p-ISSN: 2502-1621
e-ISSN: 2656-161

Jurnal 7 Samudra
Politeknik Pelayaran
Surabaya
Vol. 9, No.2, November 2024
Hal: 53 - 60

Implementasi Protokol AMQP pada Manajemen Data *Automatic Identification System Multi-Station*

Ahmad Alam Ardiansyah[1], Afif Zuhri Arfianto[2], Dimas Pristovani Riananda[3], Zindhu Maulana Ahmad Putra[4], Edy Setiawan[5], Adianto[6]

[1256]Teknik Otomasi, Politeknik Perkapalan Negeri Surabaya
[34]Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya

ahmadalamardiansyah@gmail.com

ABSTRAK

Tingginya angka kecelakaan kapal di Indonesia memerlukan sistem navigasi dan komunikasi yang andal. Salah satunya adalah *automatic identification system* (AIS), pengembangan AIS sampai saat ini masih terus dilakukan. Implementasi protokol *advanced message queuing protocol* (AMQP) pada manajemen data *automatic identification system* (AIS) *multi-station* dilakukan untuk meningkatkan efisiensi dan keamanan navigasi maritim. AIS PPNS sebagai objek penelitian memiliki beberapa station penerima data AIS yang tersebar di berbagai wilayah Indonesia seperti Surabaya, Semarang, dan Batam. Metode yang digunakan meliputi identifikasi masalah, studi literatur, analisis kebutuhan sistem, perancangan, implementasi, dan evaluasi performa sistem. RabbitMQ berperan sebagai broker dalam pengaturan pengiriman pesan AIS dari *producer* (*station* AIS) ke *consumer* (server AIS PPNS). Hasil pengujian menunjukkan bahwa manajemen data AIS dengan protokol AMQP mampu menangani trafik data yang tinggi dengan performa yang baik. *Throughput* pengiriman pesan stabil antara 2500 hingga 3500 pesan per detik, dan latensi pesan tetap rendah dan stabil, menunjukkan sistem yang dibangun dapat meningkatkan efisiensi dalam *monitoring* aktivitas kapal.

Kata kunci : AIS, AMQP, Manajemen Data, Performa, RabbitMQ

PENDAHULUAN

Angka kecelakaan kapal di Indonesia pada tahun 2003 - 2019 masih cukup tinggi, berdasarkan data yang dirilis oleh KNKT (Komite Nasional Keselamatan Transportasi) sebanyak 120 kejadian yang menyebabkan 513 korban jiwa, 726 korban luka, dan 701 korban hilang. Kecelakaan kapal tersebut sebagian besar terjadi karena tubrukan kapal sebanyak 22 kejadian dan kapal tenggelam sebanyak 28 kejadian, dimana 44 kecelakaan terjadi di perairan Laut Jawa [1].

Berdasarkan kejadian tersebut dibutuhkan peranti navigasi untuk penyedia informasi kapal dan sebagai sistem komunikasi supaya

kapal dapat terhubung dengan *station* di daratan. Salah satu peranti navigasi adalah AIS (Automatic Identification System) yang merupakan sebuah *automatic vessel tracking system* atau sistem pelacakan kapal otomatis [2]. Penggunaan sistem ini sudah diatur oleh IMO (International Maritime Organization) dalam *International Convention for the Safety Life of Sea* (SOLAS), *Chapter V* tentang *Safety of Navigation* [3]. Pengembangan AIS terus dilakukan sampai saat ini, salah satunya adalah pengembangan pada sistem manajemen data AIS. AIS PPNS sebagai objek penelitian memiliki beberapa *station* untuk menerima data AIS yang tersebar di berbagai wilayah Indonesia, seperti di Surabaya, Madura, Semarang, Banyuwangi, Indramayu, dan Batam. Setiap *station* menghasilkan data AIS dengan trafik yang bervariatif tergantung pada banyaknya kapal di daerah tersebut. Data-data AIS yang didapatkan akan dikirimkan ke dalam server AIS PPNS untuk *logging* dan *monitoring*.

Berdasarkan kebutuhan tersebut perlu dilakukan manajemen data yang baik dalam pengiriman data ke server menggunakan sebuah protokol pengiriman. Protokol pengiriman dari *multi-station* ke server yang digunakan adalah *Advanced Message Queuing Protocol* (AMQP), sebuah *messaging middleware* yang memungkinkan perangkat komputer saling berkomunikasi dan bertukar pesan secara efisien dan andal [4]. AMQP berperan dalam memastikan pengiriman data AIS dari *multi-station*, dengan menggunakan AMQP, setiap AIS *station* bertindak sebagai *producer* yang mengirimkan data ke *broker* untuk kemudian data didistribusikan ke *consumer* untuk *logging* dan *monitoring* atau dalam konteks ini adalah AIS PPNS.

Manajemen data AIS ini hanya mencakup 3 dari 6 *station* yang meliputi *station* Surabaya, Semarang, dan Batam. Dari ketiga *station* tersebut dikirimkan ke *broker* sebagai *exchange*, kemudian diterima oleh satu *consumer* untuk *logging* dan *monitoring*. Dengan pendekatan ini, diharapkan sistem dapat menangani trafik data AIS yang tinggi dengan performa yang baik.

Mengacu pada permasalahan dan rencana penyelesaiannya, tujuan dari penelitian ini adalah untuk membangun sistem manajemen data AIS *multi-station* menggunakan protokol AMQP dan mengevaluasi performa AMQP dalam menangani data AIS dengan trafik yang tinggi.

Dalam melakukan penelitian ini diperlukan beberapa landasan teoritik yang akan memberikan dasar yang kuat untuk membangun dan mengevaluasi sistem manajemen data AIS. Landasan teori terbagi menjadi tiga yaitu, AIS, AMQP, dan RabbitMQ sebagai *cloud-AMQP*. Berikut landasan teoritik dalam penelitian ini.

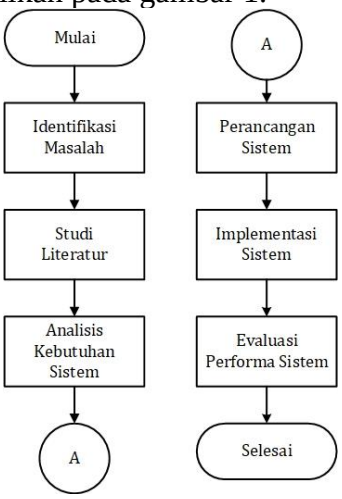
Automatic Identification System (AIS) adalah sistem pelacakan otomatis yang digunakan untuk memonitor pergerakan kapal di lautan [5]. *Multi-station* AIS memungkinkan *monitoring* kapal dari berbagai lokasi sekaligus, memberikan cakupan yang lebih luas dan data yang lebih komprehensif mengenai lalu lintas maritim [6]. AIS *multi-station* ini terdiri dari beberapa *station* penerima yang tersebar di berbagai lokasi di Indonesia, yang kemudian mengirimkan data ke server pusat AIS PPNS untuk analisis dan monitoring lebih lanjut. Penggunaan *multi–station* AIS ini penting untuk meningkatkan keamanan dan efisiensi navigasi maritim.

Advanced Message Queuing Protocol (AMQP) adalah standar terbuka untuk *messaging middleware* yang memungkinkan sistem komputer saling berkomunikasi dan bertukar pesan secara efisien dan andal [7]. AMQP menyediakan fitur seperti *message orientation*, *queuing*, *routing*, *reliability*, dan *security* yang penting dalam manajemen data lintas jaringan [8].

RabbitMQ adalah salah satu implementasi dari AMQP yang populer digunakan dalam aplikasi *cloud*. RabbitMQ memungkinkan pengelolaan pesan yang *scalable* dan *reliable*, kompatibel untuk aplikasi yang membutuhkan pengolahan data dalam jumlah besar dan berkecepatan tinggi seperti AIS [9]. RabbitMQ mendukung berbagai fitur AMQP dan dapat diintegrasikan dengan berbagai aplikasi lain untuk memastikan alur data yang efisien dan aman [10].

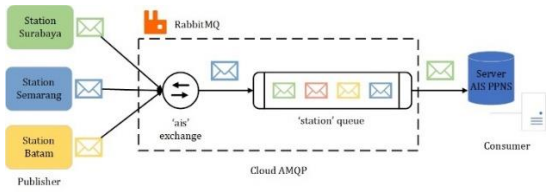
METODE PENELITIAN

Tahapan untuk melakukan penelitian ini dimulai dari identifikasi masalah, studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, dan evaluasi performa sistem. Tahapan penelitian ini ditampilkan pada gambar 1.



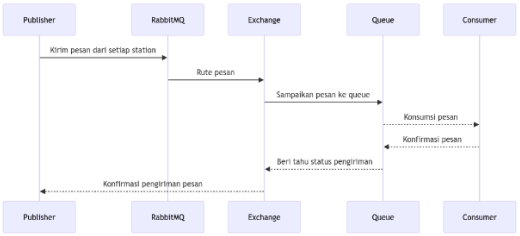
Gambar 1. *Tahapan Penelitian*

Diagram blok sistem menjelaskan tentang gambaran umum penelitian ini. Dimulai dari tiga *station* (Surabaya, Semarang, dan Batam) yang berfungsi sebagai *publisher* atau pengirim pesan AIS, RabbitMQ sebagai *broker* atau *cloud AMQP* sebagai perantara antara pengirim dan penerima pesan, dan server AIS PPNS sebagai *consumer* atau penerima pesan AIS. Diagram blok sistem ditampilkan pada gambar 2.



Gambar 2. *Diagram Blok Sistem*

Diagram alir sistem menjelaskan tentang urutan sistem ini akan berjalan. Dimulai dari pengiriman pesan AIS oleh setiap *station*, rute atau *routing* pesan berdasarkan *routing key* dan *exchange*, menyampaikan pesan ke *queue*, sampai pesan di konsumsi oleh server. Selanjutnya *consumer* akan memberikan *feedback* berupa *acknowledgement* untuk memastikan bahwa pesan telah diterima. Diagram alir sistem ditampilkan pada gambar 3.



Gambar 3. *Diagram Alir Sistem*

HASIL DAN PEMBAHASAN

Pada bagian hasil dan pembahasan dibagi menjadi empat bagian, yaitu *producer*, *broker*, *consumer*, dan performa.

Producer

Dalam perancangan *producer* mengimplementasikan *package* ‘amqplib’ untuk berinteraksi dengan RabbitMQ. Setiap *station* terdiri dari AIS *receiver*, komputer, dan jaringan internet. Data AIS yang didapatkan oleh *receiver* diterima komputer lewat AIS Dispatcher. AIS Dispatcher adalah sebuah AIS *data forwarding utility* yang dapat *dispatches* NMEA *stream* via UDP. Selanjutnya dilakukan konfigurasi pada AIS Dispatcher untuk *forward* data AIS ke *address* 127.0.0.1 dan port 2024. *Address* tersebut yang dijadikan *input datastream* untuk dikirimkan ke *exchange* RabbitMQ. Perancangan sistem *producer* dibagi menjadi 3 bagian yaitu *datastream*, *producer*, dan *app* sebagai berikut.

- ***datastream***

Datastream difungsikan untuk *stream* pesan AIS yang sudah di-*forward* oleh AIS Dispatcher. Pada bagian ini menggunakan bantuan *package* ‘dgram’ untuk melakukan *binding* ke *address* dan *port* dari *dispatcher* dan menangani *streaming data* AIS supaya dapat dipanggil oleh *producer*. Berikut *script* dari *datastream*.

```
function datastream(port, address, onDataReceived) {
    const udpStream = createUdpStream();
    udpStream.pipe(split()).on('data', data => {
        onDataReceived(data);
    });
    server.on('listening', () => {server.address()});
    server.bind({address:
```

```
address, port: port});
};
```

- ***producer***

Producer digunakan untuk membuat koneksi kanal dan *publish* pesan ke *cloud* RabbitMQ. Pada bagian ini dilakukan *publish* pesan berdasarkan koneksi, kanal, *routing key*, dan *exchange*. Perancangan *publish* pesan dimasukkan ke dalam fungsi *pub(message)*, *message* adalah parameter yang didapat dari *datastream*. Dalam fungsi ini dilakukan pengecekan dan pembuatan kanal, mendaftarkan *exchange*, pembuatan variabel konstan berisi tipe pesan sesuai dengan *routing key* dan isi pesan (*message*), dan *publish* ke *broker* sesuai *exchange* dan *routing key*. Berikut *script* dari *producer*.

```
async pub(message) {
     if (!this.channel) {await
this.createChannel();}
     await
this.channel.assertExchange(exc
hange, 'direct', {
          durable: true});
     const msg = {type:
routingKey,message: message};
     await
this.channel.publish(exchange,
routingKey,

     Buffer.from(JSON.stringify
(msg)),
          {persistent: true}
     );
}
```

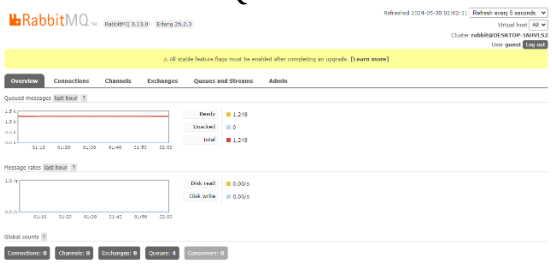
- ***app***

app difungsikan untuk mengintegrasikan *datastream* dan *producer*. Kedua bagian tersebut dipanggil di bagian ini tujuannya untuk mempermudah *running* sistem dalam pengiriman pesan ke *cloud* RabbitMQ. Berikut adalah *script* dari *app*.

```
const ADDRESS =
config.udp.address;
const PORT = config.udp.port;
datastream(PORT, ADDRESS, data
=> {producer.pub(data);});
```

Broker

Dalam perancangan *broker* menggunakan RabbitMQ dilakukan penerimaan, penyimpanan, dan penerusan pesan. *Broker* diinstalasi di komputer yang memiliki *IP public* dengan *address* 103.24.49.246 tujuannya untuk kemudahan akses produser dan konsumer untuk membuat koneksi dengan *broker* atau *cloud* AMQP. Ditampilkan pada gambar 4 *overview* dari *cloud* RabbitMQ.



Gambar 4. *Overview RabbitMQ*

Untuk menjalankan (*start*), memastikan (*status*), menghentikan (*stop*), dan memulai ulang (*restart*) RabbitMQ sebagai *broker* dilakukan menggunakan *command-line argument* sebagai berikut.

start

```
sudo systemctl start rabbitmq-
server
```

status

```
sudo systemctl status rabbitmq-
server
```

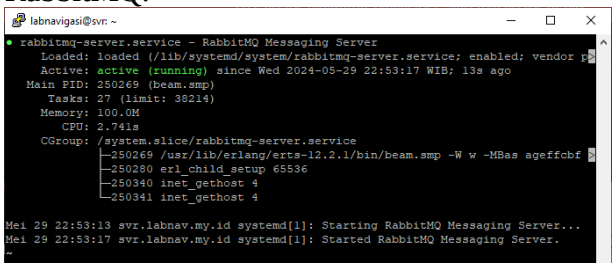
stop

```
sudo systemctl stop rabbitmq-
server
```

restart

```
sudo systemctl restart rabbitmq-
server
```

Pada gambar 5 ditampilkan status *running* RabbitMQ.



Gambar 5. *Status Running RabbitMQ*

Consumer

Perancangan *consumer* difungsikan untuk

penerimaan pesan AIS yang dikirimkan oleh *producer* ke *broker*, *consumer* akan mengonsumsi pesan dari *broker* berdasarkan *uri*, *exchange*, *routing key*, dan *queue*. *Consumer* diaplikasikan pada satu sistem dengan *backend* pada server AIS PPNS, tujuannya untuk pengolahan lebih lanjut seperti penyimpanan pesan AIS ke dalam *database* dan *monitoring* setiap *station*.

Pada bagian ini dilakukan konsumsi pesan yang di dalam fungsi *sub(callback)*, *callback* merupakan parameter yang akan dioperasikan lebih lanjut oleh server *backend*. Dalam fungsi ini, dilakukan pengecekan apakah kanal sudah dibuat atau belum. Selanjutnya adalah mendaftarkan *exchange* dengan opsi *direct* dan *durable:true* artinya jika koneksi terputus pesan masih tertahan di *exchange*, pendaftaran *exchange* menggunakan *assertExchange*. Ketika berhasil, dilakukan pendaftaran *queue* yang digunakan sebagai rute pesan akan dikonsumsi lewat antrean mana, dan terakhir adalah *consume* pesan dari *broker* dengan rute sesuai *exchange*, *binding queue*, dan *routing key*. Ditambahkan juga pemanggilan *callback* di dalam fungsi *consume*, tujuannya untuk *forwarding* data yang diterima untuk disimpan dalam *database server* AIS PPNS. Berikut adalah *script* yang merepresentasikan hasil perancangan dari *consumer*.

```
async sub(callback) {
     if (!this.channel) {await this.createChannel()}
     await this.channel.assertExchange(exchange, 'direct', {
          durable: true
     });
     const q = await this.channel.assertQueue(queueName, {
          durable: true
     });
     await this.channel.bindQueue(q.queue, exchange, routingKey);
     this.channel.consume(q.queue, (msg) => {
          const data = JSON.parse(msg.content);
          callback(data.message);

          this.channel.ack(msg);
     });
}
```

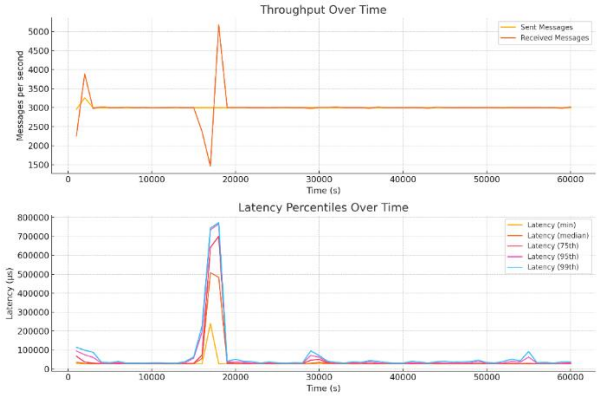
Performa

Selama pengujian performa dari protokol amqp oleh RabbitMQ, *tools* yang digunakan adalah *rabbitmq-perf-test* untuk *benchmark* dan mengukur metrik performa dan *rabbitmq_management* untuk monitoring aktivitas setiap *station*.

Dalam pengujian performa menggunakan *rabbitmq-perf-test* perlu dilakukan *running test* menggunakan *argument* sebagai berikut:

```
java -jar perftest-2.21.0.jar --uri amqp://guest:1012@labnav.my.id:5672 --producers 3 --consumers 1 --rate 1000 --time 60
```

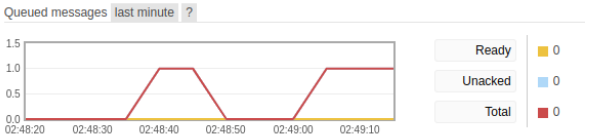
Setelah melakukan *running test* selama 60 detik berikut hasil pengujian berupa *throughput over time* dan *latency percentiles over time* ditampilkan pada gambar 6.



Gambar 6. *Hasil Pengujian Benchmark dan Metrik Performa*

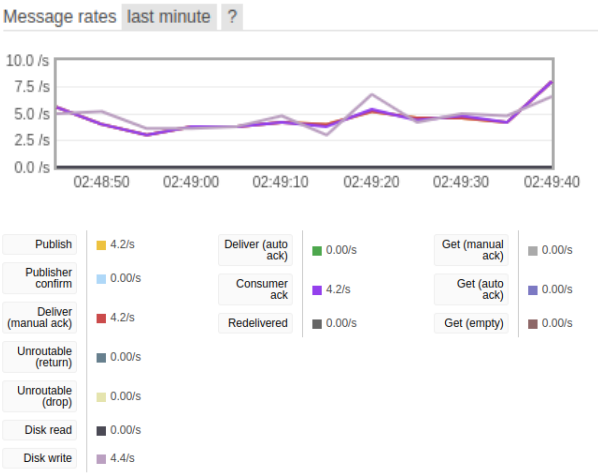
Pada grafik *throughput over time* sebagian besar waktu *throughput* untuk pesan yang dikirim dan diterima cukup stabil, berkisar antara 2500 sampai 3500 pesan per detik. Hal ini menunjukkan sistem mempertahankan laju pengiriman dan penerimaan pesan yang konsisten dalam kondisi normal. Selanjutnya pada grafik *latency percentiles over time* menunjukkan persentil latensi minimum, median, 75th, 95th, dan 99th selam *running test*

tetap rendah dan stabil berkisar antara 0 hingga 100000 mikrodetik (100 ms). Hal ini menunjukkan bahwa dalam kondisi normal, sistem dapat memproses pesan dengan latensi rendah. Namun dalam kedua grafik menunjukkan lonjakan pada sekitar waktu 20000 detik disebabkan oleh beban kerja yang tiba-tiba meningkat dan perubahan konfigurasi. Setelah pengujian metrik performa, dilakukan analisis pada RabbitMQ Management Dashboard berdasarkan aktivitas setiap *node* yang meliputi *queued message* satu menit terakhir dan *message rates* satu menit terakhir. Berikut adalah hasilnya.



Gambar 7. *Hasil Queued Message Last Minute*

Grafik pada gambar 7 menunjukkan total antrean pesan (*ready*, *unacked*, dan *total*) selama satu menit terakhir. Jumlah total pesan berfluktuasi antara 1 dan 0 menunjukkan bahwa pesan diproses dengan cepat. Tidak ada pesan dalam keadaan '*ready*' dan '*unacked*' menunjukkan bahwa tidak ada pesan yang menunggu untuk diproses dan semua pesan dikonfirmasi (*acknowledge*) dengan segera.



Gambar 8. *Hasil Message Rates Last Minute*

Pada gambar 8 menunjukkan grafik *message rates* per detik selama satu menit terakhir. Laju *publish* pesan stabil di angka 4.2/s, *manual ack* memiliki laju yang sama dengan *publish* yaitu 4.2/s. Sedangkan untuk *disk write* di angka 4.4/s menunjukkan aktivitas *disk* yang konsisten, bagian ini krusial untuk persistensi pesan. Selanjutnya adalah *consumer ack* memiliki laju 4.2/s yang memiliki nilai paramater yang sama dengan *manual ack* dan *publish*. Untuk parameter yang lain tidak menunjukkan laju pesan seperti pada *publisher confirm*, *unroutable (return)*, *unroutable (drop)*, *disk read*, *deliver (auto ack)*, *redelivered*, *get (manual ack, auto ack, empty)*

KESIMPULAN

Manajemen data AIS menggunakan protokol AMQP mampu mengelola trafik data AIS yang tinggi. Sistem ini berhasil menangani pengiriman data dari beberapa *station* (Surabaya, Semarang, dan Batam) ke server pusat AIS PPNS melalui *broker* RabbitMQ, yang berfungsi sebagai perantara dalam komunikasi data. Pengujian performa menunjukkan bahwa *throughput* pengiriman pesan berkisar antara 2500 hingga 3500 pesan per detik, serta latensi yang tetap rendah dan stabil pada berbagai persentil. Hal ini menunjukkan bahwa sistem manajemen data AIS yang dibangun mampu memenuhi kebutuhan pengiriman data *real-time* dengan kinerja yang baik, sehingga dapat meningkatkan efisiensi dalam *monitoring* aktivitas kapal.

Pengembangan lanjutan masih dapat dilakukan seperti seperti penambahan *producer* untuk *station-station* lain sehingga area *monitoring* akan lebih luas dan tersebar. Selain itu, penambahan keamanan SSL/TLS dalam pengiriman data juga menjadi salah satu peluang untuk mengembangkan sistem manajemen ini.

DAFTAR PUSTAKA

A. D. Saputra, "Studi Kecelakaan Kapal di

Indonesia dari Tahun 2003-2019 Berdasarkan Data Investigasi Komite Nasional Keselamatan Transportasi," *Warta Penelitian Perhubungan*, vol. 33, no. 2. 2021. doi: 10.25104/warlit.v33i2.1502.

D. Yang, L. Wu, S. Wang, H. Jia, and K. X. Li, "How big data enriches maritime research – a critical review of Automatic Identification System (AIS) data applications," *Transp. Rev.*, vol. 39, no. 6, pp. 755–773, Nov. 2019, doi: 10.1080/01441647.2019.1649315.

Muhammad Syahid Messiah, "Pengembangan Web Based User Interface pada Automatic Identification System (AIS) Berbasis Angular dan Open Street Map," 2023.

N. Q. Uy and V. H. Nam, "A comparison of AMQP and MQTT protocols for Internet of Things," in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, 2019, pp. 292–297. doi: 10.1109/NICS48868.2019.9023812.

S. Enda, Depandi. Kurniati, Rezki. Mawarni, "PENGEMBANGAN APLIKASI FRONT-END MONITORING LALU LINTAS KAPAL DI PERAIRAN SELAT MALAKA BERBASIS DATA AIS Jurusan Teknik Informatika , Politeknik Negeri Bengkalis , Jl . Bathin Alam , Sungai PENDAHULUAN Selat malaka merupakan salah satu bentangan laut tersi," *Semin. Nas. Terap. Ris. Inov. (SENTRINOV)Ke-9, Ser. Eng. Sci.*, vol. 9, no. 1, pp. 272–282, 2023, [Online]. Available: https://proceeding.isas.or.id/index.php/sentrinov/article/view/1298/659

M. Abiraihan, R. Marta, Syukhri, and H. K. Saputra, "Designing a Desktop Application for Ship Monitoring and AIS Data Storage Based on RTL-SDR and Raspberry Pi Using Python and PyQt," *J. Hypermedia Technol. Learn.*, vol. 2, no. 2 SE-Articles, pp. 93–111, Feb. 2024, doi: 10.58536/j-hytel.v2i2.118.

C. B. Gemirter, Ç. Şenturca, and Ş. Baydere, "A Comparative Evaluation of AMQP, MQTT and HTTP Protocols Using Real-Time Public Smart City Data," in *2021 6th International Conference on Computer Science and Engineering (UBMK)*, 2021, pp. 542–547. doi: 10.1109/UBMK52708.2021.9559032.

C. S. Krishna and T. Sasikala, "Healthcare Monitoring System Based on IoT Using AMQP Protocol BT - International Conference on Computer Networks and Communication Technologies," S. Smys, R. Bestak, J. I.-Z. Chen, and I. Kotuliak, Eds., Singapore: Springer Singapore, 2019, pp. 305–319.

U. Rahardja, "Skema Catatan Kesehatan menggunakan Teknologi Blockchain dalam Pendidikan," *J. MENTARI Manajemen, Pendidik. dan Teknol. Inf.*, vol. 1, no. 1, pp. 29–37, 2022, doi: 10.33050/mentari.v1i1.134.

D. R. Agustina, A. Y. Vandika, W. Susanty, T. Tanjung, and R. Nur Afiani, "Implementasi Service Data untuk Pemantauan Lighting pada Smart Agriculture," *Digit. Transform. Technol.*, vol. 3, no. 2, pp. 380–388, 2023, doi: 10.47709/digitech.v3i2.2851.