecting to IP HOST MQTT : 203.194.112.89 success
14:38:27.525 -> MQTT Sudah Terkoneksi
```

Gambar 4. 8 Konektivitas perangkat Bike Bike Aja

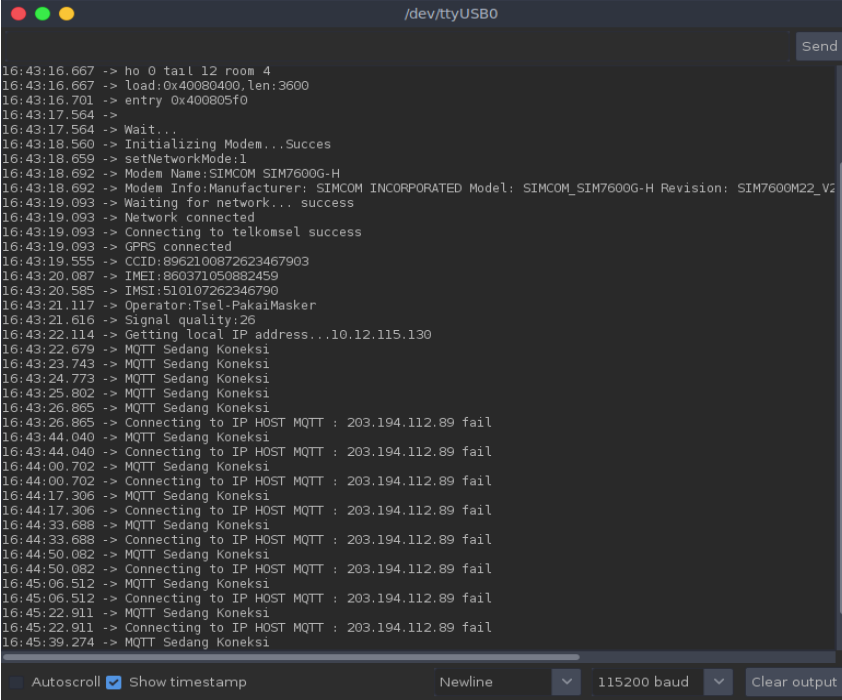


Gambar 4. 7 Pesan yang diterima di server

Dari gambar yang disajikan, terlihat bahwa saat perangkat BikeBikeAja mencoba terkoneksi ke server, status server dapat dilihat pada gambar tersebut. Koneksi berhasil terjalin pada rentang waktu antara 14.38.10 hingga 14.38.27 dengan selisih waktu sebesar 17 ms, kemudian pesan akan terkirim ke server dan ditampilkan pada MQTTBox. Dari hasil pengujian tersebut, dapat disimpulkan bahwa konektivitas IoT pada server menggunakan protokol mqtt berjalan dengan sangat cepat dan stabil untuk server IoT pada perangkat BikeBikeAja, sehingga dengan cepatnya proses transfer data atau pengiriman pesan yang semakin cepat akan mempermudah akseibilitas IoT pada perangkat tersebut

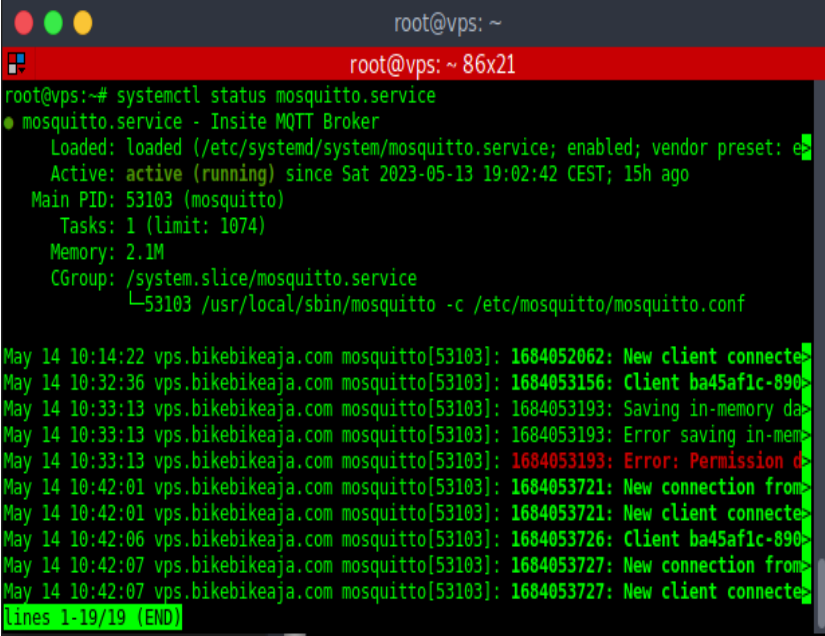
Ketika Pengujian MQTT dalam keadaan fail hal ini menyebabkan tidak terkoneksiya perangkat BikeBikeAja ke *server*. Hal ini dikarenakan beberapa hal

seperti konfigurasi yang salah atau *service* yang berjalan pada MQTT tidak berjalan normal. Untuk itu perlu adanya maintenance untuk memperbaiki hal tersebut.



```
/dev/ttyUSB0
16:43:16.667 -> ho 0 tail 12 room 4
16:43:16.667 -> load:0x40080400, len:3600
16:43:16.701 -> entry 0x400805f0
16:43:17.564 ->
16:43:17.564 -> Wait...
16:43:18.560 -> Initializing Modem...Success
16:43:18.659 -> setNetworkMode:1
16:43:18.692 -> Modem Name:SIMCOM SIM7600G-H
16:43:18.692 -> Modem Info:Manufacturer: SIMCOM INCORPORATED Model: SIMCOM_SIM7600G-H Revision: SIM7600M22_V2
16:43:19.093 -> Waiting for network... success
16:43:19.093 -> Network connected
16:43:19.093 -> Connecting to telkomsel success
16:43:19.093 -> GPRS connected
16:43:19.555 -> CCID:8962100872623467903
16:43:20.087 -> IMEI:860371050882459
16:43:20.585 -> IMSI:510107262346790
16:43:21.117 -> Operator:Tsel-PakaiMasker
16:43:21.616 -> Signal quality:26
16:43:22.114 -> Getting local IP address...10.12.115.130
16:43:22.679 -> MQTT Sedang Koneksi
16:43:23.743 -> MQTT Sedang Koneksi
16:43:24.773 -> MQTT Sedang Koneksi
16:43:25.802 -> MQTT Sedang Koneksi
16:43:26.865 -> MQTT Sedang Koneksi
16:43:26.865 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:43:44.040 -> MQTT Sedang Koneksi
16:43:44.040 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:00.702 -> MQTT Sedang Koneksi
16:44:00.702 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:17.306 -> MQTT Sedang Koneksi
16:44:17.306 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:33.688 -> MQTT Sedang Koneksi
16:44:33.688 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:50.082 -> MQTT Sedang Koneksi
16:44:50.082 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:06.512 -> MQTT Sedang Koneksi
16:45:06.512 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:22.911 -> MQTT Sedang Koneksi
16:45:22.911 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:39.274 -> MQTT Sedang Koneksi
```

Gambar 4. 9 MQTT Fail



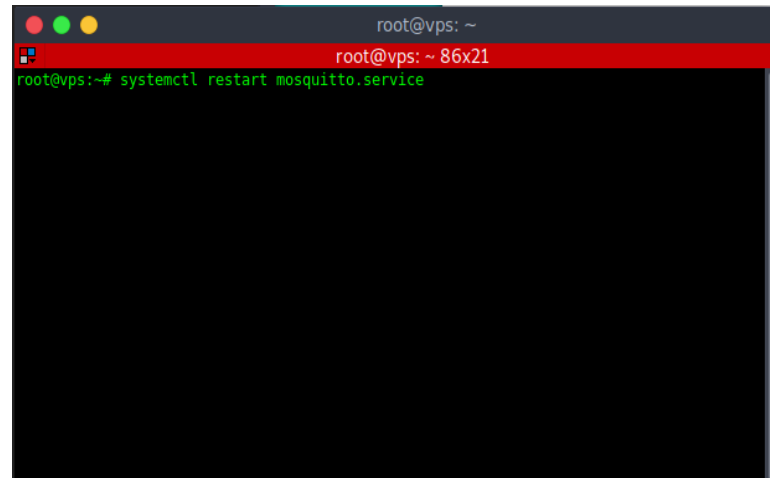
```
root@vps: ~
root@vps: ~ 86x21
root@vps:~# systemctl status mosquitto.service
● mosquitto.service - Insite MQTT Broker
   Loaded: loaded (/etc/systemd/system/mosquitto.service; enabled; vendor preset: ena
   Active: active (running) since Sat 2023-05-13 19:02:42 CEST; 15h ago
     Main PID: 53103 (mosquitto)
       Tasks: 1 (limit: 1074)
      Memory: 2.1M
    CGroup: /system.slice/mosquitto.service
            └─53103 /usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

May 14 10:14:22 vps.bikebikeaja.com mosquitto[53103]: 1684052062: New client connecte
May 14 10:32:36 vps.bikebikeaja.com mosquitto[53103]: 1684053156: Client ba45af1c-890
May 14 10:33:13 vps.bikebikeaja.com mosquitto[53103]: 1684053193: Saving in-memory da
May 14 10:33:13 vps.bikebikeaja.com mosquitto[53103]: 1684053193: Error saving in-mem
May 14 10:33:13 vps.bikebikeaja.com mosquitto[53103]: 1684053193: Error: Permission d
May 14 10:42:01 vps.bikebikeaja.com mosquitto[53103]: 1684053721: New connection from
May 14 10:42:01 vps.bikebikeaja.com mosquitto[53103]: 1684053721: New client connecte
May 14 10:42:06 vps.bikebikeaja.com mosquitto[53103]: 1684053726: Client ba45af1c-890
May 14 10:42:07 vps.bikebikeaja.com mosquitto[53103]: 1684053727: New connection from
May 14 10:42:07 vps.bikebikeaja.com mosquitto[53103]: 1684053727: New client connecte
lines 1-19/19 (END)
```

Gambar 4. 10 Service MQTT

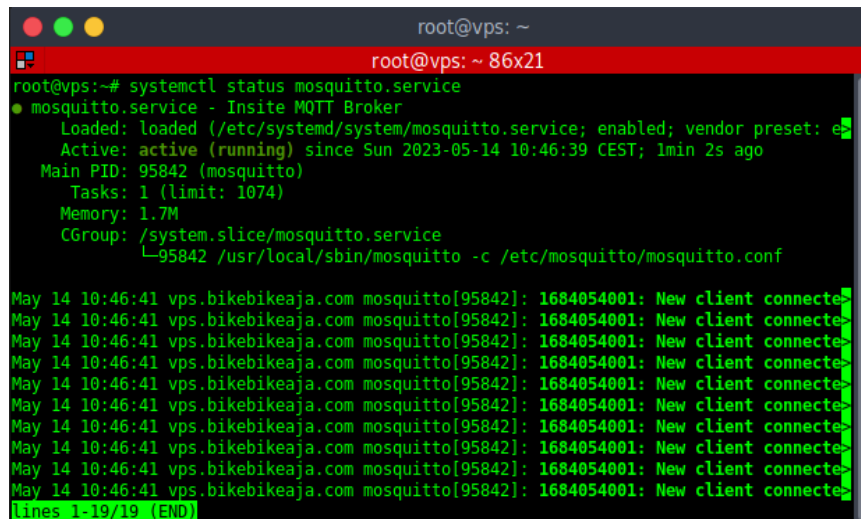
Bermasalah

Untuk memperbaiki permasalahan atau maintenance hal tersebut terdapat beberapa cara seperti gambar di bawah ini.

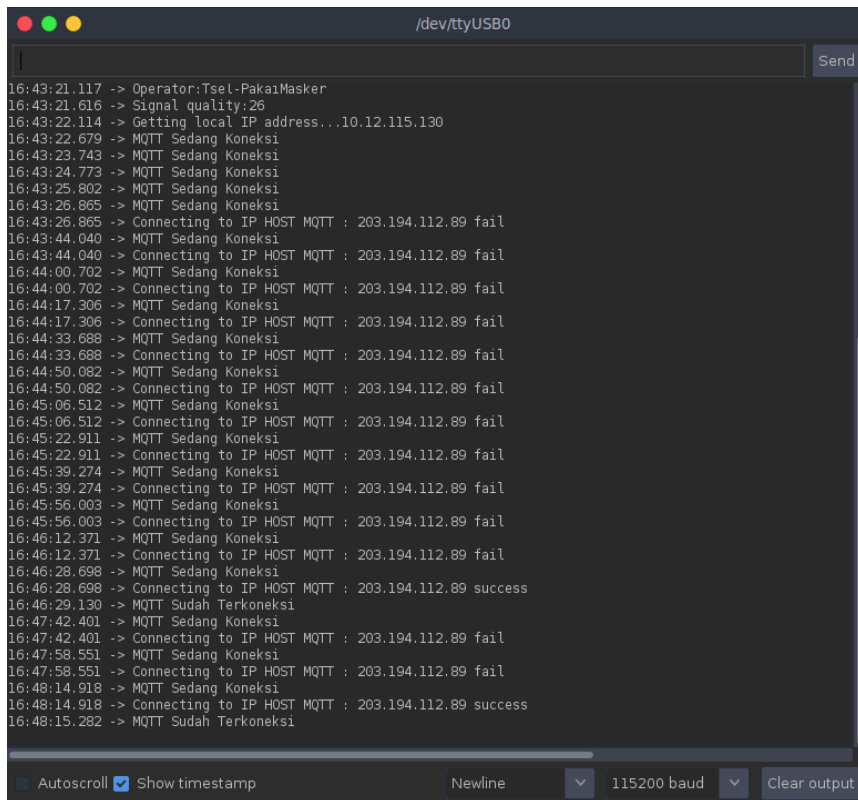


Gambar 4. 11 Maintenance *Service* MQTT

Dengan menggunakan perintah "systemctl restart mosquito service", *Service* yang berjalan pada MQTT akan direset. Dari hasil tersebut, kita dapat memeriksa status MQTT menggunakan perintah "systemctl status mosquito service", dan hasilnya dapat dilihat pada gambar yang terlampir.



Gambar 4. 12 Status Mosquitto



```
16:43:21.117 -> Operator:Tsel-PakaiMasker
16:43:21.616 -> Signal quality:26
16:43:22.114 -> Getting local IP address...10.12.115.130
16:43:22.679 -> MQTT Sedang Koneksi
16:43:23.743 -> MQTT Sedang Koneksi
16:43:24.773 -> MQTT Sedang Koneksi
16:43:25.802 -> MQTT Sedang Koneksi
16:43:26.865 -> MQTT Sedang Koneksi
16:43:26.865 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:43:44.040 -> MQTT Sedang Koneksi
16:43:44.040 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:00.702 -> MQTT Sedang Koneksi
16:44:00.702 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:17.306 -> MQTT Sedang Koneksi
16:44:17.306 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:33.688 -> MQTT Sedang Koneksi
16:44:33.688 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:44:50.082 -> MQTT Sedang Koneksi
16:44:50.082 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:06.512 -> MQTT Sedang Koneksi
16:45:06.512 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:22.011 -> MQTT Sedang Koneksi
16:45:22.011 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:39.274 -> MQTT Sedang Koneksi
16:45:39.274 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:45:56.003 -> MQTT Sedang Koneksi
16:45:56.003 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:46:12.371 -> MQTT Sedang Koneksi
16:46:12.371 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:46:28.698 -> MQTT Sedang Koneksi
16:46:28.698 -> Connecting to IP HOST MQTT : 203.194.112.89 success
16:46:29.130 -> MQTT Sudah Terkoneksi
16:47:42.401 -> MQTT Sedang Koneksi
16:47:42.401 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:47:58.551 -> MQTT Sedang Koneksi
16:47:58.551 -> Connecting to IP HOST MQTT : 203.194.112.89 fail
16:48:14.918 -> MQTT Sedang Koneksi
16:48:14.918 -> Connecting to IP HOST MQTT : 203.194.112.89 success
16:48:15.282 -> MQTT Sudah Terkoneksi
```

Gambar 4. 13 Perangkat Bike Bike Terkoneksi



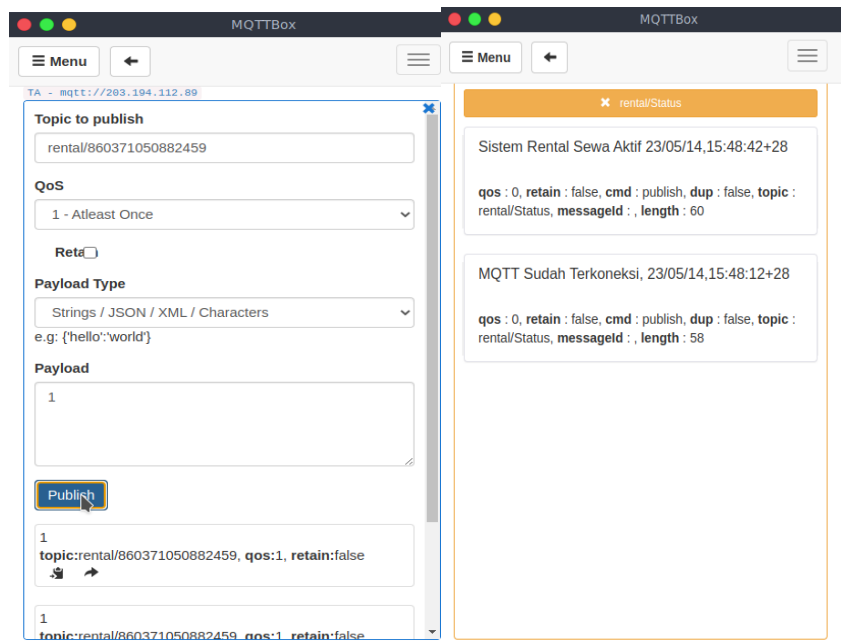
Gambar 4. 14 Pesan dari MQTT

Setelah restart MQTT di dalam server, *Service* mosquito akan kembali ke kondisi normal dan memungkinkan perangkat Bike Bike terhubung kembali ke server IoT yang ditunjukkan dalam Gambar 4.10 "Perangkat Bike Bike Terkoneksi" serta Gambar 4.11 "Pesan dari MQTT".

4.2.2 Pengujian Publisher dan subscriber

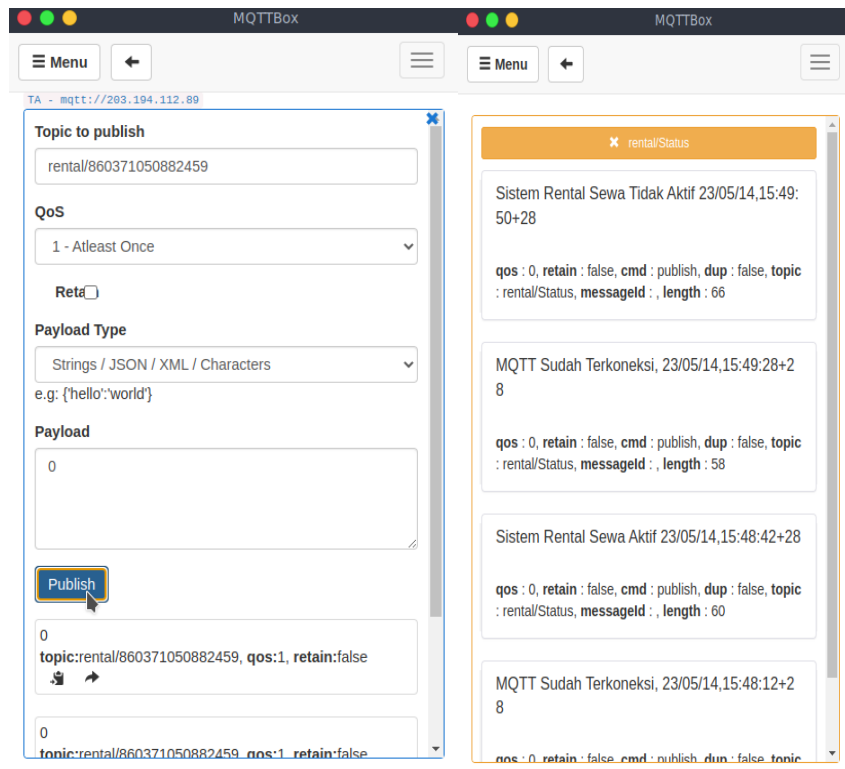
Pada pengujian publisher dan subscriber bertujuan untuk memastikan bahwa sistem MQTT berfungsi dengan baik dalam mengirim dan menerima pesan atau data. Pada pengujian publisher, dilakukan uji coba pengiriman pesan atau data dari perangkat BikeBikeAja atau aplikasi ke broker MQTT yang ada di server, sedangkan pada pengujian subscriber, dilakukan uji coba penerimaan pesan atau data dari broker MQTT ke perangkat IoT BikeBikeAja atau aplikasi yang terhubung.

Berikut ini hasil dari pengujian publisher dan subscriber.



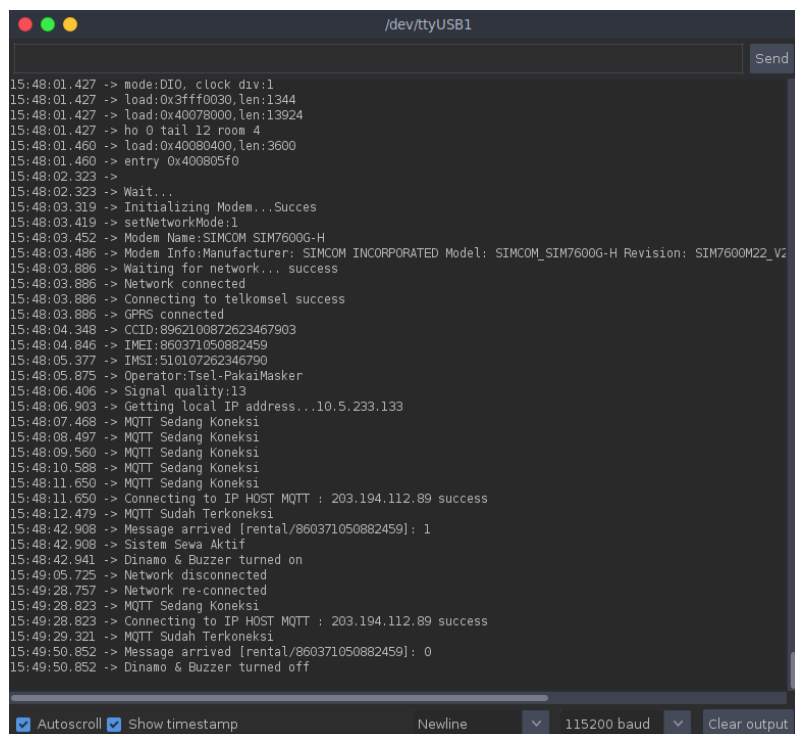
Gambar 4. 15 MQTT
Publisher rental 1

Gambar 4. 16 MQTT
Subscriber



Gambar 4. 17 MQTT
Publisher rental 0

Gambar 4. 18 MQTT
Subscriber



Gambar 4. 19 Proses Sistem Penyewaan

Dari gambar di atas, dapat diketahui bahwa pada MQTT publisher pada topik “rental/860371050882459” berhasil mengirimkan payload bernilai 1 yang berfungsi untuk mengaktifkan sistem penyewaan dengan indikator dinamo dan buzzer on. Pada subscriber, pesan tersebut berhasil terkirim pada server dan ditampilkan di melalui MQTTBox. Waktu yang diperlukan ketika mempublisch pesan berupa pengaktifan penyewaan antara publisher dan subscriber dari menit 15:48:42 sampai 15:48:42 dan waktu yang diperlukan ketika menonaktifkan penyewaaan antara menit 15:49:50 sampai 15:49:50.

Tabel 4. 2 Tabel pengiriman topik Rental Publisher dan Subscriber

Publisher	Waktu subscriber	Waktu Perangkat Berkerja
Sistem Sewa Aktif	15:48:42:28	15:48:42:908
Sistem Sewa Tidak Aktif	15:49:50:28	15:49:50:852

Sehingga Dari hasil pengujian tersebut dapat disimpulkan bahwa waktu pengiriman dari publisher ke subscriber hanya sedikit memiliki delay waktu pada saat pengiriman, yang artinya pengiriman pesan sekaligus penerima pesan sangat cepat.

4.2.3 Pengujian Keamanan Server dan MQTT

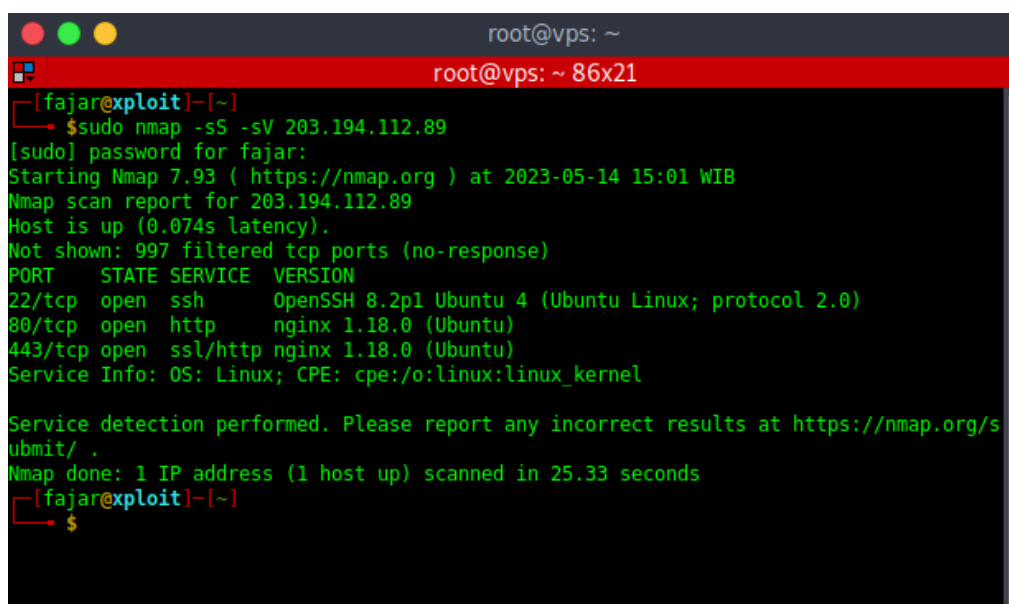
Keamanan Server VPS pada MQTT sebagai protokol IoT sangat penting untuk menjaga integritas dan kerahasiaan data yang dikirimkan antara perangkat IoT dan server. Langkah-langkah yang perlu dilakukan untuk meningkatkan keamanan meliputi penggunaan enkripsi pada komunikasi MQTT menggunakan SSL/TLS, implementasi autentikasi dan otorisasi untuk mengendalikan akses perangkat, serta penerapan firewall dan tindakan pencegahan serangan DDoS untuk melindungi server dari serangan luar. Selain itu, perlu dilakukan pemantauan aktif terhadap server VPS, pembaruan perangkat lunak secara teratur, dan pengelolaan kredensial dengan baik untuk mencegah akses yang tidak sah ke server MQTT.

4.2.3.1 Keamanan Server VPS

Pengujian keamanan server VPS melibatkan serangkaian tes, termasuk pengujian penetrasi terhadap port yang terbuka khususnya port MQTT, pemindaian pada perangkat lunak sistem operasi yang digunakan serta analisis kerentanan, untuk mengidentifikasi celah keamanan yang mungkin ada pada server VPS. Dengan melakukan pengujian keamanan yang komprehensif, dapat diidentifikasi dan

diperbaiki kelemahan-kelemahan yang ada sehingga server VPS menjadi lebih aman dan terlindungi dari ancaman yang mungkin muncul. Berikut ini hasil pengujian penetrasi testing menggunakan tool Nmap.

Nmap (Network Mapper) adalah sebuah perangkat lunak atau tools yang digunakan untuk pemindaian jaringan dan analisis keamanan. Nmap dirancang untuk melakukan pemindaian port dan penemuan host dalam suatu jaringan. Tools ini dapat digunakan untuk mengidentifikasi host yang aktif, port yang terbuka, protokol yang digunakan, dan layanan yang berjalan pada host tersebut[17].



```
root@vps: ~
root@vps: ~ 86x21
[fajar@exploit]-[~]
└─$ sudo nmap -sS -sV 203.194.112.89
[sudo] password for fajar:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-14 15:01 WIB
Nmap scan report for 203.194.112.89
Host is up (0.074s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
443/tcp   open  ssl/http nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.33 seconds
[fajar@exploit]-[~]
└─$
```

Gambar 4. 20 Hasil Scanning Server

Setelah melakukan pemindaian menggunakan Nmap dengan perintah "sudo nmap -sS -sV 203.194.112.89" untuk server VPS dengan alamat IP "203.194.112.89", hasil pemindaian dapat dilihat pada Gambar 4.17 "Hasil Pemindaian Server VPS". Gambar tersebut menampilkan informasi seperti alamat IP host, sistem operasi yang digunakan, serta daftar port dan layanan yang terbuka pada server VPS.

Berdasarkan hasil analisis, dapat disimpulkan bahwa terdapat beberapa port yang terbuka, termasuk port 22, 80, dan 443. Namun, tidak terdeteksi adanya port MQTT (port 1883) oleh Nmap. Hal ini menunjukkan bahwa sistem keamanan pada server VPS cukup baik, karena port MQTT tidak terlihat oleh alat pemindaian seperti Nmap. Dengan demikian, jika ada upaya hacker untuk melakukan serangan ilegal menggunakan port tersebut, tidak akan terdeteksi.

Gambar 4.20 memberikan gambaran tentang keamanan server VPS, dengan informasi mengenai sistem operasi, port yang terbuka, serta ketiadaan port MQTT yang terlihat.

4.2.3.2 Keamanan MQTT

Pada pengujian keamanan MQTT melibatkan pemeriksaan terhadap implementasi, autentikasi, dan otorisasi, serta kebijakan keamanan yang telah diterapkan. Hasil pengujian keamanan MQTT dapat digunakan untuk mengidentifikasi dan memperbaiki kelemahan keamanan, meningkatkan kebijakan keamanan, dan mengimplementasikan tindakan pencegahan yang tepat. Hal ini akan membantu memastikan bahwa protokol MQTT dan infrastruktur IoT terkait tetap aman dari ancaman dan serangan yang mungkin terjadi.

Berikut ini implementasi pada pengujian keamanan MQTT pada perangkat BikeBikeAja

```
// MQTT details
const char *broker = "203.194.112.89";
int port = 1883;
const char *mqtt_user = "bikebikeaja";
const char *mqtt_pass = "Bikebike4ja";
const char *topicStatus = "rental/Status";
const char *topicImei = "rental/UUID";
const char *topicRental = "rental/860371050882459";
const char *topicLocations = "rental/location";
const char *topicBattray = "rental/battray";
const char *topicWarning = "rental/warning/860371050882459";
const char *topicTime = "rental/time";

boolean mqttConnect()
{
    Serial.print("Connecting to ");
    Serial.print(broker);

    // Connect to MQTT Broker
    // boolean status = mqtt.connect("Active");

    // Or, if you want to authenticate MQTT:
    boolean status = mqtt.connect("Active", mqtt_user, mqtt_pass);

    if (status == false) {
        Serial.println(" fail");
        return false;
    }
    Serial.println(" success");
    mqtt.publish(topicStatus, imei.c_str());
    mqtt.subscribe(topicRental, 1);
    mqtt.subscribe(topicWarning, 0);
    return mqtt.connected();
}
```

Gambar 4. 21 Implementasi Autentikasi MQTT

Implementasi autentikasi pada pengujian keamanan MQTT pada perangkat BikeBikeAja dapat ditemukan dalam Gambar 4.21 "Implementasi Autentikasi

MQTT menggunakan program Arduino". Dalam program tersebut, terdapat pengaturan autentikasi dengan menggunakan `mqtt_user` dan `mqtt_pass`. Melalui autentikasi ini, akses pengguna yang tidak sah terhadap perangkat IoT BikeBikeAja dapat difilter dan dibatasi, sehingga hanya pengguna yang memiliki kredensial autentikasi yang valid yang dapat terhubung dan berinteraksi dengan perangkat tersebut.

BAB V SIMPULAN

5.1 Kesimpulan

1. Server adalah lokasi di mana sistem komputer menyediakan layanan (*service*) tertentu. Pada perangkat BikeBikeAja , server telah diterapkan sebagai layanan tertentu dengan menggunakan layanan MQTT sebagai protokol IoT dari perangkat BikeBikeAja .
2. Server menunjukkan kehandalan yang tinggi dalam memproses data baik sebagai penerima maupun pengirim dari perangkat IoT BikeBikeAja . Penggunaan *cloud computing* (VPS) dan protokol MQTT meningkatkan kecepatan secara signifikan dibandingkan dengan menggunakan protokol lain.
3. Konektivitas IoT pada perangkat BikeBikeAja menggunakan protokol MQTT terbukti sangat cepat, dengan hampir tidak ada keterlambatan yang signifikan, dalam kisaran 0 hingga 1 ms.
4. Untuk mengontrol perangkat BikeBikeAja , dapat menggunakan publisher pada MQTT yang memiliki keterlambatan waktu yang sangat rendah dalam mengaktifkan atau menonaktifkan sistem rental pada perangkat tersebut, dengan waktu delay kurang dari 1 ms. Hal ini telah terbukti melalui pengujian publisher dan subscriber.
5. Pengujian keamanan pada Server dan MQTT telah terbukti bahwa keduanya tidak memiliki kerentanan yang dapat menyebabkan serangan ilegal. Hasil pemindaian keamanan server menggunakan Nmap menunjukkan bahwa port 1883 yang digunakan oleh protokol MQTT tidak terdeteksi oleh Nmap. Pada pengujian, dalam hal keamanan protokol MQTT pada perangkat BikeBikeAja , telah diterapkan autentikasi berupa pengguna dan kata sandi untuk otentikasi pada MQTT.

5.2 Saran

Adapun beberapa saran yang disampaikan oleh penulis untuk pengembangan produk BikeBikeAja, diantaranya adalah:

1. Produk BikeBikeAja tidak hanya dirancang untuk penyewaan sepeda, tetapi juga dapat diperluas untuk diterapkan dalam penyewaan tempat dan barang lainnya, dengan mengadopsi konsep scan & use
2. Dalam pengembangan, mekanisme penguncian yang lebih beragam dapat diterapkan untuk penyewaan di bidang lain, seperti penguncian ruangan atau penguncian barang yang dapat terbuka secara otomatis saat proses penyewaan sedang berlangsung.
3. Terdapat potensi untuk melakukan lebih banyak penelitian pada bagian perangkat BikeBikeAja, terutama dalam hal pasokan daya. Di masa depan, diharapkan bahwa perangkat keras yang terintegrasi akan memiliki pasokan daya yang lebih kompak dan membutuhkan sedikit perawatan.

DAFTAR PUSTAKA

- [1] S. Sunaryo, A. Tedyyana, and K. Kasmawi, "Rancang Bangun Server Cloud Computing Di Politeknik Negeri Bengkalis," *INOVTEK Polbeng - Seri Inform.*, vol. 2, no. 1, p. 33, 2017, doi: 10.35314/isi.v2il.114.
- [2] A. Abdullah, C. Cholish, and M. Zainul haq, "Pemanfaatan IoT (Internet of Things) Dalam Monitoring Kadar Kepekatan Asap dan Kendali Pergerakan Kamera," *CIRCUIT J. Ilm. Pendidik. Tek. Elektro*, vol. 5, no. 1, p. 86, 2021, doi: 10.22373/crc.v5il.8497.
- [3] R. Dimas Prakoso and Asmunin, "Implementasi dan Perbandingan Performa Proxmox dalam Virtualisasi dengan Tiga Virtual Server (Studi Kasus : Jurusan Teknik Informatika UNESA)," *J. Manaj. Inf.*, vol. 8, no. 1, pp. 79–86, 2017, [Online]. Available: <https://jurnalmahasiswa.unesa.ac.id/index.php/11/article/view/22864>
- [4] K. Wirawan, "Wisata Sepeda Dalam Mewujudkan Pariwisata Berkelanjutan Di Sanur," *J. Master Pariwisata*, vol. 2, no. 2008, pp. 1–16, 2016, doi: 10.24843/jumpa.2016.v02.i02.p01.
- [5] M. N. Nizam, Haris Yuana, and Zunita Wulansari, "Mikrokontroler Esp 32 Sebagai Alat Monitoring Pintu Berbasis Web," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 6, no. 2, pp. 767–772, 2022, doi: 10.36040/jati.v6i2.5713.
- [6] "ESP32-WROVER-E - WI-FI/BT/BLE Module, PCB Antenna - Espressif Systems - 4MB Flash – Grid Connect," 2023. <https://www.gridconnect.com/collections/ble-modules-combo-modules/products/esp32-wrover-e-module> (accessed May 27, 2023).
- [7] SIMcom, "SIMCom Wireless Solutions - Wireless Modules and Solutions Supplier," 2022. <https://www.simcom.com/>
- [8] "MQTT Things and Channels - Bindings | openHAB." <https://www.openhab.org/addons/bindings/mqtt.generic/> (accessed May 27,

2023).

- [9] B. Sosinsky, *The book you need to succeed! Cloud Computing*. 2011.
- [10] E. Rusnandi and D. Susanti, "Perencanaan Strategis Cloud Computing Technology Berbasis Gafe (Google Apps For Education) Bagi Perguruan Tinggi Swasta Di Wilayah Iii Cirebon Propinsi Jawa Barat," *J. Comput. Bisnis*, vol. 6, no. 1, pp. 1–16, 2012, [Online]. Available: <http://jurnal.stmik-mi.ac.id/index.php/jcb/article/view/93>
- [11] "Cloud Computing Is Important, HD Png Download - kindpng." https://www.kindpng.com/imgv/Tibobib_cloud-computing-is-important-hd-png-download/ (accessed May 27, 2023).
- [12] F. Susanto, N. Komang Prasiani, and P. Darmawan, "Implementasi Internet of Things Dalam Kehidupan Sehari-Hari," *J. IMAGINE*, vol. 2, no. 1, pp. 2776–9836, 2022, [Online]. Available: <https://jurnal.std-bali.ac.id/index.php/imagine>
- [13] A. Junaidi, "Internet Of Things, Sejarah, Teknologi Dan Penerapannya : Review," *J. Ilm. Teknol. Inf.*, vol. IV, no. 3, pp. 62–66, 2015.
- [14] D. Pratama and N. Sariana, "Rancang Bangun Sistem Informasi Penyewaan Kendaraan Berbasis Web," *J. Sist. Inf. dan Sains Teknol.*, vol. 1, no. 1, pp. 1–10, 2019, doi: 10.31326/sistek.v1i1.321.
- [15] D. Nurhannavi, F. Yumono, and P. N. Rahayu, "RANCANG BANGUN ALAT KEAMANAN SEPEDA MOTOR BERBASIS IoT MENGGUNAKAN NODEMCU DAN GPS," *JTECS J. Sist. Telekomun. Elektron. Sist. Kontrol Power Sist. Komput.*, vol. 1, no. 1, pp. 23–32, 2021.
- [16] Satria Turangga, Martanto, and Yudhistira Arie Wijaya, "Analisis Internet Menggunakan Paramater Quality of Service Pada Alfamart Tuparev 70," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 6, no. 1, pp. 392–398, 2022, doi: 10.36040/jati.v6i1.4693.
- [17] M. R. Fahlevi and D. R. D. Putri, "Analisis Monitoring & Kinerja Sistem Keamanan Jaringan Komputer Menggunakan Nmap (Studi Kasus: Raz Hotel

& Convention Medan),” *It (Informatic Tech. J.*, vol. 9, no. 1, p. 35, 2021, doi: 10.22303/it.9.1.2021.35-43.

Lampiran

```
void mqttCallback(char *topic, byte *payload, unsigned int len)
{
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("]: ");
    Serial.write(payload, len);
    Serial.println();
    // memcpy(receivedPayload, payload, len);
    // receivedPayload[len] = '\0';

    // // Only proceed if incoming message's topic matches

    if (String(topic) == topicRental) {
        if ((char)payload[0] == '1') {
            Serial.println("Sistem Sewa Aktif");
            mqtt.publish(topicStatus, "Sistem Sewa Aktif");
            digitalWrite(Dinamo, HIGH);
            digitalWrite(Buzzer, HIGH);
            delay(50);
            digitalWrite(Buzzer, LOW);
            delay(50);
            digitalWrite(Buzzer, HIGH);
            delay(50);
            digitalWrite(Buzzer, LOW);
            delay(50);
            delay(903);
            digitalWrite(Dinamo, LOW);
            digitalWrite(Buzzer, LOW);
        } else if ((char)payload[0] == '0') {
            digitalWrite(Dinamo, LOW);
            Serial.println("Dinamo turned off");
            mqtt.publish(topicStatus, "Waktu Habis");
        }
    }
}

boolean mqttConnect()
{
    Serial.print("Connecting to ");
    Serial.print(broker);

    // Connect to MQTT Broker
    // boolean status = mqtt.connect("Active");

    // Or, if you want to authenticate MQTT:
    boolean status = mqtt.connect("Active", mqtt_user, mqtt_pass);

    if (status == false) {
        Serial.println(" fail");
        return false;
    }
    Serial.println(" success");
    mqtt.publish(topicStatus, imei.c_str());
    mqtt.subscribe(topicRental, 1 );
    mqtt.subscribe(topicWarning, 0);
    return mqtt.connected();
}
```

```

if (!mqtt.connected()) {
    Serial.println("=== MQTT NOT CONNECTED ===");
    // Reconnect every 10 seconds
    uint32_t t = millis();
    if (t - lastReconnectAttempt > 10000L) {
        lastReconnectAttempt = t;
        if (mqttConnect()) {
            lastReconnectAttempt = 0;
        }
    }
    delay(100);
    return;
}

```

The image shows two screenshots of the Arduino IDE interface. The top screenshot displays a sketch named 'TTGO_T-SIM_7600G-H' with the following code:

```

86 Buzzer_Blink() {
87 }
88 }
89
90 else if (String(topic) == topicWarning) {
91     if ((char)payload[0] == '1') {
92         Serial.println("Sistem Sewa Aktif");
93         mqtt.publish(topicStatus, "Diluar Area");
94         digitalWrite(Buzzer, HIGH);
95         delay(100);
96         digitalWrite(Buzzer, LOW);
97         delay(100);
98         digitalWrite(Buzzer, HIGH);
99         delay(100);
100        digitalWrite(Buzzer, LOW);
101        delay(200);
102    }
103    else if ((char)payload[0] == '0') {
104        digitalWrite(Buzzer, LOW);
105        Serial.println("Dinamo turned off");
106        mqtt.publish(topicStatus, "Didalam Area");
107    }
108 }
109 }
110 }
111
112 boolean mqttConnect()
113 {
114     Serial.print("Connecting to ");
115     Serial.print(broker);
116 }
117

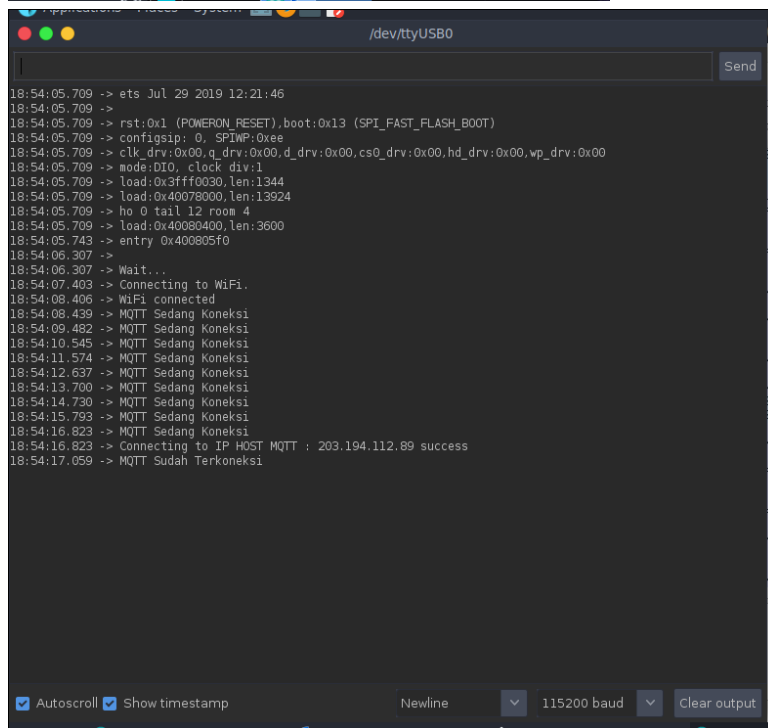
```

The bottom screenshot displays another sketch named 'TTGO_T-SIM_7600G-H' with the following code:

```

54 void mqttCallback(char *topic, byte *payload, unsigned int len)
55 {
56     Serial.print("Message arrived [");
57     Serial.print(topic);
58     Serial.print("]: ");
59     Serial.write(payload, len);
60     Serial.println();
61     memcpy(receivedPayload, payload, len);
62     receivedPayload[len] = '\0';
63
64     // Only proceed if incoming message's topic matches
65
66     if (String(topic) == topicRental) {
67         if ((char)payload[0] == '1') {
68             Serial.println("Sistem Sewa Aktif");
69             mqtt.publish(topicStatus, "Sistem Sewa Aktif");
70             digitalWrite(Dinamo, HIGH);
71             digitalWrite(Buzzer, HIGH);
72             delay(50);
73             digitalWrite(Buzzer, LOW);
74             delay(50);
75             digitalWrite(Buzzer, HIGH);
76             delay(50);
77             digitalWrite(Buzzer, LOW);
78             delay(50);
79             delay(500);
80             digitalWrite(Dinamo, LOW);
81             digitalWrite(Buzzer, LOW);
82         } else if ((char)payload[0] == '0') {
83             digitalWrite(Dinamo, LOW);
84             Serial.println("Dinamo turned off");
85             mqtt.publish(topicStatus, "Waktu Habis");

```



MQTT

No.	Time	Source	Destination	Protocol	Length	Info
45	6.288894832	192.168.43.151	203.194.112.89	MQTT	68	Ping Request
46	6.326273827	203.194.112.89	192.168.43.151	MQTT	68	Ping Response
50	6.694611860	192.168.43.151	91.121.93.94	MQTT	68	Ping Request
51	6.813772145	91.121.93.94	192.168.43.151	MQTT	68	Ping Response
63	11.755402913	203.194.112.89	192.168.43.151	MQTT	126	Publish Message [rental/Status]
80	16.296598728	192.168.43.151	203.194.112.89	MQTT	68	Ping Request
82	16.343215525	203.194.112.89	192.168.43.151	MQTT	68	Ping Response
84	16.612986597	192.168.43.151	91.121.93.94	MQTT	68	Ping Request
85	16.829786518	91.121.93.94	192.168.43.151	MQTT	68	Ping Response

MQ Telemetry Transport Protocol: Protocol

Packets: 91 - Displayed: 9 (9.9%) - Dropped: 0 (0.0%)

Profile: Default

```

/usr/bin/bash
/usr/bin/bash 80x24
[fajaro@exploit]~$
$ssh -v root@203.194.112.89

```



```
root@vps: ~
root@vps: ~ 88x26
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat 13 May 2023 07:00:26 PM CEST

System load:  0.03           Users logged in:      0
Usage of /:   27.1% of 19.38GB IPv4 address for eth0: 203.194.112.89
Memory usage: 34%           IPv6 address for eth0: 2001:df0:27b::276
Swap usage:   1%             IPv4 address for eth1: 10.0.162.192
Processes:    115

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

300 updates can be installed immediately.
201 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Fri May 12 13:52:52 2023 from 116.206.39.118
root@vps:~#
```

```
root@vps: ~
root@vps: ~ 162x20
# mosquitto.service - Insite MQTT Broker
Loaded: loaded (/etc/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
Active: active (running) since Sat 2023-05-13 19:02:42 CEST; 2s ago
Main PID: 53103 (mosquitto)
Tasks: 1 (limit: 1074)
Memory: 1.9M
CGroup: /system.slice/mosquitto.service
└─53103 /usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47882 as mqttjs_1679dc4 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47884 as mqttjs_8cc27fa4 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47886 as mqttjs_85a38cf5 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997363: New client connected from 203.194.112.89:47888 as mqttjs_9de352eb (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47890 as mqttjs_55ebef71 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47892 as mqttjs_01553396 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47894 as mqttjs_56a9b02e (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47896 as mqttjs_46651d8b (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47898 as mqttjs_03a2da02 (p2, c1, k60, u'bikebikeaja').
May 13 19:02:44 vps.bikebikeaja.com mosquitto[53103]: 1683997364: New client connected from 203.194.112.89:47900 as mqttjs_96a21c61 (p2, c1, k60, u'bikebikeaja').
lines 1-19/19 (END)
```