JIKO (JURNAL INFORMATIKA DAN KOMPUTER)
Juni 2025, Volume: **9**, No. **2** | Pages 463–473
doi: 10.26798/jiko.v9i2.1934
e-ISSN : 2477-3964 – p-ISSN : 2477-4413

ARTICLE

Sistem IoT Pengubah Informasi Running Text Berbasis MQTT dan *FreeRTOS*

IoT System for Changing Running Text Information Based on MQTT and FreeRTOS

Totok Budioko[*,1] dan Berta Bednar[2]

[1]Teknik Komputer, Fakultas Teknologi Informasi, Universitas Teknologi Digital Indonesia, Yogyakarta, Indonesia
[2]Teknologi Komputer, Fakultas Teknologi Informasi, Universitas Teknologi Digital Indonesia, Yogyakarta, Indonesia
*Penulis Korespondensi: budioko@utdi.ac.id

(Disubmit 25-04-25; Diterima 26-05-25; Dipublikasikan online pada 20-06-25)

Abstrak

Salah satu media yang sering digunakan untuk memberikan informasi adalah *Running Text*. Untuk *Running Text* yang jumlahnya banyak dan jaraknya jauh akan kesulitan jika harus mengubah informasi yang tampil secara langsung karena pengubahan dilakukan dengan Wifi lokal, atau *Bluetooth* yang jarak aksesnya hanya beberapa meter. Sistem IoT Pengubah Informasi *Running Text* Berbasis MQTT dan *FreeRTOS* terdiri atas *Running Text* yang diimplementasikan menggunakan ESP32, Modul *LED Dotmatix* tipe FC-16, Borker MQTT, dan aplikasi *Desktop* untuk mengubah informasi *Running Text*. Pengubahan informasi dapat dilakukan untuk semua *Running Text* degan informasi yang sama maupun untuk masing-masing *Running Text*. Pemrograman pada *Running Text* menggunakan *multitasking* basis *FreeRTOS* sedangkan aplikasi *Dekstop* menggunakan Pyhton dengan pustaka TKInter. Sistem berhasil dirancang dan diimplementasikan dengan tiga buah *Running text*, penampil dot matrik ukuran 8x32 dan berhasil diubah informasi yang tampil menggunakan aplikasi *Desktop*. Berdasarkan hasil pengujian waktu kirim informasi tidak dipengaruhi oleh jumlah karakter. Rata-rata waktu dikirim 0,406 detik.

Kata kunci: MQTT; IoT; FreeRTOS; DotMatrix; RunningText

Abstract

One of the media that is often used to provide information is Running Text. For Running Texts that are many and far away, it will be difficult if you need to change the information that appears directly, because the changes are made using local Wifi or Bluetooth whose access distance is only a few metres. The MQTT and FreeRTOS based Running Text Information Changing IoT System consists of Running Text implemented using ESP32, Dotmatix LED Module type FC-16, MQTT Borker and a desktop application to change Running Text information. Changing information can be done for all Running Texts with the same information or for each Running Text. Programming on Running Text uses FreeRTOS based multitasking, while the desktop application uses Python with TKInter. The system was successfully designed and implemented with three Running Texts, an 8x32 dot matrix viewer and successfully changed the displayed information using the desktop application. Based on the test results, the time to send information is not affected by the number of characters. The average transmission time is 0.406 seconds.

KeyWords: MQTT; IoT; FreeRTOS; DotMatrix; RunningText

How to Cite: T. Budioko *et al.*, "Sistem IoT Pengubah Informasi Running Text Berbasis MQTT dan FreeRTOS", *JIKO (JURNAL INFORMATIKA DAN KOMPUTER)*, Volume: **9**, No.**2**, Pages 463–473, Juni 2025, doi: 10.26798/jiko.v9i2.1934.

1. Pendahuluan

Running Text merupakan salah satu media yang digunakan untuk menampilkan informasi. Informasi yang ditampilkan dapat berupa iklan, jadwal sholat, pemandu arah atau himbauan. *Running Text* sering digunakan di supermarket, toko, kampus, sekolah, hotel, gedung pernikahan, lampu lalu lintas atau tempat-tempat lainnya yang membutuhkan informasi. *Running Text* membuat orang lebih tertarik membaca informasi karena warna dan animasinya.[1],[2],[3], [4], [5], [6], [7].

Pengubahan informasi *Running Text* pada sistem konvensional dilakukan secara langsung pada sistem *Running Text* atau mengunakan PC yang dikoneksikan menggunakan interface USB. Pada sistem modern pengubahan informasi pada *Running Text* menggunakan PC atau smartphone menggunakan koneksi wirelesss dengan protokol *Bluetooth, LoRa, Wifi,* atau *SMS*. Pengubahan informasi menggunakan koneksi *Bluetooth* hanya dapat dilakukan terbatas sampai jarak 15m [2], [4], [6]. Koneksi menggunakan *LoRa* dengan topologi *Point to Point* dapat mengubah informasi *Running Text* sampai jarak 60m. Jarak akan bertambah jika dikonfigurasi menggunakan jaringan *LoRa*[1], walaupun secara teoritis komunikasi *LoRa* mencapai beberapa ratus kilometer[7]. Koneksi menggunakan *Wifi* dengan topologi *Point to Point* hanya dapat diakses sampai jarak sekitar 100m[7]. Namun jika *Wifi* terkoneksi dengan jaringan internet maka akan dapat diakses dari manapun selama ada koneksi internet[3],[5]. Untuk koneksi melalui *SMS* atau jaringan telepon seluler tergantung dari infrastruktur seluler yang ada dan secara umum dapat diakses secara global[7].

Sistem pengubahan informasi *Running Text* yang menggunakan *Bluetooth, LoRa* dan *SMS* betrsifat *Point to Point* antara PC atau *Smartphone* dan *Running Text*, sehingga hanya dapat mengubah untuk satu sistem *Running Text*. Penelitian [1] yang menggunakan *LoRa* telah dapat melakukan pengubahan untuk beberapa *Running Text* dengan teknik *multihop*. Untuk menambah jarak dan jumlah *Running Text* yang akan diubah informasinya dapat menggunakan *Internet of Things (IoT)*. Salah satu protokol *IoT* yang digunakan adalah protokol *Message Queuing Telemetry Transport (MQTT)* . Protokol ini bekerja di atas jaringan TCP/IP sehingga dapat diakses dari jaringan internet. Implemetasi sistem pengubah informasi *Running Text* yang menggunakan protokol *MQTT* dan koneksi *Wifi*, telah diteliti seperti pada artikel[3], [5].

Protokol *MQTT* telah banyak digunakan di berbagai aplikasi seperti pada artikel [8] yang menggunakan protokol *MQTT* untuk mengirimkan data tingkat ketinggian air sungai dari beberapa titik. Protokol ini banyak digunakan pada aplikasi yang berhubungan dengan kesehatan seperti pada artikel [9], digunakan untuk memantau cairan infus dengan mengukur berat cairnnya. Sedangkan pada artikel [10] digunakan untuk memonitor mesin pemotong tortilla.

Pada penelitian[3] berhasil membuat sistem pengubah informasi *Running Text* berbasis protokol *MQTT* yang mampu mengirimkan informasi pengubahan melalui protokol *MQTT* dari *smartphone* ke satu *Running Text* dengan model pemrograman *single task*. Pada peneltian [5] berhasil mengintegrasikan antara *MQTT* dan *Google Asisten*, sehingga pengubahan informasi dapat dilakukan menggunakan suara dari *smartphone* ke satu *Running Text* dan model pemrograman *single task*.

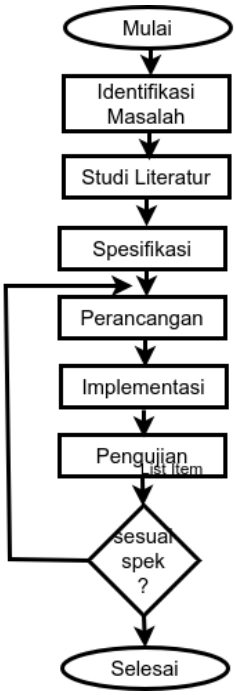
Implementasi sistem *Running Text* dari beberapa penelitian sebelumnya menggunakan model pemrograman *single task*. Penggunaan *single task* akan menyulitkan terutama ketika menangani masukan dan keluaran serta komunikasi data *internet*, namun tidak mengganggu proses yang lainnya. Pengunaan sistem operasi waktu nyata *(realtime operating system)* akan meningkatkan ketangguhan sistem sehingga pengubahan informasi pada *Running Text* akan terjamin keberhasilan dan keakuratannya. Sistem operasi waktu nyata yang banyak digunakan dan bersifat sumber terbuka serta bebas *(Open Source and Free)* adalah *FreeRTOS*. Beberapa penelitian yang menggunakan*FreeRTOS* untuk implementasi aplikasinya antara lain, penelitian Septian dan Arkan [11] digunakan untuk pengukuran parameter lingkungan seperti kelembaban, suhu dan gas CO2, artikel [12] digunakan untuk piranti perekam bioelektrik dan artikel [8] digunakan untuk monitoring level ketinggian air sungai.

Penelitian ini akan melengkapi beberapa penelitian yang sudah pernah dilakukan pada sistem pengubah informasi *Running Text*. Sistem dirancang menggunakan arsitektur basis protokol *MQTT* untuk mengubah beberapa sistem *Running Text* yang lokasinya berbeda-beda. Program pada sistem *Running Text* diimplementasikan model pemrograman *multitask* pada *platform* sistem operasi *freeRTOS*.

2. Metode

Metodologi penelitian seperti pada Gambar 1. Pertama melakukan identifikasi masalah, yaitu bagaimana mengubah informasi pada *Running Text* yang jumlahnya banyak serta lokasi yang jauh menggunakan infrastruktur internet. Kemudian melakukan Studi Literatur dengan menelaah beberapa publikasi yang berhubungan dengan masalah yang sudah diidentifikasi. Beberapa publikasi yang telah menghasilkan sistem pengubah informasi *Running Text* [1],[2], [3],[4], [4], [5], [6], [7]. Spesifikasi sistem *Running Text* pada penelitian ini adalah *dot matrix* menggunakan modul FC-16 menggunakan antarmuka *Serial Peripheral Interface* (SPI) dengan IC seri MAX7219. Ukuran pixelnya LED 8x32 (barisxkolom). Koneksi internet melalui jaringan *WiFi* yang tertanam pada *System on Chip* (SoC) ESP32. *Dashboard* menggunakan aplikasi *Desktop*. Arsitektur sistem *IoT* menggunakan basis protokol *MQTT*.

Protokol *MQTT* merupakan protokol yang ringan dengan model *client server*. Server pada protokol *MQTT* berfungsi sebagai *broker* yang akan meneruskan data dari satu *client* ke *client* yang lainnya. Komunikasi data pada protokol *MQTT* menggunakan model *Publish-Subcribe*. *Client* yang mengirimkan data melakukan *Publish* ke *broker/MQTT Server* dan *client* yang akan menggunakan data melakukan *Subcribe*. C *Client Publish dan Subcribe* pada topik yang sama [13].



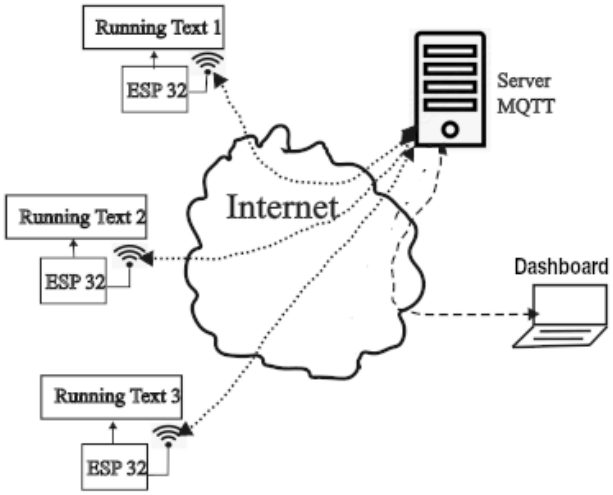
Gambar. 1. Metodologi

2.1 Perancangan Sistem

Diagram sistem secara keseluruhan diperlihatkan pada Gambar 2. Sistem terdiri atas tiga buah *Node Running Text*. Jumlah *Node Running Text* dapat ditambah sesuai kebutuhan, yang dalam rancangan ini dapat mencapai jumlah 999 buah. Koneksi internet melalui Wifi sehingga membutuhkan koneksi kabel, jaringan data seluler atau jaringan satelit agar tersambung ke jaringan internet . Pada arsitektur ini *Node Running Text* menjadi *client MQTT*. Server *MQTT* berfungsi sebagai *broker* yang meneruskan informasi dari satu *client* ke *client* yang lainnya. *Dashboard* menggunakan aplikasi *Desktop* berfungsi untuk mengubah informasi yang akan ditampilkan pada *Node Running Text*.

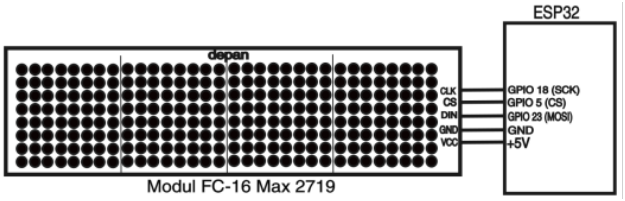
2.2 Perancangan Perangkat Keras *Node Running Text*

Dot matriks menggunakan modul basis IC MAX2719 dan System On Chip (SoC) ESP32. Modul *Dotmatix* menggunakan tipe FC-16 dengan jumlah LED sebanyak 8x32. Modul ini dapat diekspansi baik baris maupun kolomnya sehingga dimensinya akan lebih besar. SoC ESP32 digunakan untuk mengendalikan modul FC-16, koneksi WIfi, dan komunikasi dengan node pengendali. Komunikasi antara *node* pengendali dengan



Gambar. 2. Diagram sistem *IoT* pengubah informasi *Node Running Text*

Running Text menggunakan protokol MQTT.

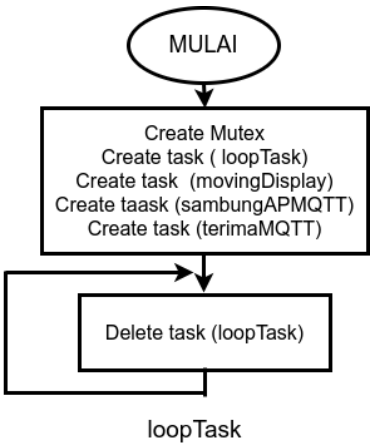


Gambar. 3. Diagram Blok *Hardware Node Running Text*

2.3 Perancangan Perangkat Lunak *Node Running Text*

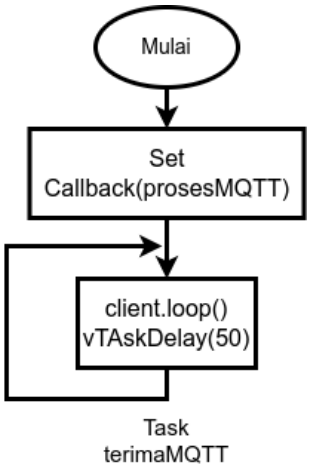
Perangkat lunak *Node Running Text* diimplementasikan menggunakan sistem operasi *Realtime FreeRTOS* pada *framework Arduino*. Perangkat lunak *Node Running Text* terdiri atas lima buah *task*, yaitu loopTask, UpdateDisplay, terimaMQTT, sambungAPMQTT, dan runningText. Diagam alir masing-masing task diperlihatkan pada Gambar 4, 5, 6, 8, dan 7.

Task loopTask adalah *task* yang berisi fungsi setup() dan loop() pada *framework Arduino* yang berfungsi membuat *task* lainnya. loopTask hanya dieksekusi sekali dengan prioritas 1. Diagram alir *Task* loopTask diperlihatkan pada Gambar 4.



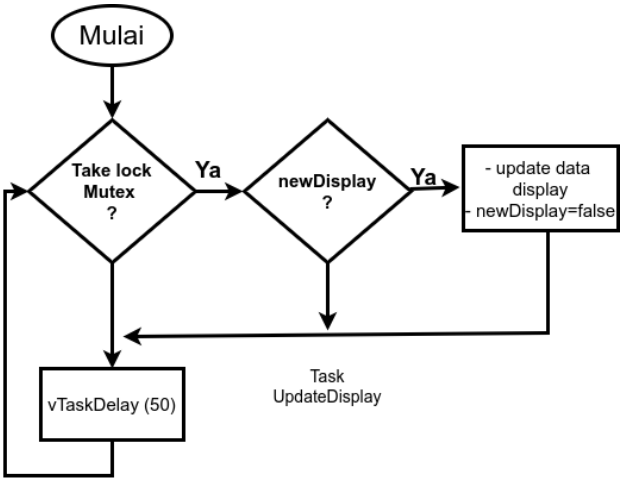
Gambar. 4. Diagram Alir *Task* LoopTask

Task TerimaMQTT berfungsi untuk menerima data dari *broker MQTT*. *Task* TerimaMQTT bersifat periodik dengan prioritas 4. Diagram alir *task* TerimaMQTT diperlihatkan pada Gambar 5.



Gambar. 5. Diagram Alir *Task* TerimaMQTT

Task UpdateDisplay berfungsi untuk mengganti data pada variabel informasi yang ditampilkan pada *Running Text*. Karena pengubahan data harus dilakukan pada saat penampil menyelesaikan satu siklus penampilan maka *task* ini menggunakan fungsi *lock* untuk mensikronkan dengan *task RunningText*. Diagram alir *task UpdateDisplay* diperlihatkan pada Gambar 6.

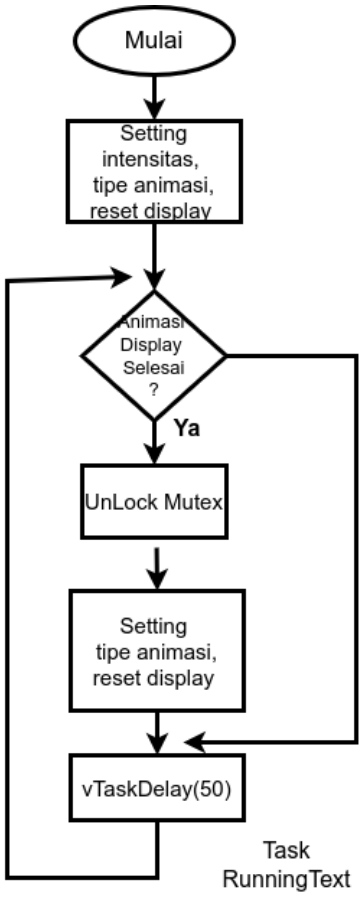


Gambar. 6. Diagram Alir *Task* UpdateDisplay

Task RunningText berfungsi untuk mengendalikan animasi pada *dot matrix*. Agar animasi tampilan dapat diselesaikan satu siklus maka *task RunningText* disinkronkan dengan *task UpdateDisplay*. *Task* ini bersifat periodik dengan prioritas 2. Diagram alir *task RunningText* diperlihatkan pada Gambar 7.

Task SambungAP-MQTT berfungsi untuk menjaga koneksi ke jaringan internet dan koneksi ke *broker MQTT*. *Task* ini juga bersifat periodik dengan prioritas 4. Diagram alir *task* SambungAP-MQTT diperlihatkan pada Gambar 8.

Prioritas untuk kelima task tersebut diperlihatkan pada Tabel **??**. Nilai yang besar menandakan prioritasnya tinggi. Berdasarkan Tabel 1. *task* sambungAP-MQTT dan terimaMQTT mempunyai prioritas paling tinggi. Dua task atau lebih yang mempunyai prioritas sama akan menggunakan algoritma *Time Slicing* yaitu task akan dieksekusi bergantian dengan waktu eksekusi yang sama dengan *Interruption Tick* [14]. Pemilihan prioritas yang sama pada *task* sambungAP-MQTT dan terimaMQTT dimaksudkan agar kedua *task* dapat dieksekusi tanpa terblokir karena prioritasnya lebih rendah. *Task* sambungAP-MQTT dan terimaMQTT dilakukan setiap 50mdetik sehingga sebagian besar waktunya digunakan untuk IDLE. Oleh karena itu task RunningText dan *updateDisplay* diberi prioritas lebih rendah. Walaupun ESP32 mempunyai dual core namun kelima *task* tersebut dijalankan pada satu core untuk mengurangi waktu beralih dari *context* satu ke *context* dan mengurangi *jitter*[15].



Gambar. 7. Diagram Alir *Task* RunningText

Tabel. 1. Prioritas, Periode dan Penggunaan Core CPU

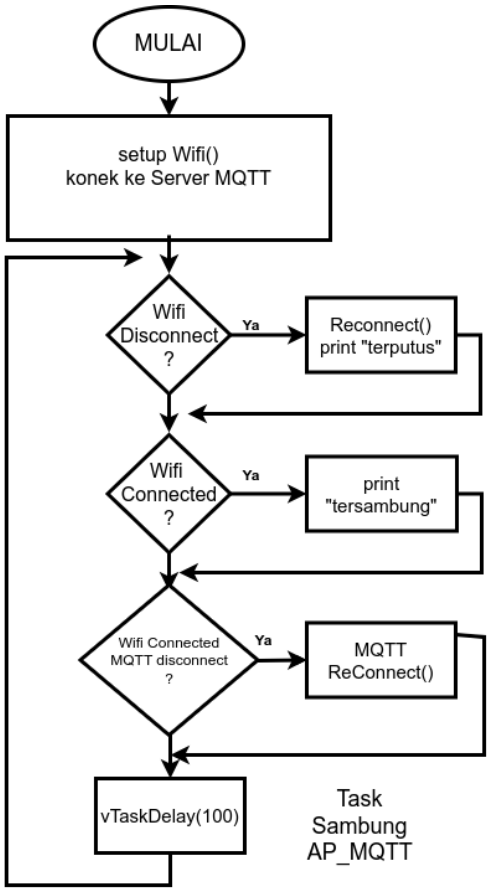
No	Nama Task	Prioritas	Periode(ms)	No.CPU
1	loopTask	1	sekali eksekusi	1
2	runningText	2	50	1
3	updateDisplay	3	50	1
4	sambungAPMQTT	4	100	1
5	terimaMQTT	4	50	1

Gambar 9 memperlihatkan rancangan eksekusi *task* berdasarkan prioritas dan periodenya. Pada Siklus 1 *task loopTask* dibuat dan dieksekusi dengan prioritas 1 dan tidak berulang. LoopTask adalah *task* yang digunakan untuk membuat *task* yang lainnya. *Task* dibuat dimulai dari *task* yang mempunyai prioritas rendah ke prioritas tinggi. *Task RunningText* dibuat dan dieksekusi pertama kali sehingga *Node Running Text* sudah dapat menampilkan animasi teksnya walaupun tidak terkoneksi ke jaringan sama sekali. Dalam kondisi waktu eksekusi *task* tidak melebihi dari periode *task* dengan prioritas lebih tinggi maka eksekusi *task* akan sesuai dengan diagram Gambar 9.

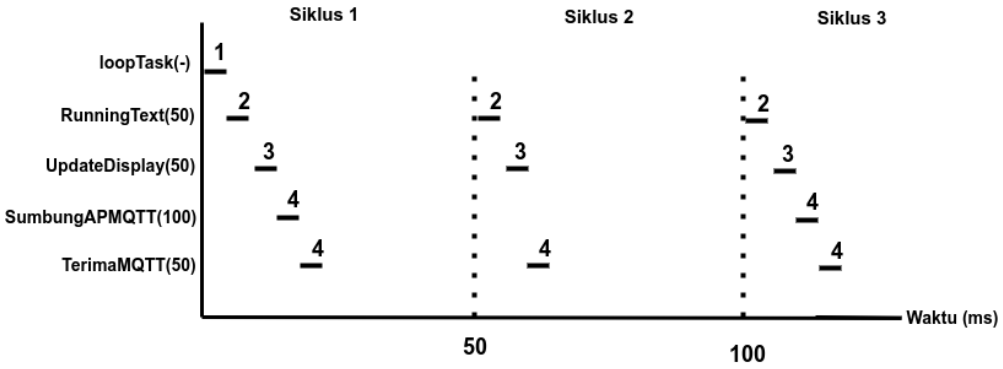
Untuk pengubahan informasi pada *Node Running Text* dilakukan melalui aplikasi *Desktop* yang dibangun menggunakan bahasa pemrograman *Python* dengan pustaka fungsi *TkInter*. Rancangan antarmuka diperlihatkan pada Gambar 10. *Input Text* digunakan untuk memasukkan informasi yang akan dikirimkan ke *Node Running Text* sesuai nomor yang dimasukkan pada masukan Nomor *Device*. Jika pengiriman data informasi berhasil maka akan ditampilkan respon berupa Nomor *Device*.

3. Hasil

Implementasi di kode sumber *task* dibuat dari *task* yang mempunyai prioritas rendah ke *task* prioritas tinggi sebagaimana diperlihatkan pada Gambar 9. Hal ini dilakukan agar *task* yang mempunyai prioritas



Gambar. 8. Diagram Alir *Task* SambungAP-MQTT



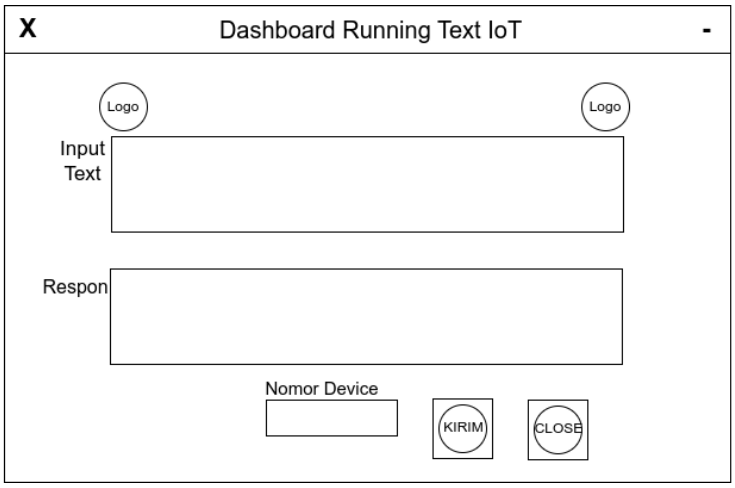
Gambar. 9. Diagram Eksekusi *Task*

rendah dapat dieksekusi terlebih dahulu. Hasil eksekusi *task* diperlihatkan pada Gambar 11. Hasil ini hanya memperlihatkan urutan eksekusi *task* dan tidak memperlihatkan waktu eksekusi *task*.

Selain urutan eksekusi *task* waktu eksekusi *task* juga diukur. Pengukuran dilakukan dengan menyisipkan instruksi pewaktuan dengan membaca waktu pada saat awal pernyataan yang akan diukur dan membaca waktu pada akhir pernyataan. Selisih antara waktu akhir dan waktu awal adalah waktu eksekusi *task*. Hasil pengukuran waktu eksekusi *task* periodik diperlihatkan pada Gambar 11.

Untuk melihat aliran data dari aplikasi *Desktop* ke server MQTT dan sebaliknya digunakan *tools Wireshark*. Hasil *capture* diperlihatkan pada Gambar 12.

Untuk mengubah informasi *running text* dibuat *Dashboard* klien MQTT yang diimplementasikan menggunakan bahasa *Python* dan pustaka *Paho MQTT*. Fungsional |*dashboard* masih hanya untuk mengirimkan teks yang akan ditampikan pada *running text* tertentu dengan alamat tertentu atau ke semua *running text* dengan alamat "000". Alamat *running text* terdiri atas tiga angka Desimal sehingga jumlah *running text*

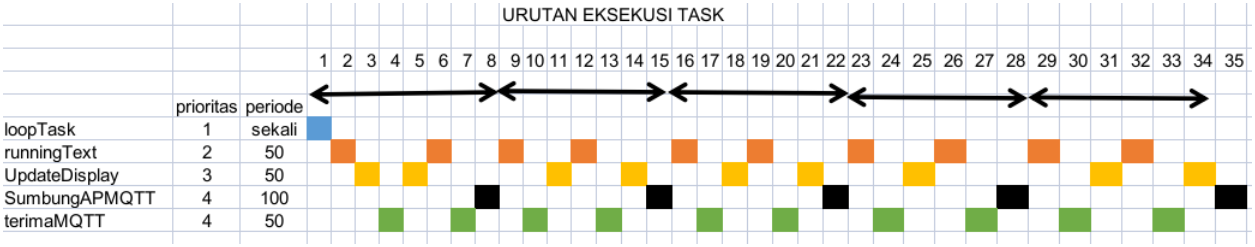


Gambar. 10. Rancangan Antarmuka Aplikasi

yang dapat terkoneksi sebanyak 999 buah.

Tabel 3 adalah hasil pengujian waktu kirim informasi dari aplikasi *Desktop* ke *node Running text* sampai diterima informasi ACK dengan berbagai jumlah karakter yang dikirim. Rata-rata waktu kirim sebesar 0,406 detik.

Prototip *Node Running Text* dan peralatan uji yang berupa *smartphone* untuk koneksi internet melalui jaringan WiFi dan telepon seluler, laptop untuk menjalankan aplikasi *Desktop*, dan catu daya untuk memberikan daya ke *Running Text*.



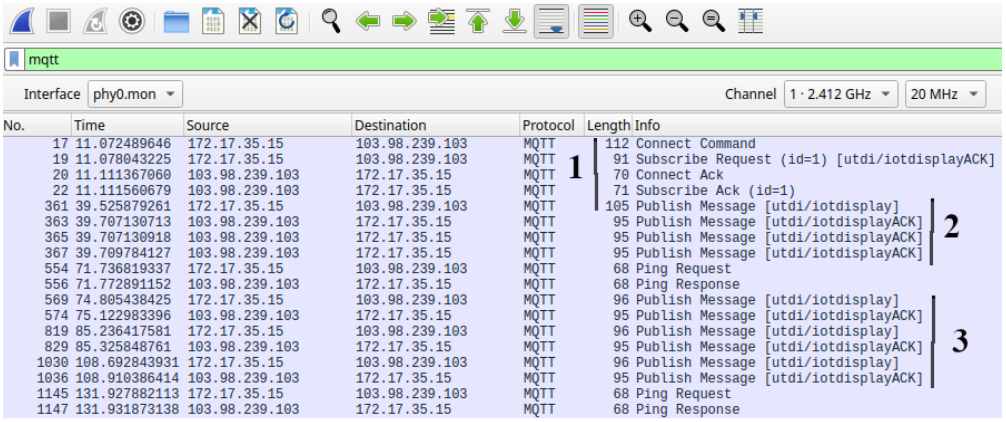
Gambar. 11. Plotting Eksekusi Task

Tabel. 2. Hasil Pengukuran Waktu Eksekusi Task Periodik

No	Nama Task	Periode (us)	Waktu Eksekusi Min (us)	Waktu Eksekusi Maks(us)
1	runningText	50.000	8	345
2	updateDisplay	50.000	22	67
3	sambungAPMQTT	100.000	35	453
4	terimaMQTT	50.000	56	1851

4. Pembahasan

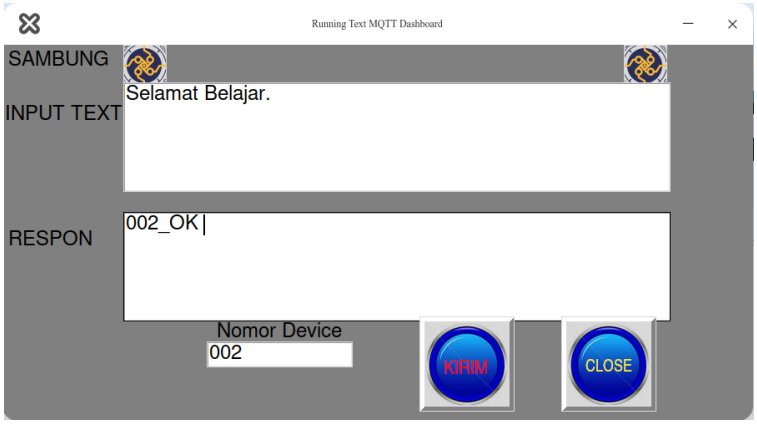
Running Text dapat diimplementasikan menggunakan *multitasking*. Urutan pembuatan dan eksekusi *task* dibuat dari *task* dengan prioritas rendah. Penjadwalan pada *FreeRTOS* menjamin bahwa *task* prioritas tinggi yang dapat dijalankan akan dipilih untuk dijalankan. Sedangkan *task* dengan prioritas yang sama akan dijalankan bergantian[14]. Pemilihan prioritas didasarkan pada periode eksekusi **task**, dan potensi *task* masuk pada kondisi *blocked*. *Task* akan masuk pada kondisi *Blocked* pada dua kondisi yaitu menunggu waktu dan menunggu kejadian atau dua-duanya[14]. *Task runningText* diberi prioritas 2 (prioritas rendah) karena kondisi *Blocked* hanya menunggu waktu saja. Sedangkan untuk *task* yang lainnya diberi prioritas yang lebih tinggi karena menangani update data dan penerimaan data dari peralatan eksternal. Hasil eksekusi pada Gambar 11 *task* memperlihatkan bahawa *task* dieksekusi berdasarkan prioritasnya selama



Gambar. 12. *Capture* Aliran Data Dari dan Ke Aplikasi *Desktop* Menggunakan Wireshark

Tabel. 3. Waktu Pengiriman Informasi dan Penerimaan ACK

No	Jumlah Karakter	Waktu (detik)
1	10	0.274610043
2	20	0.44907999
3	30	0.578629971
4	40	0.197710037
5	50	0.395659924
6	60	0.566109896
7	70	0.296550035
8	80	0.332230091
9	90	0.368469954
10	100	0.415649891
11	200	0.634940147
12	400	0.36461997
	Rata-rata=	**0.406188329**

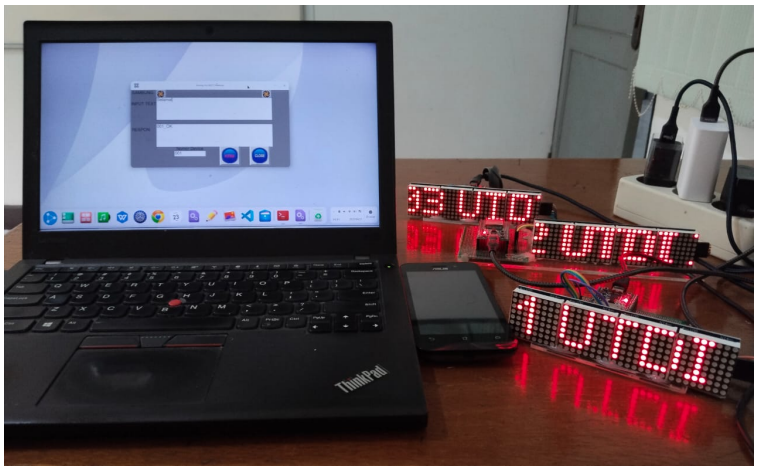


Gambar. 13. Aplikasi Desktop Untuk Dashboard Update Running Text

task dengan prioritas yang tinggi tidak masuk pada kondisi *Blocked*.

Berdasarkan pengukuran eksekusi *task* sebagaimana pada Tabel2 waktu eksekusi jauh lebih cepat dibandingkan dengan periodenya, oleh karena itu semua *task* yang mempunyai prioritas rendah dapat dieksekusi tanpa melewati periodenya.

Gambar 12 memperlihatkan *capture* aliran data dari klien aplikasi *Desktop* ke server MQTT atau sebaliknya. Nomor satu merupakan proses koneksi dan *subcribe* ke server MQTT. Nomor dua merupakan aliran data *publish* dari aplikasi *Desktop* untuk mengubah informasi semua *Running Text*, dengan menggunakan ala-



Gambar. 14. Tiga Buah Prototip Running Text

mat "000" pada topik "utdi/iotdisplay". Jika informasi tersebut diterima oleh klien *Running Text* maka akan dibalas dengan mengirimkan alamat *Running Text* dengan mempublish menggunakan topik "utdi/iotdisplay/ACK". Pada nomor tiga memperlihatkan pengiriman informasi ke masing-masing *Running Text*. Pada penelitian ini hanya diimplementasikan tiga buah *Running Text*.

Waktu kirim informasi dari aplikasi *Desktop* ke *node Running Text* berdasarkan Tabel 3 varisi jumlah karakter yang dikirim tidak mempengaruhi waktu kirim. Berarti kondisi lalulintas di jaringan yang banyak mempengaruhi waktu kirim. Waktu kirim rata-rata sebesar 0,406 detik.

Gambar 13 memperlihatkan aplikasi *Desktop* untuk *Dashboard update* informasi yang akan ditampilkan pada *Running Text*. Diiplementasikan menggunakan bahasa pemrograman *Python* menggunakan pustaka GUI TkInter dan pustaka *PAHO MQTT client*. Gambar 13 memperlihakan aplikasi mengirim ke alamat *Running Text* "002" dengan informasi "Selamat Belajar" dan pengirimana berhasil dengan ditandai respon "002-OK".

Gambar 14 memperlihatkan hasil prototipe tiga buah *Running Text* dengan alamat "001", "002", dan "003" yang masing-masing menampilkan informasi yang berbeda dan aplikasi *Desktop* untuk pengujian sistem.

5. Simpulan

Penelitian ini berhasil merancang dan mengimplementasikan *Running Text* dalam bentuk prototipe yang tampilan informasinya dapat diubah dari jarak jauh menggunakan infrasturktur internet menggnakan protokol MQTT. Program pada *Running Text* menggunakan *multitasking* pada sistem operasi *FreeRTOS* dan semua *task* dapat dieksekusi tanpa melewati periodenya. Waktu kirim informasi tidak dipengaruhi jumlah karakter yang dikirim dengan rata-rata waktu kirim sebesar 0,406 detik.

Ucapan Terima kasih

Kami ucapkan banyak terima kasih kepada Bagian Infrastruktur, Kemanan dan Inovasi Sistem Universitas Teknologi Digital Indonesia berkat dukungan infrastruktur IoT khususnya penggunaan server MQTT dan internet.

Pustaka

[1] M. Pardede, E. Hutajulu, R. Sirait, J. Junaidi, and A. Pakpahan, "Implementation of multi-hop lora network for centralized remote display of running text message based on IoT," *Jurnal Mantik*, vol. 7, no. 2, pp. 1585–1596, Aug. 2023. [Online]. Available: https://iocscience.org/ejournal/index.php/mantik/article/view/4117/2943

[2] Wijaya, A. Tafrikhatin, R. Witadi, and J. Sumarah, "Sistem Pengatur Running Text Menggunakan Bluetooth Dengan Interface Android Berbasis Arduino," *JASATEC : Journal of Students of*

Automotive, Electronic and Computer, vol. 3, no. 1, pp. 7–14, Sep. 2023. [Online]. Available: https://www.jurnal.politeknik-kebumen.ac.id/jasatec/article/view/1397/629

[3] N. Triwahyuni and S. Beta, "Running Text Information System Design Internet-Based for Small Outlets," *JAICT*, vol. 7, no. 2, p. 97, Oct. 2022. [Online]. Available: https://jurnal.polines.ac.id/index.php/jaict/article/view/3270/pdf

[4] A. W. A. Antu, S. Abdussamad, and I. Z. Nasibu, "Rancang Bangun Running Text pada Dot Matrix 16X160 Berbasis Arduino Uno Dengan Update Data System Menggunakan Perangkat Android Via Bluetooth," *Jambura Journal of Electrical and Electronics Engineering*, vol. 2, no. 1, pp. 8–13, Jan. 2020. [Online]. Available: http://ejurnal.ung.ac.id/index.php/jjeee/article/view/4321

[5] E. Ekta, V. Kumar, and S. Sandeep, "Design and Implementation of IoT based Smart Controlling Application for LED Scrolling Text Display with Integration of Google Assistant," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 8, pp. 2647–2652, Aug. 2023. [Online]. Available: https://www.irjet.net/archives/V7/i8/IRJET-V7I8443.pdf

[6] F. Putrawansyah, "Application Running Text Information Berbasis Android," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 6, no. 1, pp. 116–125, Sep. 2019. [Online]. Available: https://jurnal.mdp.ac.id/index.php/jatisi/article/view/161/104

[7] I. U. Simanjuntak and A. Suhendar, "Rancang Bangun Running Text P10 16x32 Berbasis Arduino UNO Dengan Komunikasi SMS (Short Message Service)," *Jurnal Ilmiah Teknologi Infomasi Terapan*, vol. 4, no. 2, Apr. 2018. [Online]. Available: http://journal.widyatama.ac.id/index.php/jitter/article/view/157

[8] T. Budioko, "Node Sensor Pada Sistem Monitoring Tinggi Permukaan Air Sungai Berbasis FreeRTOS dan MQTT," *JuTI "Jurnal Teknologi Informasi"*, vol. 1, no. 1, p. 36, Aug. 2022. [Online]. Available: https://ejournal.akakom.ac.id/index.php/JuTI/article/view/643

[9] Nur Afiyat, Raizly Helmi Navilla, and Mohamad Hariyadi, "Sistem Monitoring Cairan Infus Berbasis IoT Menggunakan Protokol MQTT," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 12, no. 1, pp. 50–55, Feb. 2023. [Online]. Available: https://jurnal.ugm.ac.id/v3/JNTETI/article/view/5862

[10] A. R. Thayyib, I. Salamah, and H. R.A., "IoT Based Monitoring System Using MQTT Protocol on Tortilla Chips Cutting Machine," *SISTEMASI*, vol. 12, no. 3, p. 973, Sep. 2023. [Online]. Available: https://sistemasi.ftik.unisi.ac.id/index.php/stmsi/article/view/3328/586

[11] B. Septian and F. Arkan, "FreeRTOS Based Air Quality Monitoring System Using Secure Internet Of Things," *Jurnal Teknik Informatika (JUTIF)*, vol. 3, no. 1, p. 7, 2022. [Online]. Available: http://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/172

[12] E. D. Kusuma, P. Nugroho, and W. Dewanto, "Perancangan Piranti Perekam Isyarat Bioelektrik Portable berbasis ESP32 dan FreeRTOS," *Prosiding Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika (SNESTIK)*, vol. 1, no. 1, pp. 149–155, Apr. 2022, number: 1. [Online]. Available: http://ejurnal.itats.ac.id/snestik/article/view/2705

[13] S. Cope, *MQTT for complete beginners: learn the basic of the MQTT protocol.* Miejsce nieznane: wydawca nieznany, 2020, oCLC: 1424164340.

[14] R. Barry and FreeRTOS Team, *Mastering the FreeRTOS™ Real Time Kernel A Hands-On Tutorial Guide*, 1st ed. Amazon.com, Inc, 2023.

[15] J. Arm, O. Baštán, O. Mihálik, and Z. Bradáč, "Measuring the Performance of FreeRTOS on ESP32 Multi-Core," *IFAC-PapersOnLine*, vol. 55, no. 4, pp. 292–297, 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896322003639