ANALISIS PENGGUNAAN FREERTOS PADA KONSEP MULTITASKING BERBASIS SINGLE CORE

Muhammad Akbar [1], Sabaruddin[2], Guntur[3], Matalangi[4], Andi Edeth Fuari A.[5], Muhammad Risal[6], Adi Sadli[7], Syamsu Alam[8]
[1-2] Universitas Tadulako,[3-7] Universitas Handayani Makassar
[1]akbar.stmikhdy@gmail.com, [2]sabaruddinsaputra15@gmail.com, [3]guntur@handayani.ac.id,
[4]matalangi@handayani.ac.id, [5]edetanatasya@handayani.ac.id, [6]risal@handayani.ac.id, [7]adisadli@gmail.com,
[8]syamsualam@handayani.ac.id

ABSTRAK

Banyak dari mikrokontroller yang beredar di pasaran hanya memiliki teknologi single core di dalamnya, yang tentu saja akan menjadi penghambat untuk dapat mengelola beberapa task sekaligus dalam satu waktu. Tujuan dari penelitian ini adalah melakukan analisa perbandingan antara penggunaan konsep Sistem Waktu Nyata FreeRTOS dan tanpa FreeRTOS pada single core mikrokontroller Arduino untuk mengeksekusi 3 thread berbeda, dimana akan dilakukan pengujian pada parameter waktu eksekusi, jumlah task yang dieksekusi serta konsistensi pembacaan nilai pada sensor. Adapun perangkat keras yang digunakan yakni mikrokontroller Arduino Uno, keluaran akan menggunakan diode LED dan nilai pada serial monitor. 3 thread berbeda yakni 1. Blink LED 2. Pembacaan sensor Cahaya 3. Counting angka, yang akan dieksekusi pada sketch menggunakan FreeRTOS dan sketch biasa. Hasil menunjukkan bahwa sketch dengan FreeRTOS mampu mengeksekusi 3 thread sekaligus dimana dengan total delay pada program sebesar 1100ms, dalam 1 menit mampu mengeksekusi 437 task dan 872 task dalam 2 menit. Sedangkan pada sketch tanpa FreeRTOS dalam 1 menit hanya mampu mengeksekusi 122 task dan 242 task pada waktu 2 menit dan hanya mampu mengeksekusi 1 thread dalam 1 waktu, sehingga pada thread counting angka, eksekusi akan dilakukan terus menerus tanpa looping ke thread pertama dan kedua.

Kata kunci — Sistem Waktu Nyata, Multitasking, FreeRTOS, Arduino.

1. PENDAHULUAN

Dalam era perkembangan teknologi yang pesat, konsep multitasking telah menjadi semakin penting dalam kehidupan sehari-hari. Dengan kemunculan berbagai perangkat pintar dan aplikasi yang semakin canggih, kemampuan untuk melakukan lebih dari satu tugas secara bersamaan telah menjadi kebutuhan yang tak terhindarkan. Perkembangan ini telah mengubah cara kerja sebuah system yang semakin kompleks.

Mikrokontroller merupakan perangkat utama di dunia IoT dimana terdapat banyak proses dan task yang akan diproses didalamnya, baik meliputi development environtment (event-based) ataupun kondisi-kondisi berbasis clock-based maupun interactive based. Akan tetapi keterbatasan eksekusi pada mikrokontroller single core yang hanya mampu mengeksekusi 1 task dalam 1 waktu menjadi penghambat pada sistem yang lebih kompleks dan memerlukan konsistensi waktu eksekusi yang real time [1].

Arduino merupakan salah satu mikrokontroller yang menggunakan teknologi single core di dalamnya. Keterbatasan spesifikasi arduino dengan implementasi realtime system dapat mempengaruhi kinerja realtime itu sendiri dalam beberapa cara. Meski demikian, Arduino menjadi pilihan yang baik untuk implementasi realtime yang lebih sederhana dan membutuhkan biaya murah. Namun untuk sistem yang lebih kompleks dapat menggunakan platform yang lebih kompleks atau dengan memanfaatkan mekanisme multitasking, salah satunya dengan menggunakan fungsi RTOS atau RealTime Operating System [2].

Real-Time Operating System (RTOS) adalah suatu sistem operasi yang digunakan untuk melayani aplikasi real-time yang memproses data saat masuk dalam waktu singkat atau dalam tenggat waktu tertentu dan sebagian

besar tanpa penundaan buffer. Sistem ini digunakan dalam sistem-sistem di mana hasil perhitungan digunakan untuk mempengaruhi suatu proses saat sedang dieksekusi. Sensor menghasilkan sinyal yang ditafsirkan oleh sistem operasi sebagai interupsi. Saat menerima interupsi, sistem operasi memanggil proses tertentu atau serangkaian proses untuk melayani interupsi. Dengan menggunakan RTOS, waktu pemrosesan diukur dalam sepersepuluh detik. Sistem ini terikat waktu dan memiliki tenggat waktu tetap. Jika tidak, Ini akan menyebabkan kegagalan sistem [3][4].

FreeRTOS merupakan sistem operasi sumber terbuka, netral terhadap cloud, dan waktu nyata yang menawarkan kernel yang cepat, dapat diandalkan, dan responsif. FreeRTOS didistribusikan secara bebas berdasarkan lisensi sumber terbuka Massachusetts Institute of Technology (MIT) dan diimplementasikan di lebih dari 40 arsitektur, menghadirkan pilihan perangkat keras yang beragam kepada developer, bersama dengan serangkaian pustaka perangkat lunak yang sudah dikemas sebelumnya [5].

FreeRTOS sendiri merupakan aplikasi yang relatif berukuran kecil. Inti dari FreeRTOS hanya terdiri 3 file kode sumber C, yang dapat dibagi menjadi tiga bagian, yaitu task, communication dan antarmuka perangkat keras [6].

Penelitian terkait penggunaan FreeRTOS pada mikrokontroller sebelumnya telah dilakukan oleh beberapa peneliti, seperti oleh Eka Nanda Sugianto, dkk, dengan judul "Implementasi Sistem Operasi Real-Time pada Arduino Nano dengan media Komunikasi NRF24L01 Untuk Pengukuran Suhu, Kelembaban, dan Intensitas Cahaya", penelitian ini bertujuan untuk mengukur 3 parameter berbeda dari 3 jenis sensor berbeda, Dimana setiap task akan diolah menggunakan Arduino Nano dengan memanfaatkan library FreeRTOS dengan metode Preemptive Priority Based-Scheduling. Hasil dari penelitian memperlihatkan bahwa ketiga task yang ada mampu dieksekusi sesuai dengan prioritas yang telah ditetapkan dan berjalan sesuai delay yang ditentukan [7].

Perbedaan dengan penelitian yang dilakukan yakni dari metode yang akan digunakan. Penelitian yang dilakukan menggunakan metode observasi Dimana akan membandingkan 3 parameter dari 3 task berbeda yang akan diuji.

Penelitian lain berjudul "Sistem Monitoring Struktur Jembatan dengan metode Real Time Operating System (RTOS)" oleh Komang Deha Abhimana Kader dan Agung Setia Budi. Penelitian ini diimplementasikan pada pengukuran kondisi struktur jembatan dimana akan mengukur 3 task berbeda, yakni beban, lendutan, dan regangan. Masukan sensor akan diolah oleh mikrokontroller Arduino Uno dan juga memanfaatkan library FreeRTOS. Hasil dari penelitian ini menunjukkan dengan FreeRTOS pembacaan data dari sensor lebih mendekati hasil yang diinginkan dari pada tanpa FreeRTOS [8].

Perbedaan dengan penelitian yang dilakukan yakni dari parameter yang akan diukur dan jenis sensor yang digunakan.

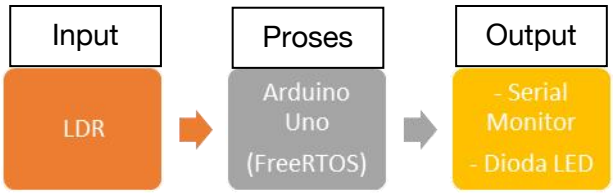
Berdasarkan hasil penelitian-penelitian terdahulu, analisis penggunaan library FreeRTOS terkait parameter utama yang berhubungan dengan sistem waktu nyata seperti kecepatan eksekusi, perbedaan waktu antara FreeRTOS dan tanpa FreeRTOS serta perubahan nilai pengukuran merupakan hal yang harus diuji dan diketahui lebih lanjut melalui penelitian

2. METODE PENELITIAN

2.1 Pembuatan Hardware (Flow Chart)

Keseluruhan penelitian dilakukan menggunakan simulasi Wokwi. Wokwi dipilih karena memudahkan dalam hal eksekusi *sketch* yang telah dibuat tanpa proses *upload* terlebih dahulu. Serta telah mendukung rancang bangun skematik rangkaian secara langsung [9].

Berdasarkan referensi diagram blok yang terkait dengan infrastruktur perangkat lunak sensor, maka rancangan flow chart dapat dibuat sebagai berikut [10] :



Gambar 1. Diagram Blok Sistem

2.1.1. Input

Input pada penelitian yang dilakukan menggunakan sensor Cahaya jenis LDR. Sensor ini digunakan untuk mengukur parameter lingkungan berupa besaran nilai Cahaya yang didapatkan. Selain itu diode led juga digunakan sebagai salah satu dari 3 task yang akan dieksekusi oleh Arduino.

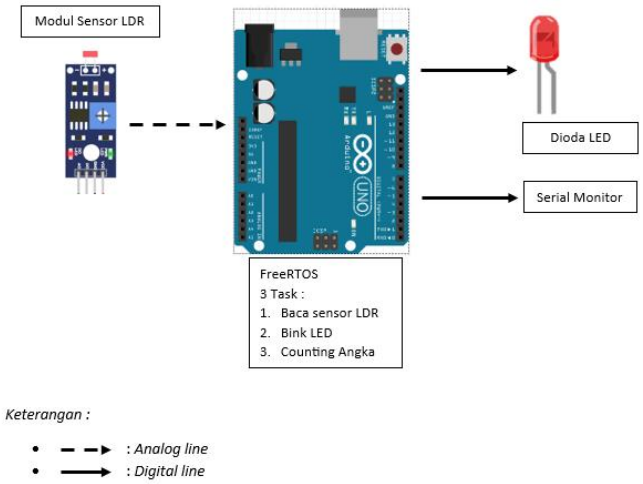
2.1.2. Proses

Untuk pengolahan data dari input dan eksekusi 3 task yang berbeda dilakukan oleh Arduino tipe Uno yang berbasis *single core*. 3 task yang ada masing-masing akan diberikan *delay* sebesar 300ms. Hasil pengolahan akan ditampilkan pada *serial monitor*. Pada Arduino Uno ini juga *library FreeRTOS* akan ditanamkan untuk mengeksekusi 3 task berbeda secara bersamaan.

2.1.3. Output

Keluaran pada rangkaian menggunakan serial monitor yang *include* pada program IDE Arduino. Serial monitor akan menampilkan kondisi ketiga task yang ada, seperti pembacaan sensor Cahaya, eksekusi *blink* led, dan *counting* angka.

2.2. Pembuatan Hardware (Arsitektur Sistem)



Gambar 2. Arsitektur Sistem

2.2.1 Modul Sensor LDR

Modul sensor LDR menjadi satu-satunya *input* yang digunakan pada sistem ini. Fungsi dari modul sensor LDR ini yakni mendeteksi *environment value* berupa Cahaya dan bekerja pada *task* 1 di dalam *sketch Arduino IDE.*

2.2.2 Arduino Uno

Arduino uno merupakan pusat dari pengolahan data pada sistem dan bertindak mengeksekusi 3 task berbeda menggunakan metode *preemptive* dengan bantuan *library FreeRTOS* dan metode *non-preemptive* tanpa *FreeRTOS.* Selain membaca nilai dari sensor LDR (*task* 1), perangkat ini juga akan mengeksekusi *blink led* (*task* 2) dan *counting* angka (*task 3*).

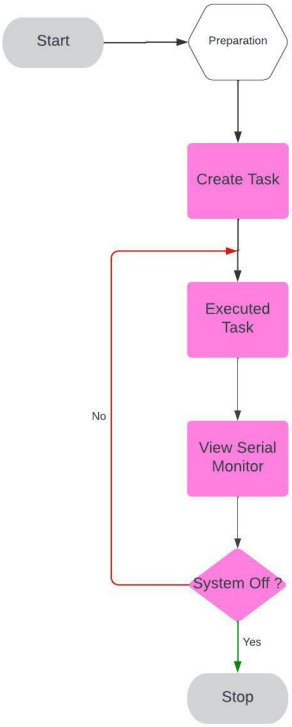
2.2.3 Dioda LED

Diode led bekerja pada *task* 2, dimana akan bernilai 1 selama 300ms. *Diode led* terhubung ke *Arduino Uno* dengan resistor 220 ohm sebagai penghubung.

2.2.4 Serial monitor

Serial monitor pada sistem berfungsi sebagai layanan monitoring yang akan berfungsi untuk menampilkan informasi terkait pembacaan nilai sensor, kondisi led dan *counting* angka

2.3 Pembuatan Software (Flowchart Sistem)



Gambar 3. Flowchart Sistem

Proses pertama pada sistem berdasarkan gambar 8 di atas yakni proses *preparation*. *Preparation* akan melakukan pemberian variabel serta inisialisasi setiap perangkat yang terhubung ke dalam sistem *hardware*, baik *input* maupun *output*.
Proses selanjutnya yaitu *Create Task*. Proses ini akan mengeksekusi 3 *task* yang berbeda di masing-masing *void* pada *sketch Arduino IDE*. 3 *task* tersebut yakni pembacaan nilai sensor yang dihasilkan oleh modul sensor *LDR*, *blink LED* dan *counting number*. Setelah task di eksekusi, kemudian hasilnya akan ditampilkan pada serial monitor dan secara fisik dapat dilihat pada indikator *dioda LED* untuk status proses *blink LED*. Terakhir sistem akan melakukan loop hingga sistem dimatikan.

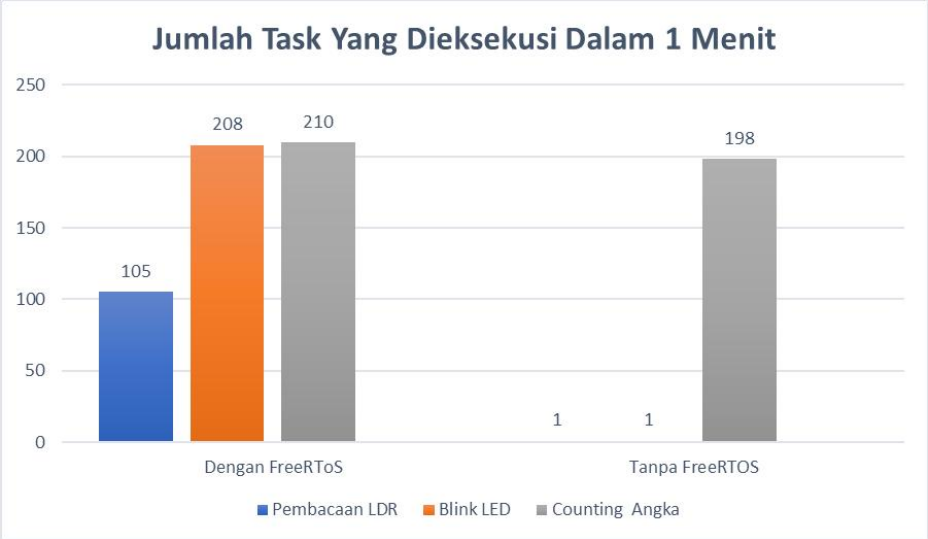
3 HASIL DAN PEMBAHASAN

3.1 Pengujian Jumlah Task Yang Dieksekusi

Tabel 1 di bawah ini menunjukkan jumlah task yang dieksekusi dengan menggunakan sketch *FreeRTOS* dan sketch tanpa *FreeRTOS*. Pengujian dilakukan dengan membandingkan jumlah task yang dieksekusi untuk masing-masing task pada periode waktu 1 dan 2 menit.
Dari hasil pengujian didapatkan bahwa hasil eksekusi tanpa *FreeRTOS* bersifat monotasking atau hanya akan mengeksekusi 1 task dalam 1 waktu. Terlihat pada tabel 1 di bawah ini, jumlah task yang dieksekusi untuk task pembacaan sensor LDR dan Blink LED hanya berjumlah 1 task, hal ini dikarenakan *sketch* yang dijalankan akan terus mengcounting angka pada task ketiga sehingga proses *looping* tidak akan terjadi pada *sketch* tanpa *FreeRTOS* ini.
Pada sketch dengan *FreeRTOS*, proses eksekusi dilakukan dengan konsep *multitasking,* dimana *3* task yang ada akan dieksekusi secara bersamaan. Terlihat pada hasil tabel 1, 3 task yang ada dieksekusi secara bersamaan jumlah *task* yang dieksekusi sebanyak 105, 208 dan 210 untuk pembacaan sensor *LDR*, *blink LED* dan *counting* angka dengan total task eksekusi sebanyak 523 *task* dalam waktu 1 menit.
Pada waktu eksekusi 2 menit *sketch* tanpa *FreeRTOS* dapat mengeksekusi *task* dengan total 400 *task*. Sedangkan pada sketch dengan *FreeRTOS* mampu mengeksekusi *task* dengan total 1043 task. Perbedaan jumlah *task* yang di eksekusi antara *sketch* dengan dan tanpa *FreeRTOS* adalah sebesar 38.24% untuk 1 menit dan 38.35% pada waktu eksekusi 2 menit.

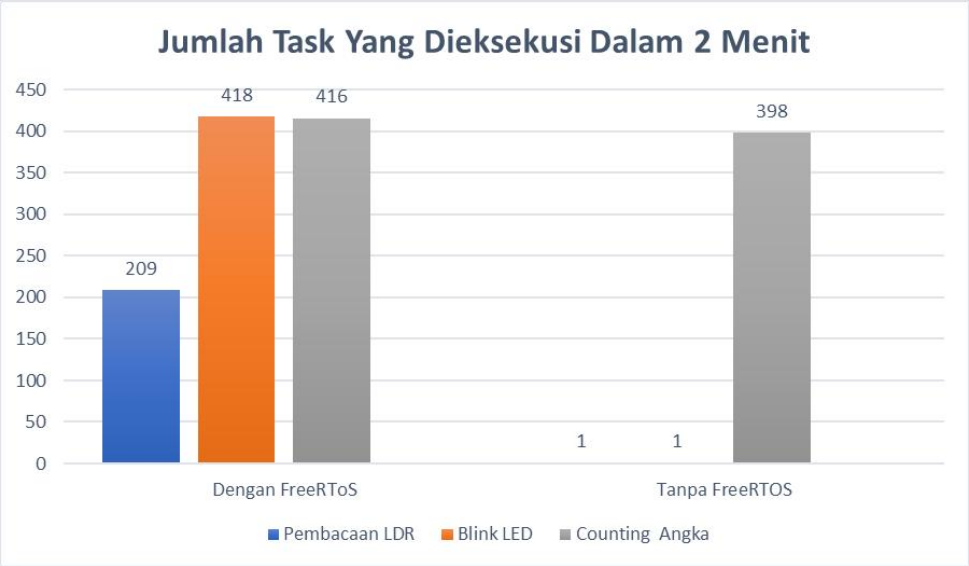
Tabel 1. Pengujian jumlah total task yang dieksekusi

No	Task	Waktu Eksekusi	Tanpa *FreeRTOS*	Dengan *FreeRTOS*
1	Pembacaan Sensor LDR	1 menit	1	105
2	Blink LED	1 menit	1	208
3	Counting Angka	1 menit	198	210
4	**Total Task**	**1 menit**	**200**	**523**
5	Pembacaan Sensor LDR	2 menit	1	209
6	Blink LED	2 menit	1	418
7	Counting Angka	2 menit	398	416
8	**Total Task**	**2 menit**	**400**	**1043**



Gambar 4. Grafik jumlah task yang dieksekusi dalam 1 menit

Gambar 3 memperlihatkan grafik jumlahh task yang dieksekusi dalam waktu 1 menit. Seperti dijelaskan sebelumnya bahwa sketch tanpa FreeRTOS hanya akan mengeksekusi sekali task pembacaan LDR dan blinkled, dikarenkan mekanisme multitasking tidak digunakan pada sketch tersebut, sehingga hanya pada counting angka proses akan terus terulang hingga sistem OFF. Berbeda dengan sketch dengan FreeRTos, 3 task yang ada akan dieksekusi secara bergantian selama sistem ON.



Gambar 5. Grafik jumlah task yang dieksekusi dalam 2 menit

Gambar 4 memperlihatkan jumlah task yang dieksekusi dalam 2 menit. Pada sketch tanpa FreeRTOS, task pembacaan LDR dan blink tetap dieksekusi sekali, yakni pada awal proses, sisanya akan mengeksekusi counting angka hingga sistem OFF. Pada sketch dengan FreeRTOS, sistem akan terus mengeksekusi ketiga task yang ada menggunakan konsep multitasking.

3.2 Pengujian Konsistensi Pembacaan Nilai Sensor LDR

Tabel 2 di bawah ini bertujuan untuk mengetahui Tingkat pembacaan nilai sensor yang dilakukan oleh modul sensor LDR, dimana akan dibandingkan 2 sketch dengan dan tanpa *FreeRTOS*. Pengukuran dilakukan dengan mengubah nilai luminens modul sensor LDR, yang mana pada simulator Wokwi hal ini dapat dilakukan dengan cara manual. Nilai luminens pada uji coba yang dilakukan bervarisasi mulai dari 10 hingga 5012 lux.

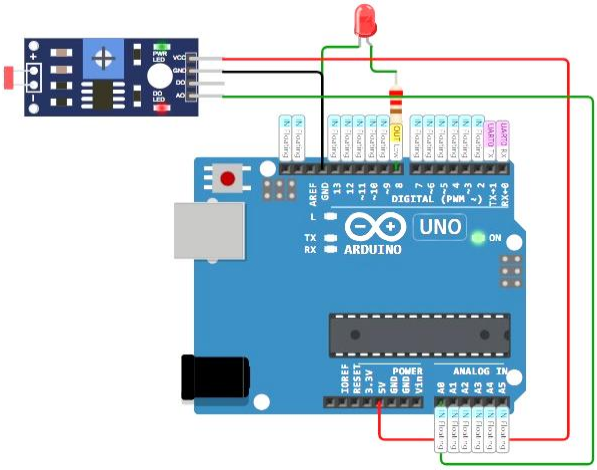
Secara keseluruhan pembacaan nilai sensor yang ditampilkan pada serial monitor (*analog value*) secara konsisten sama antara 2 *sketch* tersebut, kecuali hanya pada nilai 10 lux, terlihat bahwa nilai analog value pada serial monitor tanpa *FreeRTOS* didapatkan bernilai 853, sedangkan pada sketch dengan *FreeRTOS* bernilai 848. Sehingga dari 10 kali percobaan yang dilakukan didapatkan 90% tingkat kesamaan pembacaan *analog value* pada *serial monitor,* dan *margin error* pada nilai *analog value* yang berbeda didapatkan sebesar 0.59%.

Tabel 2. Pengujian *Konsistensi Pembacaan Nilai Sensor LDR*

No	Luminens	Pembacaan Sensor LDR Serial Monitor (*Analog Valuie*)	
		Tanpa *FreeRTOS*	Dengan *FreeRTOS*
1	10	853	848
2	100	511	511
3	501	249	249
4	1000	169	169
5	1514	132	132
6	2089	108	108
7	2512	96	96
8	3020	86	86
9	4169	69	69
10	5012	61	61

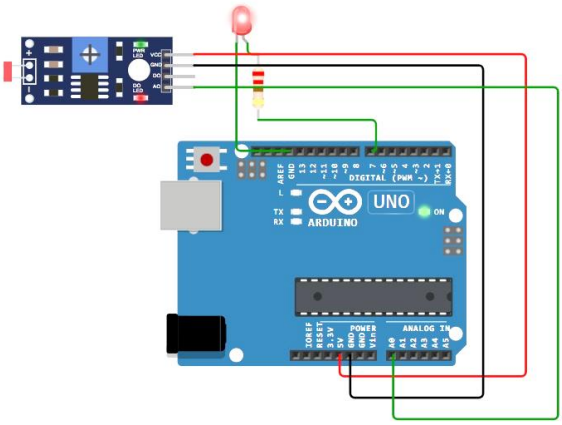
4 *Pengujan fisikal rangkaian untuk Blink LED*

Pengujian selanjutnya yakni keluaran indikator dioda LED dalam merespon 3 task yang akan dieksekusi. pengujian ini bertujuan untuk mengetahui perbedaan keluaran yang dihasilkan oleh sketch dengan dan tanpa *library FreeRTOS.*



Gambar 6. Tampilan indikator *dioda LED* tanpa *FreeRTOS*

Gambar 5 menunjukkan kondisi rangkaian dengan sketch tanpa *FreeRTOS*. Terlihat bahwa lampu LED yang terpasang dalam keadaan OFF, hal ini dikarenakan eksekusi ke 3 task dilakukan secara bergantian, sehingga lampu LED pada rangkaian hanya akan bernilai ON selama 300ms saja. Selanjutnya lampu LED tidak aktif Kembali yang disebabkan karena *counting* angka yang terus dieksekusi



Gambar 7. Tampilan indikator *dioda LED* dengan *FreeRTOS*

Dapat dilihat pada gambar 6 di atas, berbeda dengan sebelumnya, indikator LED pada rangkaian dengan *sketch FreeRTOS* memperlihatkan kondisi LED yang ON, hal ini dikarenakan *FreeRTOS* akan mengeksekusi 3 *task* yang berbeda secara bersamaan, sehingga kondisi lampu LED akan aktif bersamaan dengan 2 *task* lainnya.

4 KESIMPULAN DAN SARAN

4.1 Kesimpulan

Sketch dengan FreeRTOS terbukti dapat mengeksekusi 3 task yang berbeda secara bersamaan, berbeda dengan sketch tanpa FreeRTOS, 3 task akan dieksekusi secara bergantian. Task counting angka pada sketch tanpa FreeRTOS yang akan dieksekusi secara kontinu mengakibatkan proses looping tidak terjadi, sehingga 2 task awal tidak aktif. Sketch tanpa FreeRTOS mengeksekusi 200 task dalam waktu 1 menit dengan 300ms delay

disetiap task dan 400 task dalam 2 menit, sedangkan sketch dengan FreeRTOS mampu mengeksekusi 523 task dalam 1 menit dan 1043 task dalam 2 menit. Sehingga perbedaan jumlah eksekusi task adalah 38,24% untuk 1 menit dan 38.35% untuk 2 menit.

4.2 Saran

Untuk pengembangan ke depan, baiknya rangkaian tidak hanya diuji pada simulasi saja tapi juga pada rangkaian yang sebenarnya, hal ini berguna untuk proses pengujian yang lebih bervariatif dan tepat. Ke depannya pengujian dapat dilakukan menggunakan jumlah task yang lebih banyak dan dengan waktu yang lebih lama.

DAFTAR PUSTAKA

[1] P. Buonocunto, A. Biondi, and P. Lorefice, "Real-Time Multitasking in Arduino," *Proc. 9th IEEE Int. Symp. Ind. Embed. Syst. (SIES 2014)*, 2015.

[2] M. Akbar, "Realtime Database Sensor Menggunakan Arduino Uno Untuk Keperluan Sistem Informasi," *Ilk. J. Ilm.*, vol. 9, no. 1, pp. 91–95, 2017, doi: 10.33096/ilkom.v9i1.115.91-95.

[3] "Real-Time Operating System | BINUS UNIVERSITY BANDUNG - Kampus Teknologi Kreatif." https://binus.ac.id/bandung/2022/11/real-time-operating-system/ (accessed Jun. 29, 2024).

[4] Y. Putra, "Dasar Sistem Waktu Nyata - Google Books," 2020. https://www.google.co.id/books/edition/Dasar_Sistem_Waktu_Nyata/3ePgDwAAQBAJ?hl=jv&gbpv=1&dq=sistem+waktu+nyata+adalah&pg=PA1&printsec=frontcover (accessed Jun. 29, 2024).

[5] "Sistem Operasi untuk Mikrokontroler – FreeRTOS – Amazon Web Services." https://aws.amazon.com/id/freertos/ (accessed Jun. 29, 2024).

[6] "Tutorial FreeRTOS : #1 RTOS ber-'open source' Untuk Perangkat Embedded dan IoT - embeddednesia.com." https://embeddednesia.com/v1/tutorial-freertos-1-rtos-ber-open-source-untuk-perangkat-embedded-dan-iot/ (accessed Jun. 29, 2024).

[7] E. N. Sugianto, W. Kurniawan, and D. Syauqy, "Implementasi Sistem Operasi Real-Time pada Arduino Nano dengan media Komunikasi NRF24L01 Untuk Pengukuran Suhu, Kelembaban, dan Intensitas Cahaya," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 4, pp. 3589–3596, 2019, [Online]. Available: http://j-ptiik.ub.ac.id

[8] K. Deha, A. Kader, and A. Setia Budi, "Sistem Monitoring Struktur Jembatan dengan metode Real Time Operating System (RTOS)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 2, pp. 566–571, 2021, [Online]. Available: http://j-ptiik.ub.ac.id

[9] Rizki Prasetyo Tulodo, Nur Tulus Ujianto, Eko Budiraharjo, and Yustia Hapsari, "Pengembangan Sistem Pendeteksi Hujan Berbasis Internet of Things (IoT) dengan Simulasi Wokwi," *Engineering*, vol. 14, no. 2, 2023.

[10] M. Akbar and L. Affandy, "Impelementasi Sistem Pendeteksi Kebakaran Berbasis TeknologiInternet of Things," *Elektroda*, vol. 8, no. 1, pp. 283–288, 2023, Accessed: Jun. 14, 2023. [Online]. Available: https://elektroda.uho.ac.id/index.php/journal/article/view/23