Analisis Performansi Protokol MQTT Pada Sistem Pemantauan Kualitas Udara Ruangan Berbasis IoT

Raden Mumtaz Rafly Akbar[1], Teuku Yuliar Arif[2], Muhammad Irhamsyah[3]

Jurusan Teknik Elektro dan Komputer, Universitas Syiah Kuala
Jln. Teuku Nyak Arief Darussalam, Banda Aceh, Aceh, 23111, Indonesia

[1]rafly2018@mhs.unsyiah.ac.id
[2]yuliar@unsyiah.ac.id
[3]irham.ee@unsyiah.ac.id

Abstrak— **Dalam konteks pandemi, kesehatan menjadi krusial, dan perhatian terhadap kualitas udara dalam ruangan pun semakin penting. Kajian ini berfokus pada evaluasi performansi protokol komunikasi untuk sistem pemantauan kualitas udara berbasis *Internet of Things* (IoT), yang dapat diakses melalui perangkat *mobile*. Penelitian ini menggunakan protokol *Message Queuing Telemetry Transport* (MQTT), terkenal sebagai protokol ringan dan hemat energi dengan model *messaging publish-subscribe*. MQTT dipilih karena fleksibilitasnya yang memungkinkan implementasi pada berbagai situasi jaringan terbatas. Dalam analisis performansinya, dua parameter utama digunakan: konsumsi energi dan Quality of *Service* (QoS) jaringan, *termasuk throughput, delay, jitter,* dan *packet loss*. Hasil menunjukkan bahwa konsumsi energi pada tiga tingkatan QoS (0, 1, dan 2) selama 30 menit cukup serupa, sementara QoS jaringan, dengan QoS 0 dan QoS 1 sangat memuaskan serta QoS 2 memuaskan, mengindikasikan kehandalan dan responsivitas dalam pengiriman pesan.**

Kata kunci— ***indoor air quality*, MQTT, IoT, *energy consumption*, QoS.**

I. PENDAHULUAN

Polusi udara, baik di dalam maupun di luar ruangan, mengancam kesehatan manusia dengan menyebabkan lebih dari 6,5 juta kematian dini setiap tahunnya [1]. WHO mengidentifikasi polusi udara sebagai masalah lingkungan utama yang dapat menyebabkan 7 juta kematian per tahun, melebihi angka kematian dari penyakit seperti malaria dan AIDS. Dampaknya meluas pada kesehatan jantung dan risiko stroke [2]. Kualitas udara yang buruk mempengaruhi kesejahteraan manusia dengan menurunkan kondisi fisik, meningkatkan risiko penyakit, dan mengganggu kinerja kerja [3].

Dalam era IoT, pemilihan komponen seperti sensor, mikrokontroler, dan protokol komunikasi sangat penting karena berpengaruh kepada system yang akan dibangun. Penelitian ini mengembangkan sistem pemantauan kualitas udara IoT menggunakan ESP8266 dan sensor seperti DHT-11, GY-302 BH1750, dan DSM501A. Evaluasi protokol komunikasi MQTT pada tiga tingkatan QoS berbeda dilakukan, dengan fokus pada kualitas layanan jaringan dan konsumsi energi. Hasilnya memberikan panduan untuk pengembangan pemantauan kualitas udara yang efisien dan andal.

II. DASAR TEORI

A. Kualitas Udara

Udara, penting bagi hidup manusia, mengandung oksigen esensial dan juga zat-zat seperti virus dan karbon monoksida. Konsentrasi zat ini harus terjaga dalam batas normal untuk penetralisiran yang efektif [4]. Kualitas udara dibagi menjadi dalam ruangan dan luar ruangan. Kualitas udara dalam ruangan sangat penting karena banyak aktivitas manusia terjadi di dalam ruangan. Parameter fisik kualitas udara ruangan diatur dalam Keputusan Menteri Kesehatan Republik Indonesia nomor 1077/MENKES/PER/V/2011, termasuk suhu, pencahayaan, kelembaban, laju ventilasi, dan *Particulate Matter* (PM2,5 dan PM10). Namun, penelitian ini fokus pada 4 parameter utama, sesuai dengan efisiensi alat seperti yang tercantum dalam Tabel I.

TABEL I
PARAMETER FISIK KUALITAS UDARA RUANGAN [5]

No	Parameter	Satuan	Kadar yang dipersyaratkan
1	Suhu	ºC	18-30
2	Pencahayaan	Lux	Minimal 60
3	Kelembaban	% Rh	40-60
4	$PM_{2,5}$	µg/ m^3	0-15 [6]

B. Internet of Things (IoT)

Internet of Things (IoT) memanfaatkan konektivitas internet untuk menghubungkan perangkat dengan sensor dan aktuator. Perangkat ini mampu memperoleh serta mengolah data secara independent, menghasilkan respons atau tindakan secara *real-time*. Konsep dasar IoT melibatkan perangkat fisik dengan modul sensor, konektivitas internet, dan sentral data di server. Dalam penerapannya, perangkat IoT yang terkoneksi ke internet mengumpulkan data menjadi "*big data*" yang dapat diproses oleh berbagai pihak, seperti perusahaan atau instansi pemerintah, sesuai dengan kebutuhan mereka [7].

C. Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) adalah protokol komunikasi terbuka dengan model publish-subscribe, didesain khusus untuk perangkat terbatas dalam aplikasi

telemetri. MQTT dirancang agar sederhana bagi pengguna baik di pihak *subscriber* maupun *publisher*. Kompleksitas sistem diatur oleh broker, yang bertanggung jawab atas berbagai tindakan dalam fungsionalitas MQTT. Broker berperan sebagai perantara antara *publisher* dan *subscriber*, mengatur aliran pesan, topik penyimpanan, serta memastikan pengiriman pesan kepada penerima [8]. Dalam menjamin pesan MQTT memiliki 3 layanan penjamin yaitu QoS 0, Qos 1, dan QoS 2.

1) Quality of Services 0 (QoS 0) MQTT: QoS 0 adalah tingkatan terendah dari layanan MQTT dan biasa dikenal dengan "*fire and forget*". Pada tingkatan ini, pengirim atau *publisher* tidak menunggu respon "*acknowledgement*" atau menyimpan dan mengirimkan kembali pesan, sehingga kemungkinan pesan duplikat diterima tidak akan terjadi namun kemungkinan pesan tidak sampai ke penerima bisa saja terjadi [9].

2) Quality of Services 1 (QoS 1) MQTT: Dalam menjamin pengiriman pesan, QoS 1 melibatkan fungsi respon "*acknowledgement*" serta mekanisme pengiriman ulang pesan. Ketika pengirim atau *publisher* menerima paket "*PUBACK*" dari penerima, hal ini menandakan bahwa pesan telah berhasil diterima oleh broker. Sampai saat itu, pengirim harus menyimpan paket "*PUBLISH*" untuk tujuan pengiriman ulang. Pengirim menggunakan paket ID (*Identification*) dalam setiap paket untuk mencocokkan paket "*PUBLISH*" dengan paket "*PUBACK*" yang sesuai. Hal ini memungkinkan pengirim untuk mengidentifikasi dan menghapus paket "*PUBLISH*" yang benar dari *cache*-nya [9].

3) Quality of Services 2 (QoS 2) MQTT: *Dalam QoS 2* pengirim tidak diizinkan mengirim kembali paket "*PUBLISH*" sebelum menerima paket "*PUBREC*" dari penerima. Setelah pengirim menerima paket "*PUBREC*" dan mengirim paket "*PUBREL*", proses pelepasan paket ID dimulai. Selama proses ini berlangsung, pengirim tidak dapat mengirim ulang paket "*PUBLISH*" atau mengirim pesan baru dengan paket ID yang sama sampai ia menerima paket "*PUBCOMP*" dari penerima hal ini yang menyebabkan QoS 2 tidak akan kehilangan ataupun menduplikasi pesan yang sama [9].

D. Performansi MQTT

MQTT menyatakan dirinya sebagai protokol komunikasi publish/ subscribe antar mesin dan IoT yang sangat ringan [10]. Dalam penelitian ini dilakukan pengujian untuk membuktikan pernyataan tersebut, dalam melakukan pengujian dibutuhkan parameter yang relevan untuk membuktikannya, untuk itu penulis memilih parameter konsumsi energi dan QoS jaringan yang tersusun dari throughput, delay, jitter, dan packet loss. Sehingga dari penelitian ini dapat dilihat performa dari protokol komunikasi MQTT yang disebut sangat ringan.

1) Konsumsi Energi: *Efisiensi energi* adalah faktor krusial dalam desain dan implementasi solusi IoT berkelanjutan, yang mencakup perangkat keras, protokol komunikasi, dan manajemen sumber daya. Pemahaman tentang konsumsi dan penghitungan energi dalam konteks IoT sangat penting untuk pengembangan solusi yang hemat energi dan tahan lama. Dalam mengukur konsumsi energi perangkat IoT, parameter seperti daya listrik, energi terintegrasi dari waktu, efisiensi energi, *duty cycle* (persentase waktu perangkat aktif), dan tingkat transmisi penting untuk dihitung. Pertimbangan efisiensi juga berperan dalam pemilihan protokol komunikasi. Protokol seperti MQTT dan CoAP yang efisien energi dapat meminimalkan penggunaan energi dalam komunikasi IoT dengan mengurangi *overhead* komunikasi dan waktu operasi dalam mode aktif.

2) Quality of Services (QoS) Jaringan: *Parameter Quality of Service (QoS)* jaringan merupakan parameter yang digunakan untuk mengukur baik/ buruknya jaringan dan merupakan salah satu upaya untuk menggambarkan karakteristik dan sifat dari suatu layanan [11]. Parameter QoS jaringan tersusun dari throughput, jitter, packet loss dan delay (latency) dengan standar persentase dan nilai QoS jaringan seperti pada Tabel II.

TABEL II
STANDAR PERSENTASE DAN NILAI QOS JARINGAN [12]

Nilai	Persentase (%)	Indeks
3,8 – 4	95 – 100	Sangat Mumuaskan
3 – 3,79	75 – 94,75	Memuaskan
2 – 2,99	50 – 74,75	Kurang Memuaskan
1 – 1,99	25 – 49,75	Buruk

E. NodeMCU ESP8266

ESP8266 adalah papan pengembangan ekonomis yang menyatukan beragam fitur termasuk GPIO, UART, I2C, PWM, dan ADC bersama dengan konektivitas WiFi untuk *prototyping* cepat. Papan ini beroperasi pada tegangan 3,3V dan disajikan dalam modul ESP-12 dengan pengatur tegangan serta *port serial* USB. ESP8266 dapat dikembangkan menggunakan Arduino IDE atau ESPlorer berbasis bahasa pemrograman Lua. Fitur-fitur canggih yang dimilikinya seperti yang disebutkan di atas menjadikannya cocok untuk aplikasi IoT, memungkinkan implementasi yang efektif dan efisien [13].

F. Sensor

Dalam sistem elektronik, sensor berfungsi mengubah besaran fisik menjadi sinyal listrik yang dapat diolah oleh sistem digital atau rangkaian listrik. Jenis sensor dibagi menjadi sensor fisika dan kimia, tergantung pada variabel yang diindra. Sensor fisika mendeteksi berdasarkan hukum fisika, misalnya cahaya, suara, dan suhu. Sensor kimia mendeteksi jumlah zat kimia seperti pH, gas, dan oksigen [14]. Dalam penelitian ini, digunakan sensor cahaya (GY-302 BH1750), Sensor suhu dan kelembaban (DHT-11), dan Sensor $PM_{2,5}$ (DSM501A).

G. MQTT Broker

MQTT broker berperan sebagai penghubung dan pengatur transmisi data antara *publisher* dan *subscriber* melalui

pengelompokan yang disebut "topik". Dalam skenario di mana *publisher* mengirim data sensor A, B, dan C dengan informasi "data1", jika *subscriber* berlangganan dengan informasi yang sama, maka *subscriber* akan menerima data dari semua sensor tersebut. Penelitian ini menggunakan dua broker MQTT, yaitu EMQX untuk broker lokal dan hiveMQ untuk broker publik. Kedua broker memiliki batas maksimum *payload* sebesar 256 MB, dengan EMQX memiliki pengaturan *default* 1 MB dan hiveMQ 256 MB [15] [16].

H. MIT App Inventor

MIT App Inventor merupakan lingkungan pengembangan terintegrasi yang memungkinkan *programmer* pemula untuk memulai membuat aplikasi untuk sistem operasi android maupun IOS [17]. Aplikasi web ini gratis, layanan berbasis *cloud* yang memungkinkan untuk membuat aplikasi sendiri menggunakan bahasa pemrograman berbasis blok.

I. RD UM24C

RD UM24C merupakan perangkat *bluetooth* USB *tester* yang dapat melakukan pengukuran tegangan (V), arus (A), kapasitas (mAh) dan energi (mWh). Perangkat ini memiliki layar LCD (*Liquid Crystal Display*) berwarna dengan ukuran 1.44 inci dengan akurasi tegangan ±(0,2% + 1 digit) dan akurasi arus ±(0,8% + 3 digit) yang memiliki tampilan seperti pada Gambar 9. Selain itu perangkat ini dapat merekam hasil bacaan nya dan dapat terhubung ke *android phone* maupun *personal computer* [18].

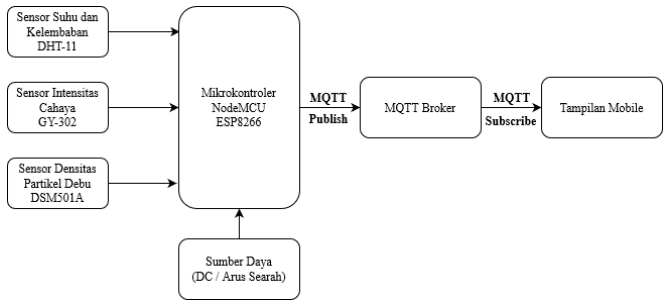
J. Wireshark

Wireshark merupakan alat yang berfungsi untuk menganalisis protokol komunikasi. Alat ini memiliki banyak fitur dan berjalan di hampir seluruh *platform* komputasi termasuk Windows, Linux, OS X, dan UNIX. *Wireshark* tersedia secara gratis sebagai ”*open source*” dan dirilis di bawah GNU *General Public License version 2* [19]. *Wireshark* memiliki kemampuan untuk menganalisis struktur lalu lintas jaringan dan dapat digunakan untuk mencari kemungkinan kesalahan konfigurasi dan serangan keselamatan [20].

III. METODE PENELITIAN

A. Perancangan Alur Kerja Alat

Dalam perancangan alur kerja alat, langkah awal adalah membuat diagram blok yang menggambarkan hubungan antara komponen-komponen yang saling terkait. Setiap komponen blok mempengaruhi komponen lainnya dalam diagram ini. Diagram blok ini memiliki arti khusus yang memberikan penjelasan tentang hubungan dan arah kerja komponen. Diagram blok ini digunakan dalam perancangan alur kerja alat pada penelitian ini, seperti yang terlihat pada Gambar 1.



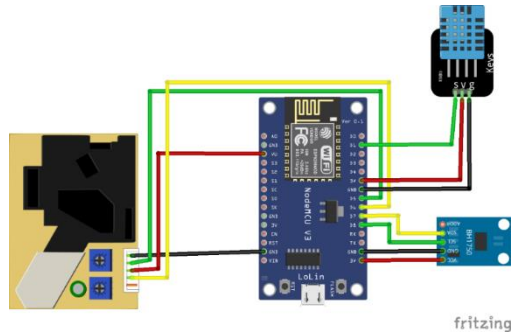
Gambar 1. Diagram Perancangan Alur Kerja Alat

Diagram blok tersebut mencakup 3 sensor utama: sensor suhu dan kelembaban, sensor cahaya, dan sensor densitas partikel debu. Ketiga sensor ini terhubung ke NodeMCU ESP8266, yang bertindak sebagai mikrokontroler untuk memproses data yang dikumpulkan oleh sensor-sensor tersebut. Data yang diolah kemudian dikirim ke broker melalui protokol komunikasi MQTT. Data dari broker selanjutnya disalurkan ke pengguna melalui tampilan mobile, yang juga menggunakan protokol komunikasi MQTT. Sumber daya untuk sistem ini disediakan oleh sumber daya DC dari adaptor atau power bank.

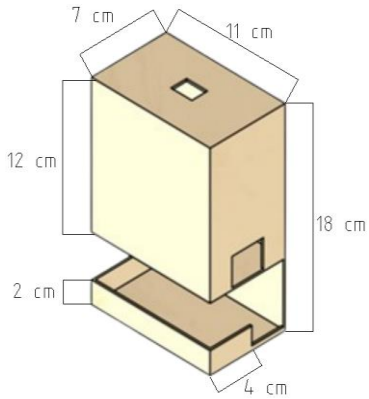
B. Pembuatan Alat

Pembuatan alat terdiri dari 2 tahapan yang saling terintegrasi yang terdiri dari perakitan perangkat keras dan pengembangan perangkat lunak untuk memastikan sistem yang dibangun dapat berjalan dengan baik.

1) Perakitan Perangkat Keras: Penentuan desain didasari pada kebutuhan system yang akan dibangun sehingga didapat hasil desain perangkat keras seperti pada Gambar 2. Setelah *layout* komponen telah ditentukan selanjutnya dilakukan desain *housing* dari komponen tersebut seperti yang dapat dilihat di Gambar 3. *Housing* berdimensi 11 x 7 x 18 cm, terbuat dari bahan akrilik, dengan bagian atas tertutup yang berfungsi menempatkan mikrokontroler dan sensor, sementara bagian bawah yang terbuka digunakan sebagai tempat penyimpanan *power bank* sebagai sumber daya perangkat.

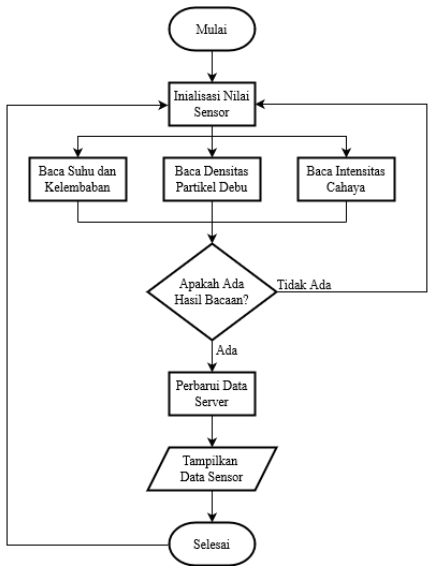


Gambar 2. *Layout* Komponen Sistem Pemantauan Kualitas Udara



Gambar 3. Desain *Housing* Sistem Pemantauan Kualitas Udara Ruangan

2) Pengembangan Perangkat Lunak: Proses ini mencakup pembuatan dan pengembangan program serta aplikasi untuk pemantauan kualitas udara. Program dibuat menggunakan aplikasi VSCode untuk diunggah ke NodeMCU ESP8266, sehingga perangkat dapat mengambil nilai/variabel dari sensor dan mengirimnya ke broker melalui protokol komunikasi MQTT. Data kemudian diteruskan ke pengguna, yang dapat mengaksesnya melalui perangkat *mobile*. Aplikasi *mobile* ini dirancang menggunakan *MIT App Inventor*. Prinsip kerja dari proses ini diilustrasikan dalam Gambar 4.



Gambar 4. Diagram Alir Program Pemantauan Kualitas Udara

C. Pengambilan dan Analisis Data Konsumsi Energi dan QoS Jaringan

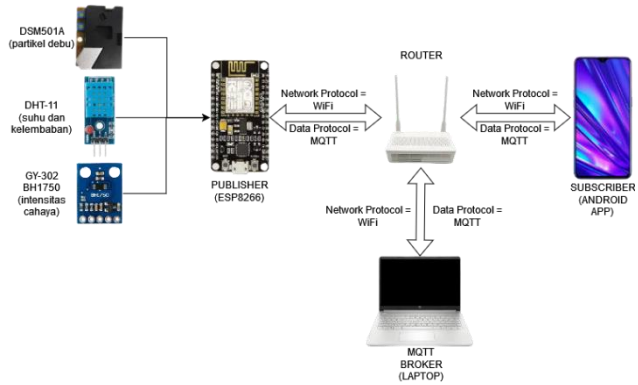
Pada tahap ini sumber daya sebelum dihubungkan dengan NodeMCU ESP8266 lebih dulu dihubungkan secara seri dengan perangkat RD UM24C yang berfungsi mengambil data konsumsi energi dari NodeMCU ESP8266. Sedangkan untuk mengambil data QoS jaringan digunakan aplikasi *Wireshark* yang terhubung dengan jaringan yang sama dengan perangkat IoT sehingga pembacaan parameter QoS jaringan dapat dilakukan. Pemantauan konsumsi energi dan QoS jaringan dilakukan selama 30 menit untuk setiap tingkat QoS MQTT, selanjutnya dilakukan komparasi dari hasil yang didapat agar diketahui performansi antar tingkat QoS MQTT.

Proses pengolahan data melibatkan perbandingan antara hasil pengukuran dengan standar TIPHON (Telecommunications and Internet Protocol Harmonization Over Network). TIPHON merupakan sebuah kerangka standar yang digunakan untuk menilai parameter QoS (Quality of Service) yang telah didefinisikan oleh ETSI (European Telecommunications Standards Institute). Setelah itu, dilakukan analisis terhadap kriteria-kriteria jaringan yang terdapat dalam standar TIPHON dan dari hasil parameter tersebut diambil kesimpulan yang relevan [12].

IV. Hasil Dan Pembahasan

A. Hasil Desain Jaringan Sistem Pemantauan Kualitas Udara Ruangan Berbasis IoT

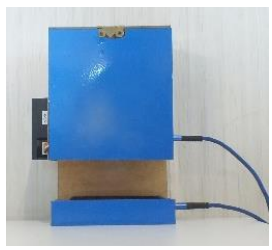
Sistem pemantauan kualitas udara ruangan berbasis IoT dengan protokol komunikasi MQTT pada penelitian ini berjalan pada jaringan lokal seperti pada Gambar 5. Meskipun Sistem ini dapat berjalan di jaringan publik tapi keterbatasan alat ukur berupa *Wireshark* yang hanya mampu memantau jaringan di tempat diinstalnya aplikasi tersebut membuat penelitian ini hanya dapat dilakukan pada jaringan lokal.



Gambar 5. Topologi Lokal Sistem Pemantauan Kualitas Udara Ruangan

B. Prototipe Alat Pemantauan Kualitas Udara Ruangan

Prototipe alat pemantauan kualitas udara ruangan didasarkan pada hasil desain alat pada poin pembuatan alat pada bagian metode penelitian, dari hasil desain tersebut dilakukan *housing* alat dan dilanjutkan penempatan sensor dan mikrokontroler sesuai desain awal juga. Dari proses diatas didapat hasil prototipe alat seperti pada Gambar 6.



Gambar 6. Alat Pemantauan Kualitas Udara Ruangan

e-ISSN: 2252-7036
Vol.8 No.3 2023: 102-109

Penempatan sensor disesuaikan untuk memastikan pengukuran kualitas udara dilakukan secara optimal. Sensor Cahaya GY-302 BH1750 diletakkan di atas untuk mendapatkan cahaya langsung dari lampu langit-langit ruangan, sementara Sensor DHT-11 untuk suhu dan kelembaban serta Sensor partikulat debu DSM501A untuk mengukur $PM_{2,5}$ disusun secara vertikal di sebelah kiri untuk efisiensi pengukuran. Terdapat lubang daya di sisi kanan dan tempat penyimpanan daya, seperti *powerbank* atau baterai, ditempatkan di bagian bawah alat.

C. *Prototipe Aplikasi Pemantauan Kualitas Udara Ruangan*

Dalam merancang aplikasi, perlu mempertimbangkan faktor-faktor berikut: antarmuka yang intuitif dan dipahami, fungsionalitas dengan fitur yang relevan, optimalisasi untuk perangkat *mobile*, serta aksesibilitas bagi pengguna dengan kebutuhan khusus, termasuk penyesuaian ukuran teks, kontras warna yang baik, dan navigasi yang *user-friendly*. Dengan memperhatikan aspek-aspek tersebut, dihasilkan prototipe aplikasi seperti yang terlihat pada Gambar 7.

Gambar 7. Aplikasi Pemantauan Kualitas Udara Ruangan

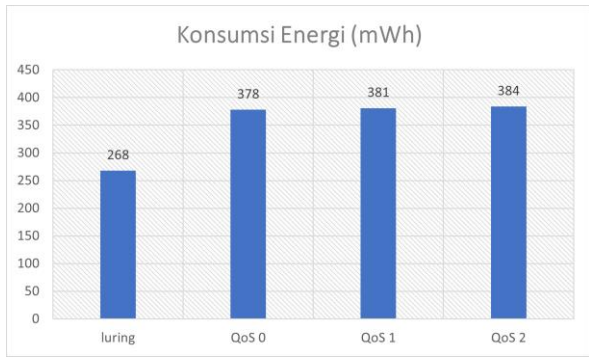
Pada Gambar 7, terlihat prototipe aplikasi sistem pemantauan kualitas udara terdiri dari tiga bagian utama. Bagian atas menampilkan judul dan status aplikasi. Bagian tengah terdiri dari empat kolom yang menampilkan empat parameter kualitas udara, yakni suhu, kelembaban, intensitas cahaya, dan $PM_{2,5}$. Bagian bawah berfungsi untuk memasukkan alamat broker agar aplikasi terhubung dengan alat pemantauan kualitas udara ruangan. Pada bagian tengah, setiap warna dalam kolom mencerminkan status parameter. Warna hijau mengindikasikan nilai parameter dalam kisaran ideal sesuai Tabel I. Warna merah menunjukkan nilai parameter di luar kisaran ideal atau Tabel I. Sedangkan warna abu-abu menandakan "*no data*" atau sensor gagal mendeteksi parameter tersebut.

D. *Hasil Pengujian Fungsionalitas*

Hasil pengujian fungsionalitas sistem pemantauan kualitas udara ruangan IoT menunjukkan dampak positif dari aspek yang diujikan seperti perangkat keras mampu membaca dan mengirimkan data sensor, serta konektivitas antara perangkat dan aplikasi tetap stabil dengan latensi minimal. Meskipun terjadi penumpukan data di aplikasi *mobile*, ini lebih terkait dengan kinerja perangkat seluler daripada sistem pemantauan itu sendiri. Dari segi antarmuka pengguna dan perangkat lunak, aplikasi beroperasi dengan baik, menampilkan hasil sensor secara akurat, dan antarmuka yang intuitif mendukung interaksi yang lancar. Kesimpulannya, pengujian fungsionalitas menunjukkan dampak positif, dengan perangkat keras berfungsi baik, konektivitas stabil, dan antarmuka pengguna mudah digunakan.

E. *Hasil Pengujian Konsumsi Energi*

Pengujian konsumsi energi bertujuan untuk mengevaluasi efisiensi perangkat atau sistem pemantauan kualitas udara ruangan berbasis IoT, serta mengidentifikasi potensi penghematan energi. Pengujian ini mencakup empat skenario yang berbeda: QoS 0, QoS 1, QoS 2, dan kondisi luring (tanpa menggunakan MQTT). Dalam penelitian ini, sumber daya DC yang digunakan adalah *powerbank* dengan kapasitas 10.000 mAh untuk pengujian pada QoS 0, QoS 1, dan QoS 2. Pengujian dalam kondisi luring menggunakan sumber daya DC dari laptop agar proses pemantauan tetap aktif. Setiap skenario diuji selama 30 menit untuk memperoleh hasil perhitungan yang tercantum pada Gambar 8.



Gambar 8. Grafik Nilai Konsumsi Energi

Grafik pada Gambar 8 menunjukkan konsumsi energi dalam mWh pada empat skenario: luring, QoS 0, QoS 1, dan QoS 2. Skenario luring memiliki konsumsi energi dasar 268 mWh. QoS 0 meningkatkan konsumsi menjadi 378 mWh karena *overhead* jaringan dan komunikasi perangkat. Skenario QoS 1 dengan tujuan pengiriman minimal satu pesan memiliki konsumsi 381 mWh. QoS 2, yang menjamin pengiriman satu pesan dengan keandalan tinggi, mencapai puncak 284 mWh karena kompleksitas protokol. Perbandingan ini menggambarkan variasi konsumsi energi terkait tingkat kualitas layanan, penting untuk optimasi energi dalam sistem IoT dan pemilihan QoS MQTT yang tepat. Kesimpulannya, grafik ini memberikan wawasan vital tentang

kebutuhan energi pada setiap tingkat QoS MQTT dalam sistem pemantauan kualitas udara berbasis IoT.

F. Hasil Pengujian QoS Jaringan

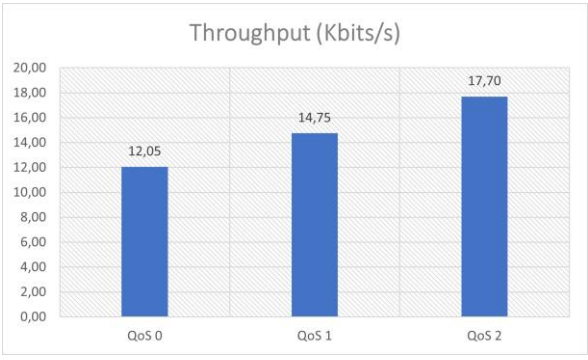
Bagian ini menyajikan hasil pengujian QoS jaringan dalam tiga skenario berbeda: QoS 0, QoS 1, dan QoS 2. Tujuan pengujian ini adalah untuk mengevaluasi keandalan dan akurasi waktu pengiriman pesan pada berbagai tingkatan QoS MQTT. Pengukuran QoS jaringan dilakukan menggunakan aplikasi *Wireshark* yang diinstal pada laptop, yang juga berfungsi sebagai broker. Hal ini memungkinkan penangkapan lalu lintas jaringan MQTT oleh *Wireshark*. Pengujian dilakukan selama 30 menit untuk setiap skenario. Komunikasi perangkat menggunakan protokol jaringan WiFi yang disediakan oleh *Internet Service Provider* (ISP) Telkom di jaringan publik Perpustakaan Wilayah Provinsi Aceh pada lantai 2 pada tanggal 8 Juni 2023, antara pukul 09:00 hingga 12:30.

1) *Throughput: Perhitungan throughput* dalam pengujian MQTT menggunakan aplikasi *Wireshark*, diperlukan analisis data lalu lintas jaringan yang diambil oleh *Wireshark* selama komunikasi MQTT. Hal ini dilakukan dengan memisahkan data lalu lintas yang terkait dengan MQTT menggunakan *display filter* "mqtt". Dengan memeriksa paket data, termasuk ukuran pesan MQTT dan waktu yang diperlukan untuk transmisi, dapat dihitung nilai *throughput*. Tabel III menggambarkan hasil tangkapan data lalu lintas MQTT yang berguna untuk menghitung nilai *throughput.*

TABEL III
TANGKAPAN DATA *THROUGHPUT* DI *WIRESHARK*

	QoS 0	**QoS 1**	**QoS 2**
Packets	21324	37481	50170
Time span, s	1801,006	1797,124	1806,54
Bytes	2713644	3313068	3997408

Setelah diperoleh tangkapan data *throughput* untuk lalu lintas MQTT Langkah berikutnya membagi nilai *Bytes* (paket data diterima) dengan nilai *Time span, s* (lama pengamatan) lalu satuan diubah ke *Kilobit per second* (Kbps), sehingga didapat nilai *throughput* seperti pada Gambar 9.



Gambar 9. Grafik Nilai *Throughput*

Grafik di atas menunjukkan perbedaan nilai *throughput* pada tingkatan QoS MQTT. QoS 0 memiliki nilai *throughput* terendah, yaitu 12,05 Kbps, karena pengiriman tanpa konfirmasi yang sederhana. QoS 1 memiliki nilai *throughput* 14,75 Kbps lebih tinggi karena keandalan yang lebih tinggi dengan minimal pengiriman pesan sekali. QoS 2 memiliki nilai *throughput* tertinggi, yaitu 17,70 Kbps, karena keandalan tertinggi yang memerlukan paket yang kompleks.

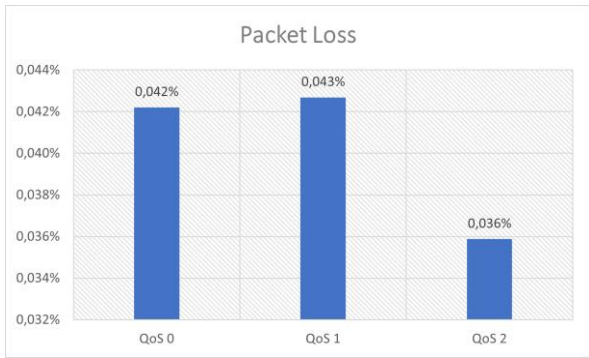
Secara singkat, grafik menunjukkan semakin tinggi tingkat QoS MQTT, semakin tinggi nilai *throughput*-nya. Peningkatan *throughput* menunjukkan pengelolaan *overhead* dan sumber daya yang efektif dalam tingkatan QoS yang lebih tinggi, memungkinkan peningkatan *throughput* pesan.

2) *Packet Loss:* Berbeda dengan *throughput*, perhitungan *packet loss* pada aplikasi *wireshark* menggunakan *display filter* "mqtt && tcp.analysis.lost_segment" untuk mendapatkan jumlah paket yang gagal dikirim. Pada Tabel IV di bawah merupakan hasil tangkapan lalu lintas MQTT untuk pesan yang gagal terkirim.

TABEL IV
TANGKAPAN DATA *PACKET LOSS* DI *WIRESHARK*

	QoS 0	**QoS 1**	**QoS 2**
Packets	9	16	18
Time span, s	1693,618	1710,674	1780,54
Bytes	5310	6840	6616

Setelah diperoleh tangkapan data *packet loss* untuk lalu lintas MQTT langkah berikutnya nilai *Packets* pada tangkapan diatas dan nilai *Packets* pada tangkapan *throughput* pada Tabel III dimasukan ke dalam rumus untuk mencari selisih antara keduanya dalam persentase, sehingga didapat nilai *packet loss* seperti pada Gambar 10.

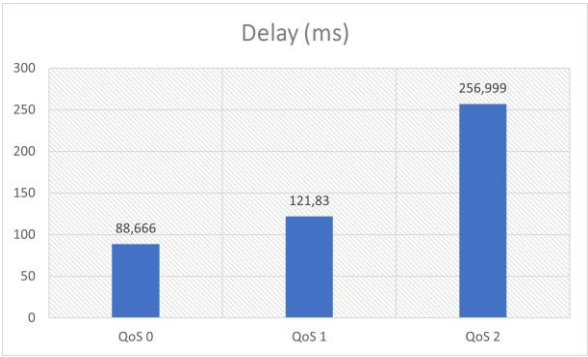


Gambar 10.Grafik Nilai *Packet Loss*

Grafik menampilkan nilai *packet loss* pada tingkatan QoS MQTT: QoS 0 memiliki nilai 0,042%, QoS 1 memiliki nilai 0,043%, dan QoS 2 memiliki nilai 0,036%. Ketiga tingkatan QoS MQTT menunjukkan persentase *packet loss* yang rendah dan termasuk dalam kategori "Sangat Bagus". Ini menegaskan keandalan protokol MQTT dalam pengiriman pesan. Namun, QoS 1 memiliki persentase *packet loss* sedikit lebih tinggi dari QoS 0 dan QoS 2, mungkin karena *overhead* proses *acknowledgment* dalam QoS 1. Sebaliknya, QoS 0

yang tidak menggunakan *acknowledgment* memiliki kehilangan yang lebih rendah daripada QoS 1. Sementara QoS 2 dengan pengiriman yang pasti memiliki persentase *packet loss* terendah. Meskipun QoS 1 memiliki sedikit peningkatan *packet loss*, total *packet loss* pada semua tingkatan QoS MQTT tetap rendah. Ini menunjukkan keandalan tinggi protokol MQTT dalam pengiriman pesan, dengan sebagian besar pesan berhasil dikirim tanpa kehilangan.

3) *Delay (Latensi): Pengukuran delay* untuk penelitian ini dilakukan terhadap antar pesan *publish*, ini bertujuan agar pengukuran keterlambatan dalam pengiriman pesan antar tingkatan QoS MQTT dapat lebih akurat. Pada pengukuran *delay display filter* diatur agar hanya menampilkan pesan *publish* MQTT, sehingga di dapat tangkapan lalu lintas hanya untuk pesan *publish* untuk masing-masing tingkatan QoS MQTT. Untuk mendapatkan nilai *delay* data tangkapan *Wireshark* diubah ke bentuk *ms.excel* dengan format *Comma-Separated Value* (CSV) yang kemudian dilakukan perhitungan untuk mencari *delay*, sehingga didapat nilai *delay* seperti pada Gambar 11.

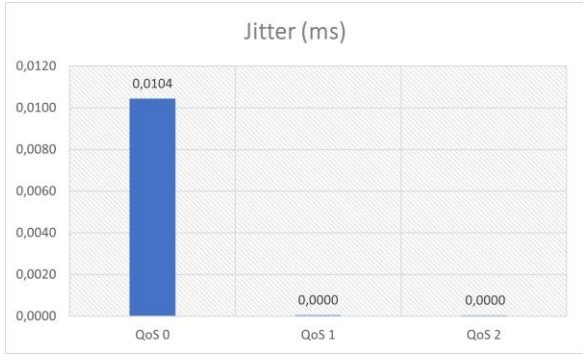


Gambar 11. Grafik Nilai *Delay*

Grafik menunjukkan nilai *delay* dalam milidetik (ms) untuk masing-masing tingkatan QoS MQTT. *Delay* untuk QoS 0, QoS 1, dan QoS 2 adalah 88,666 ms, 121,83 ms, dan 256,999 ms secara berurutan. QoS 0 dan QoS 1 termasuk kategori "Sangat Bagus," sedangkan QoS 2 masuk dalam kategori "Bagus." Grafik tersebut mengindikasikan bahwa semakin tinggi tingkatan QoS MQTT, nilai *delay* juga semakin tinggi terkait dengan tingkatan QoS tersebut.

QoS 0, yang menekankan pengiriman tanpa proses *acknowledgment*, memiliki *delay* terendah di antara tingkatan QoS lainnya. Ini karena QoS 0 lebih fokus pada kecepatan dan minim komunikasi tambahan untuk *acknowledgment*. Di sisi lain, QoS 1 dan QoS 2 menawarkan keandalan lebih tinggi dengan memastikan pengiriman pesan minimal satu kali atau tepat sekali. Proses *acknowledgment* dalam QoS ini menyebabkan nilai *delay* lebih tinggi daripada QoS 0.

4) *Jitter:* Sama seperti *delay*, *jitter* juga melakukan penangkapan data hanya untuk antar pesan *publish*. Sehingga diterapkan *display filter* yang sama dengan *delay*. Setelah data tersimpan dalam format CSV selanjutnya dilakukan perhitungan untuk mencari *jitter*, sehingga didapat nilai *jitter* seperti pada Gambar 12.



Gambar 12. Grafik Nilai *Jitter*

Grafik di atas menggambarkan nilai *jitter* dalam milidetik (ms) untuk setiap tingkatan QoS MQTT. Nilai *jitter* untuk QoS 0, QoS 1, dan QoS 2 adalah secara berurutan 0,0104 ms, 0 ms, dan 0 ms. Nilai *jitter* untuk ketiga tingkatan QoS MQTT dapat dikategorikan sebagai "Sangat Bagus." Grafik tersebut menunjukkan bahwa nilai *jitter* untuk ketiga tingkatan QoS MQTT sangat rendah, dimana *jitter* merujuk pada fluktuasi atau variasi dalam *delay*.

QoS 0 memiliki nilai *jitter* 0,0104 ms, mengindikasikan variasi *delay* yang rendah. Sementara itu, QoS 1 dan QoS 2 menunjukkan nilai *jitter* 0 ms, yang berarti tidak ada atau sangat sedikit variasi dalam *delay*. Hal ini menunjukkan kinerja yang konsisten dan stabil dalam pengiriman pesan, memastikan keandalan dan prediktabilitas. Nilai *jitter* yang rendah untuk semua tingkatan QoS MQTT menunjukkan kemampuan protokol ini untuk mengirimkan pesan dengan variasi *delay* yang minimal. Hal ini sangat penting untuk aplikasi atau sistem *real-time* yang membutuhkan pengiriman pesan yang konsisten dan dapat diprediksi.

5) *Akumulasi Nilai QoS Jaringan:* Pada bagian ini nilai-nilai parameter QoS jaringan akan diakumulasikan menggunakan indeks sesuai standar TIPHON untuk mengetahui kategori QoS jaringan untuk masing-masing tingkatan QoS MQTT. Hasil akumulasi dapat dilihat pada Tabel V.

TABEL V
AKUMULASI NILAI QOS JARINGAN

Parameter	QoS 0		QoS 1		QoS 2	
	In-deks	Hasil	In-deks	Hasil	In-deks	Hasil
Throughput (Kbps)	-	12,05	-	14,75	-	17,70
Packet loss (%)	4	0,042	4	0,043	4	0,036
Delay (ms)	4	88,666	4	121,83	3	256,999
Jitter (ms)	4	0,0104	4	0	4	0
Rata-rata Indeks	**4**		**4**		**3,67**	
Kategori	**Sangat Memuaskan**		**Sangat Memuaskan**		**Memuaskan**	

Pada tabel di atas hasil *throughput* memiliki nilai dibawah standar *tiphon* dikarenakan kebutuhan sistem pemantauan kualitas udara mengirimkan ukuran data yang kecil sehingga tidak dapat dimasukan dalam kategori buruk atau indeks 4 sehingga penentuan kualitas jaringan dapat didasari pada parameter lainnya. Dari 3 parameter lainnya dapat dilihat nilai akumulasi dari QoS Jaringan terhadap QoS MQTT, dimana QoS 0 dan QoS 1 masuk dalam kategori Sangat Memuaskan dengan rata-rata indeks 4 dan QoS 2 masuk dalam kategori Memuaskan dengan rata-rata indeks 3,67. Penilaian ini mencerminkan tingkat performa dan reliabilitas yang diharapkan dari masing-masing tingkatan QoS MQTT. QoS 0 dan QoS 1 diberi penilaian yang sangat memuaskan karena kemampuannya dalam memberikan layanan yang handal dan responsif dalam pengiriman pesan. Meskipun QoS 2 mendapatkan penilaian yang memuaskan, nilai tersebut menunjukkan tingkat kepuasan yang sedikit lebih rendah dibandingkan dengan QoS 0 dan QoS 1.

V. KESIMPULAN

Berdasarkan hasil percobaan, dapat disimpulkan bahwa sistem pemantauan kualitas udara ruangan berbasis IoT berjalan dengan baik. Data kualitas udara seperti suhu, kelembaban, intensitas cahaya, dan partikel debu $PM_{2,5}$ dapat dibaca dan ditransmisikan menggunakan protokol MQTT dengan andal. Pengguna dapat dengan mudah mengakses data melalui aplikasi di *smartphone*. Konsumsi energi dalam berbagai tingkatan QoS MQTT memiliki perbedaan yang relatif kecil, menunjukkan efisiensi penggunaan energi MQTT. Pengujian QoS jaringan menghasilkan nilai yang memuaskan untuk QoS 0 dan QoS 1 dengan *delay* minimal, *jitter* rendah, dan *packet loss* yang dapat ditoleransi. QoS 2 juga memuaskan, meskipun memiliki *delay* sedikit lebih tinggi. Secara keseluruhan, MQTT menunjukkan kemampuan QoS jaringan yang kuat, memastikan komunikasi efisien dan andal untuk aplikasi IoT, khususnya sistem pemantauan kualitas udara ruangan.

REFERENSI

[1] A. Alvarellos, A. L. Chao, J. R. Rabuñal, M. D. García-Vidaurrázaga, and A. Pazos, "Development of an Automatic Low-Cost Air Quality Control System: A Radon Application," *Appl. Sci.*, vol. 11, no. 5, p. 2169, Mar. 2021, doi: 10.3390/app11052169.

[2] H. Widowati, "Polusi Udara Sebabkan 7 Juta Kematian per Tahun di Dunia,"*databooks*,https://databoks.katadata.co.id/datapublish/2019/06/07/polusi-udara-sebabkan-7-juta-kematian-per-tahun-di-dunia (accessed Apr. 06, 2022).

[3] U. Navaseltsau, D. Navaseltsava, and V. Khaletski, "The use of mechanical ventilation systems with heat recovery to ensure air quality in residential premises," *E3S Web Conf.*, vol. 136, p. 05007, 2019, doi: 10.1051/e3sconf/201913605007.

[4] L. Fitria, R. Wulandari, and E. Hermawati, "Kualitas Udara Dalam Ruang Perpustakaan Universitas "X" Ditinjau Dari Kualitas Biologi, Fisik, dan Kimiawi," *Seri Kesehat. Health Ser. Vol 12 No 2 2008 Dec.*, vol. 12, Jan. 2010.

[5] "PMK No. 1077 ttg Pedoman Penyehatan Udara Dalam Ruang Rumah.pdf." Accessed: Aug. 15, 2023. [Online]. Available: http://hukor.kemkes.go.id/uploads/produk_hukum/PMK%20No.%201077%20ttg%20Pedoman%20Penyehatan%20Udara%20Dalam%20Ruang%20Rumah.pdf

[6] BMKG, "Informasi Konsentrasi Partikulat (PM2.5) | BMKG," *BMKG | Badan Meteorologi, Klimatologi, dan Geofisika.* https://www.bmkg.go.id/kualitas-udara/informasi-partikulat-pm25.bmkg (accessed Aug. 01, 2022).

[7] D. Setiadi and M. N. Abdul Muhaemin, "Penerapan Internet of Things (IoT) Pada Sistem Monitoring Irigasi (Smart Irigasi)," *Infotronik J. Teknol. Inf. Dan Elektron.*, vol. 3, no. 2, p. 95, Dec. 2018, doi: 10.32897/infotronik.2018.3.2.108.

[8] A. La Marra, F. Martinelli, P. Mori, A. Rizos, and A. Saracino, "Improving MQTT by Inclusion of Usage Control," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, G. Wang, M. Atiquzzaman, Z. Yan, and K.-K. R. Choo, Eds., in *Lecture Notes in Computer Science*, vol. 10656. Cham: Springer International Publishing, 2017, pp. 545–560. doi: 10.1007/978-3-319-72389-1_43.

[9] Z. Zhou, "Introduction to MQTT QoS 0, 1, 2," *www.emqx.com.* https://www.emqx.com/en/blog/introduction-to-mqtt-qos (accessed Jul. 23, 2023).

[10] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols," in *2019 IEEE International Conference on Industrial Technology (ICIT)*, Melbourne, Australia: IEEE, Feb. 2019, pp. 955–962. doi: 10.1109/ICIT.2019.8755050.

[11] R. Wulandari, "Analisis QoS (Quality of Service) Pada Jaringan Internet (Studi Kasus : UPT Loka Uji Teknik Penambangan Jampang Kulon – LIPI)," *J. Tek. Inform. Dan Sist. Inf.*, vol. 2, no. 2, Aug. 2016, doi: 10.28932/jutisi.v2i2.454.

[12] P. R. Utami, "Analisis Perbandingan Quality of Service Jaringan Internet Berbasis Wireless Pada Layanan Internet Service Provider (ISP) Indihome dan First Media," *J. Ilm. Teknol. Dan Rekayasa*, vol. 25, no. 2, pp. 125–137, 2020, doi: 10.35760/tr.2020.v25i2.2723.

[13] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, Agra, India: IEEE, Dec. 2016, pp. 1–5. doi: 10.1109/R10-HTC.2016.7906845.

[14] I. Setiawan, "Buku Ajar Sensor dan Transduser," *Diponegoro University: Faculty of Engineering*, 2009. Accessed: Aug. 19, 2023. [Online]. Available: http://eprints.undip.ac.id/4886/

[15] "What's the maximum size of EMQ publish payload? · Issue #1719 · emqx/emqx," *GitHub.* https://github.com/emqx/emqx/issues/1719 (accessed Jul. 25, 2023).

[16] "Throttling and Limits : HiveMQ Documentation." https://docs.hivemq.com/hivemq/3.4/user-guide/restrictions.html (accessed Jul. 25, 2023).

[17] "About: App Inventor for Android." https://dbpedia.org/page/App_Inventor_for_Android (accessed Jun. 10, 2023).

[18] "RD UM24 UM24C APLIKASI USB 2.0 LCD Display pengukur Tegangan Volt pengukur Amper Baterai Tegangan Current Meter Multimeter Kabel Mengukur Tester - AliExpress." https://id.aliexpress.com/item/32845522857.html (accessed Jun. 07, 2023

[19] U. Banerjee, A. Vashishtha, and M. Saxena, "Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection," *Int. J. Comput. Appl.*, vol. 6, no. 7, pp. 1–5, Sep. 2010, doi: 10.5120/1092-1427.

[20] V. Ndatinya, Z. Xiao, V. R. Manepalli, K. Meng, and Y. Xiao, "Network forensics analysis using Wireshark," *Int. J. Secur. Netw.*, vol. 10, no. 2, p. 91, 2015, doi: 10.1504/IJSN.2015.070421.