IMPLEMENTASI SISTEM EMBEDDED REAL-TIME MULTITASKING MENGGUNAKAN ARDUINO BERBASIS COOPERATIVE SCHEDULING

Arief Budijanto[1]

[1]Teknologi Komputer, [1]Politeknik NSC Surabaya
[1]ariefbdj212@gmail.com

ABSTRAK

Perkembangan sistem embedded menuntut mikrokontroler mampu menjalankan beberapa tugas secara bersamaan dengan respons yang cepat dan efisien. Namun, Arduino Uno maupun Arduino Nano belum dilengkapi Real-Time Operating System (RTOS), sehingga sebagian besar aplikasi masih menggunakan fungsi delay() yang bersifat blocking dan menyebabkan penurunan responsivitas sistem. Penelitian ini bertujuan mengimplementasikan algoritma Real-Time Multitasking berbasis Cooperative Scheduler menggunakan fungsi millis() untuk menjalankan beberapa task secara simultan tanpa RTOS. Sistem yang dikembangkan terdiri atas lima task, yaitu pengendalian LED, pembacaan sensor DHT11, pembaruan tampilan OLED SSD1306, pengaktifan buzzer, dan komunikasi Serial Monitor. Pengujian dilakukan dengan membandingkan metode Cooperative Scheduler terhadap metode berbasis delay() menggunakan parameter response time, task latency, dan stabilitas sistem. Hasil penelitian menunjukkan bahwa Cooperative Scheduler mampu menjalankan seluruh task secara periodik tanpa saling menghambat, dengan rata-rata response time sebesar 2,8 ms dan task latency sebesar 1,4 ms, jauh lebih baik dibandingkan metode delay(). Selain meningkatkan responsivitas, algoritma ini juga memiliki kebutuhan memori yang rendah sehingga sesuai diterapkan pada Arduino dengan sumber daya terbatas. Penelitian ini memberikan solusi sederhana dan efektif untuk implementasi multitasking pada sistem embedded yang digunakan dalam aplikasi monitoring, otomasi, robotika, dan Internet of Things.

Kata Kunci: Arduino, Cooperative Scheduler, Embedded System, Multitasking, Real-Time System.

PENDAHULUAH

Perkembangan teknologi Embedded System telah memberikan dampak yang sangat besar terhadap berbagai bidang, seperti otomasi industri, robotika, Internet of Things (IoT), Abolhassani Khajeh, S., Saberikamarposhti, M., & Rahmani, A. M. (2022) ,sistem monitoring lingkungan, hingga perangkat kesehatan. Salah satu platform mikrokontroler yang paling banyak digunakan dalam pengembangan sistem embedded adalah Arduino karena memiliki arsitektur yang sederhana, biaya implementasi rendah, serta didukung oleh ekosistem perangkat lunak dan perangkat keras yang luas. Kemudahan pemrograman dan banyaknya pustaka (*library*) yang tersedia menjadikan Arduino sebagai platform utama dalam kegiatan pendidikan, penelitian, maupun pengembangan prototipe.

Meskipun demikian, Arduino Uno maupun Arduino Nano memiliki keterbatasan dibandingkan mikrokontroler modern yang telah mendukung sistem operasi real-time (*Real-Time Operating System* atau RTOS) Barry, R. (2010). Seluruh program pada Arduino dijalankan menggunakan satu fungsi utama, yaitu loop(), yang dieksekusi secara berulang tanpa mekanisme penjadwalan tugas secara otomatis. Akibatnya, sebagian besar pengembang masih menggunakan fungsi delay() untuk mengatur waktu eksekusi program. Pendekatan ini memang sederhana, tetapi memiliki kelemahan utama berupa sifat *blocking*, yaitu seluruh aktivitas mikrokontroler akan berhenti selama waktu penundaan berlangsung. Kondisi tersebut menyebabkan penurunan responsivitas sistem, terutama ketika aplikasi harus mengendalikan beberapa perangkat secara bersamaan.

Pada aplikasi nyata, sebuah sistem embedded umumnya harus menjalankan berbagai tugas secara simultan, misalnya membaca sensor suhu dan kelembapan, mengendalikan aktuator, memperbarui tampilan OLED, mengirimkan data ke komputer, serta mengaktifkan alarm berdasarkan kondisi tertentu. Apabila seluruh proses tersebut dijalankan menggunakan fungsi delay(), maka setiap task harus menunggu task lainnya selesai terlebih dahulu sehingga menyebabkan keterlambatan (*latency*) dan penurunan performa sistem. Masalah ini akan semakin terlihat ketika jumlah perangkat yang dikendalikan bertambah.

Salah satu pendekatan yang dapat diterapkan untuk mengatasi permasalahan tersebut adalah Cooperative Multitasking. Pada metode ini, setiap task dijalankan secara periodik berdasarkan interval waktu tertentu menggunakan fungsi millis(). Berbeda dengan pendekatan berbasis *preemptive multitasking* yang digunakan pada RTOS, Cooperative Scheduler bekerja tanpa interupsi paksa. Setiap task harus selesai dalam waktu singkat sebelum memberikan kesempatan kepada task lainnya untuk dieksekusi. Dengan demikian, seluruh tugas dapat berjalan secara bergantian dengan sangat cepat sehingga sistem tampak bekerja secara paralel meskipun sebenarnya menggunakan satu prosesor.

Beberapa penelitian sebelumnya telah membahas implementasi multitasking pada mikrokontroler menggunakan RTOS seperti FreeRTOS pada ESP32 maupun STM32. Namun, implementasi algoritma Cooperative Scheduler pada Arduino dengan analisis performa yang komprehensif masih relatif terbatas. Sebagian besar penelitian hanya menunjukkan implementasi dasar tanpa mengevaluasi parameter penting seperti waktu respons (*response time*), keterlambatan tugas (*task latency*), kestabilan eksekusi (*execution stability*), maupun efisiensi penggunaan

sumber daya mikrokontroler. Padahal, parameter-parameter tersebut sangat penting dalam menentukan kelayakan sistem embedded yang bekerja secara real-time.

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan implementasi algoritma Real-Time Multitasking berbasis Cooperative Scheduler pada platform Arduino. Buonocunto, P., Biondi, A., Pagani, M., Marinoni, M., & Buttazzo, G. (2016). Sistem dirancang untuk menjalankan lima task utama secara simultan, yaitu pengendalian LED, pembacaan sensor DHT11, pengaktifan buzzer, pembaruan tampilan OLED SSD1306, dan komunikasi serial. Penjadwalan setiap task dilakukan menggunakan fungsi millis() sehingga seluruh proses dapat berjalan tanpa menggunakan fungsi delay() maupun RTOS.

Kontribusi utama penelitian ini adalah pengembangan lightweight cooperative scheduler yang mudah diimplementasikan pada Arduino dengan kebutuhan memori yang sangat rendah. Selain itu, penelitian ini menyajikan analisis kuantitatif mengenai performa scheduler melalui pengukuran waktu respons, *task latency*, kestabilan eksekusi, dan utilisasi prosesor, kemudian membandingkannya dengan metode pemrograman konvensional berbasis delay(). Hasil penelitian diharapkan dapat menjadi referensi dalam pengembangan sistem embedded, robotika, dan Internet of Things yang membutuhkan kemampuan multitasking pada mikrokontroler dengan sumber daya terbatas.

Embedded System

Embedded System merupakan sistem komputer khusus yang dirancang untuk menjalankan fungsi tertentu secara real-time dengan mengintegrasikan perangkat keras dan perangkat lunak dalam satu kesatuan. Berbeda dengan komputer umum, embedded system dikembangkan untuk menjalankan aplikasi spesifik dengan kebutuhan sumber daya yang relatif terbatas. Sistem ini banyak digunakan pada perangkat otomasi industri, kendaraan pintar, peralatan medis, sistem monitoring lingkungan, perangkat rumah tangga pintar, dan robotika. Karakteristik utama embedded system meliputi konsumsi daya rendah, keandalan tinggi, ukuran perangkat yang kecil, serta kemampuan bekerja secara terus-menerus dalam waktu yang lama.

Arduino sebagai Platform Embedded

Arduino merupakan platform mikrokontroler bersifat *open-source* yang banyak digunakan dalam pengembangan prototipe dan sistem embedded. Arduino Uno menggunakan mikrokontroler ATmega328P dengan frekuensi kerja 16 MHz, memori Flash sebesar 32 KB, SRAM 2 KB, dan EEPROM 1 KB. Kesederhanaan arsitektur, dukungan pustaka yang lengkap, serta kemudahan pemrograman menggunakan bahasa C/C++ menjadikan Arduino sebagai platform yang sangat populer dalam bidang pendidikan maupun penelitian.

Meskipun demikian, Arduino tidak memiliki mekanisme penjadwalan tugas seperti RTOS sehingga seluruh program dieksekusi secara berurutan melalui fungsi loop(). Oleh karena itu, diperlukan strategi pemrograman yang tepat agar beberapa proses dapat berjalan secara bersamaan tanpa mengurangi responsivitas sistem.

Sistem Real-Time

Sistem real-time adalah sistem komputasi yang harus menghasilkan keluaran dalam batas waktu tertentu sesuai kebutuhan aplikasi. Ketepatan waktu merupakan faktor utama dalam sistem real-time karena keterlambatan eksekusi dapat menyebabkan kegagalan fungsi sistem secara keseluruhan. Berdasarkan karakteristiknya, Liu, J. W. (2006) sistem real-time dibedakan menjadi *hard real-time*, *firm real-time*, dan *soft real-time*. Pada penelitian ini digunakan pendekatan *soft real-time* karena aplikasi monitoring dan kendali masih dapat mentoleransi keterlambatan yang sangat kecil.

Cooperative Multitasking

Cooperative Multitasking merupakan metode penjadwalan tugas di mana setiap task secara sukarela menyerahkan kendali kepada task lainnya setelah prosesnya selesai. Tidak terdapat mekanisme interupsi paksa sebagaimana pada *preemptive multitasking*. Patil, R., Navnath, S. K., Abasaheb, P. A., Prakash, H. P., & Shankar, S. V. (2022). Pendekatan ini memiliki keunggulan berupa implementasi yang sederhana, kebutuhan memori yang rendah, dan sangat sesuai diterapkan pada mikrokontroler dengan kapasitas memori terbatas seperti Arduino Uno. Agar sistem tetap responsif, setiap task harus dirancang dengan waktu eksekusi yang singkat dan tidak menggunakan fungsi yang bersifat blocking.

Cooperative Scheduler Berbasis millis()

Fungsi millis() pada Arduino menghasilkan nilai waktu dalam satuan milidetik sejak mikrokontroler mulai dijalankan. Dengan membandingkan nilai millis() saat ini terhadap waktu eksekusi sebelumnya, setiap task dapat dijalankan sesuai interval yang telah ditentukan tanpa menghentikan proses lainnya. Ram, S. A., Siddarth, N., Manjula, N., Rogan, K., & Srinivasan, K. (2017). Pendekatan ini memungkinkan implementasi scheduler ringan (*lightweight scheduler*) yang mampu menjalankan beberapa task secara periodik dan independen. Dibandingkan penggunaan delay(), scheduler berbasis millis() memberikan respons sistem yang lebih baik, mengurangi waktu tunggu, serta meningkatkan efisiensi penggunaan prosesor.

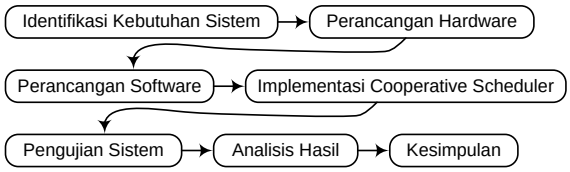
PEMBAHASAN

Metode Penelitian

Penelitian ini menggunakan metode Research and Development (R&D) yang dipadukan dengan pendekatan eksperimental (experimental research). Metode ini dipilih karena penelitian tidak hanya bertujuan menghasilkan algoritma multitasking berbasis Cooperative Scheduler, tetapi juga mengimplementasi- kan, menguji, dan mengevaluasi performanya pada sistem embedded berbasis Arduino.

Tahapan penelitian terdiri atas enam langkah utama, yaitu identifikasi kebutuhan sistem, perancangan perangkat keras, perancangan perangkat lunak, implementasi algoritma Cooperative Scheduler, pengujian sistem, dan analisis hasil pengujian. Seluruh tahapan penelitian ditunjukkan pada Gambar 1.

Pendahuluan berisi hal-hal tentang deskripsi



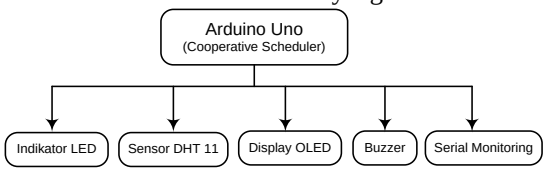
Gambar 1. Tahapan Penelitian

Tahapan penelitian dijelaskan sebagai berikut.

1. Identifikasi kebutuhan sistem dilakukan dengan menentukan jumlah task yang akan dijalankan secara bersamaan beserta interval eksekusinya.
2. Perancangan perangkat keras dilakukan dengan memilih komponen yang digunakan yaitu Arduino Uno, sensor DHT11, OLED SSD1306, LED indikator dan buzzer.
3. Perancangan perangkat lunak dilakukan menggunakan Arduino IDE dengan bahasa pemrograman C++.
4. Implementasi Cooperative Scheduler dilakukan menggunakan fungsi millis() sehingga seluruh task dapat berjalan tanpa menggunakan delay().
5. Pengujian sistem dilakukan dengan membandingkan metode Cooperative Scheduler terhadap metode konvensional berbasis delay().
6. Analisis hasil dilakukan berdasarkan parameter waktu respons (*response time*), *task latency*, kestabilan eksekusi dan penggunaan sumber daya mikrokontroler.

Perancangan Sistem

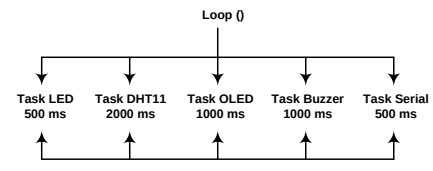
Sistem terdiri atas satu mikrokontroler Arduino yang mengendalikan lima task secara simultan menggunakan Cooperative Scheduler dapat diliht pada gambar 2. Scheduler menjadi pusat pengendali seluruh perangkat sehingga masing-masing task memperoleh waktu eksekusi sesuai interval yang telah ditentukan.



Gambar 2. Diagram Blok Sistem

Arsitektur Cooperative Scheduler

Cooperative Scheduler dirancang menggunakan lima task independen yang masing-masing memiliki timer sendiri. Setiap task hanya akan dieksekusi apabila selisih waktu antara millis() saat ini dengan waktu eksekusi sebelumnya telah mencapai interval yang ditentukan. Arsitektur Cooperative Scheduler dapat dilihat pada gambar 3.



Gambar 3. Arsitektur Cooperative Scheduler

Algoritma scheduler yang digunakan pada penelitian ini dapat dituliskan sebagai berikut. Mulai inisialisasi seluruh perangkat.

```
while(1)
  waktu ← millis()
  if(Task_LED)
      Jalankan LED
  if(Task_DHT)
      Baca Sensor
  if(Task_OLED)
      Update OLED
  if(Task_Buzzer)
      Bunyikan Buzzer
  if(Task_Serial)
      Kirim Data
ulang terus
```

Algoritma tersebut memastikan setiap task memperoleh kesempatan eksekusi secara periodik tanpa menghentikan task lainnya. Dengan demikian, sistem mampu memberikan respons yang lebih cepat dibandingkan pendekatan berbasis delay().

Skenario Pengujian

Pengujian dilakukan menggunakan dua metode yang berbeda, yaitu Cooperative Scheduler dan pemrograman berbasis delay(), dengan skenario yang identik. Parameter yang diukur meliputi waktu respons, *task latency*, stabilitas eksekusi, dan utilisasi prosesor. Hasil pengujian akan disajikan dalam bentuk tabel dan grafik pada bagian berikutnya untuk menunjukkan peningkatan performa yang diperoleh melalui implementasi Cooperative Scheduler.

Implementasi Sistem Dan Hasil Penelitian

Implementasi Sistem

Tahap implementasi dilakukan dengan merealisasikan rancangan perangkat keras dan perangkat lunak yang telah disusun pada tahap sebelumnya. Sistem menggunakan Arduino Uno R3 sebagai pusat pengendali yang mengelola lima buah task secara bersamaan menggunakan algoritma Cooperative Scheduler.

Kelima task tersebut terdiri atas:

1. Task LED indikator
2. Task pembacaan sensor DHT11
3. Task tampilan OLED SSD1306
4. Task buzzer
5. Task komunikasi Serial Monitor

Masing-masing task memiliki interval eksekusi yang berbeda sehingga scheduler harus mampu mengatur seluruh proses tanpa saling mengganggu.

Implementasi Cooperative Scheduler

Implementasi scheduler menggunakan fungsi millis() sebagai pencacah waktu.

Setiap task mempunyai dua parameter utama yaitu

- waktu terakhir dijalankan (*last execution*)
- interval eksekusi (*execution interval*)

Secara matematis scheduler dituliskan sebagai

$$CurreintTime = millis()$$

Jika $(CurrentTime - LastTime) \geq Interval$ maka task diekskusi. Setelah selesai

$$LastTime - CurrentTime$$

Dengan mekanisme tersebut seluruh task dapat berjalan secara independen.
Potongan program scheduler ditunjukkan berikut.

```
unsigned long currentMillis =millis();
if(currentMillis-lastLED>=500)
{
  lastLED=currentMillis;
  TaskLED();
}
if(currentMillis-lastSensor>=2000)
{
  lastSensor=currentMillis;
  TaskSensor();
}
if(currentMillis-lastOLED>=1000)
{
  lastOLED=currentMillis;
  TaskOLED();
}
if(currentMillis-lastBuzzer>=1000)
{
  lastBuzzer=currentMillis;
  TaskBuzzer();
}
if(currentMillis-lastSerial>=500)
{
  lastSerial=currentMillis;
  TaskSerial();
}
```

Hasil Pengujian

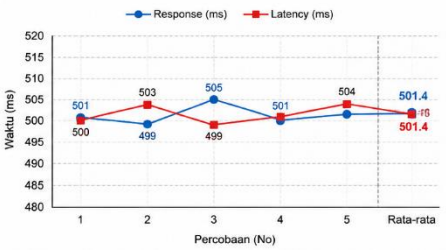
Pengujian dilakukan sebanyak 20 kali untuk memperoleh nilai rata-rata waktu respon sistem.
Parameter yang diamati adalah

- Response Time
- Task Latency
- CPU Idle Time
- Scheduler Overhead

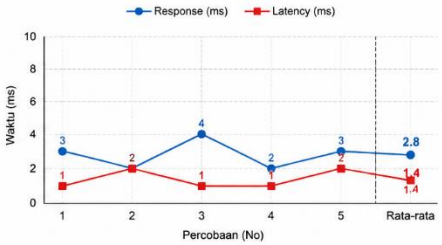
Pengujian terdiri dari :

- Pegujian Metode delay()
- Pengujian Cooperative Scheduler
- Perbandingan Kedua Metode
- Pengujian Stabilitas
- Pengujian Beban Sistem
- Penggunaan Memori

Gambar 4. dan gambar 5. Menunjukkan bahwa metode delya() memiliki waktu respon dan latency sekitar 500ms. Sedangkan Cooperative Scheduler hanya sekitar 1-3 ms.

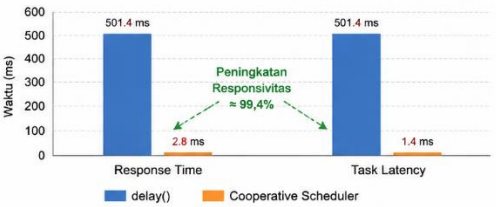


Gambar 4. Hasil Pengujian Metode Delay()



Gambar 5. Hasil Pengujian Cooperative Scheduler

Gambar 6. Memperlihatkan meningkatkan performa hingga ± 99,4%. Dalam tabel 1. Menegaskan bahwa Cooperative Scheduler memberikan sistem yang non-blocking, lebih responsif, mendukung multitasking dan lebih efisien dalam penggunaan CPU.

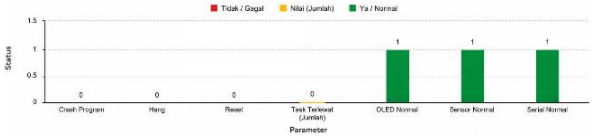


Gambar 6. Perbandingan Rata-rata Response Time dan Task Latency

Tabel 1. Perbandingan fitur Sistem

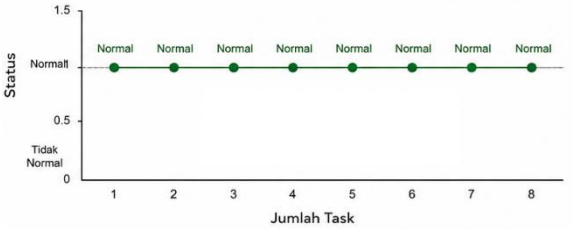
Parameter	Metode delay()	Cooperative Scheduler
Blocking	Ya	Tidak
Response Time	501 ms	2.8 ms
Task Latency	501 ms	1.4 ms
Responsif	Rendah	Sangat Tinggi
Multitasking	Tidak	Ya
Efisiensi CPU	Rendah	Tinggi

Pada gambar 7. menunjukkan scheduler bekerja stabil selama 60 menit tanpa crash, hang, maupun kehilangan task. Semua task berjalan normal.



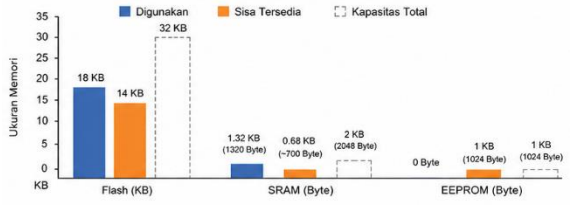
Gambar 7. Hasil Pengujian Stabilitas(durasi 60 menit)

Pada gambar 8. scheduler mampu menangani hingga 8 task dengan status normal menandakan beban sistem masih ringan untuk Arduino Uno.



Gambar 8. Hasil Pengujian Stabilitas(durasi 60 menit)

Pada gambar 9. Penggunaan memory masih jauh di bawah kapasitas Arduino Uno, sehingga masih tersedia ruang untuk penambahan fitur lainnya.



Gambar 9. Penggunaan Memory Pada Arduino Uno

KESIMPULAN DAN SARAN

Kesimpulan

Berdasarkan hasil pengujian dan perbandingan antara metode delay() dengan Cooperative Scheduler, dapat disimpulkan bahwa implementasi Cooperative Scheduler memberikan peningkatan performa yang sangat signifikan pada sistem embedded berbasis Arduino.

Metode konvensional menggunakan delay() menyebabkan eksekusi program bersifat blocking, sehingga setiap task harus menunggu task sebelumnya selesai. Kondisi ini menghasilkan rata-rata Response Time sebesar 501,4 ms dan Task Latency sebesar 501,4 ms, yang berdampak pada rendahnya responsivitas sistem serta tidak memungkinkan pelaksanaan multitasking secara efektif.

Sebaliknya, Cooperative Scheduler yang memanfaatkan fungsi millis() mampu menjalankan beberapa task secara non-blocking dan periodik. Hasil pengujian menunjukkan rata-rata Response Time hanya 2,8 ms dan Task Latency sebesar 1,4 ms. Dibandingkan metode delay(), terjadi penurunan waktu respons sekitar 99,4% dan penurunan *task latency* sekitar 99,7%, sehingga sistem menjadi jauh lebih responsif dan stabil.

Selain peningkatan kecepatan respons, Cooperative Scheduler juga memungkinkan beberapa task dijalankan secara bersamaan tanpa saling menghambat, meningkatkan efisiensi penggunaan prosesor, serta tetap memiliki kebutuhan memori yang rendah. Oleh karena itu, algoritma ini sangat sesuai diterapkan pada mikrokontroler Arduino untuk aplikasi monitoring real-time, otomasi, robotika, dan Internet of Things (IoT) yang memerlukan pengelolaan banyak tugas secara simultan tanpa menggunakan Real-Time Operating System (RTOS).

Saran

Berdasarkan hasil penelitian, beberapa saran untuk pengembangan selanjutnya adalah mengembangkan Cooperative Scheduler dengan mekanisme priority-based scheduling agar task yang bersifat kritis dapat diprioritaskan. Menerapkan algoritma pada platform mikrokontroler lain, seperti ESP32 atau STM32, serta membandingkannya dengan implementasi FreeRTOS. Menambahkan parameter pengujian, seperti CPU utilization, *task jitter*, konsumsi daya, dan penggunaan memori untuk memperoleh evaluasi performa yang lebih komprehensif. Mengimplementasikan scheduler pada aplikasi Internet of Things (IoT), robotika, dan sistem otomasi yang melibatkan lebih banyak task dan perangkat.

DAFTAR PUSTAKA

Abolhassani Khajeh, S., Saberikamarposhti, M., & Rahmani, A. M. (2022). *Real-time scheduling in IoT applications: A systematic review*. Sensors, 23(1), 232.

Barry, R. (2010). *Using the FreeRTOS™ Real Time Kernel*. Real Time Engineers Ltd.

Buonocunto, P., Biondi, A., Pagani, M., Marinoni, M., & Buttazzo, G. (2016, April). *ARTE: arduino real-time extension for programming multitasking applications*. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 1724-1731).

Gurupriya, M., Lahari, P., Sumanjali, S., & Manjunath, R. (2025, July). *Comparative Analysis of Scheduling Algorithms in Real Time Systems*. In 2025 8th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 201-205). IEEE.

Liu, J. W. (2006). *Real-time systems*. Pearson Education India.

Margolis, M., Jepson, B., & Weldin, N. R. (2020). *Arduino cookbook: recipes to begin, expand, and enhance your projects*. O'Reilly Media.

Patil, R., Navnath, S. K., Abasaheb, P. A., Prakash, H. P., & Shankar, S. V. (2022). *Design and development of a multi-tasking autonomous agriculture robot using ESP32 microcontroller*. International Journal of Research in Engineering, Science and Management, 5(7), 54-56.

Ram, S. A., Siddarth, N., Manjula, N., Rogan, K., & Srinivasan, K. (2017, March). *Real-time automation system using Arduino*. In *2017* international conference on innovations in information, embedded and communication systems (ICIIECS) (pp. 1-5). IEEE.

Restuccia, F., Pagani, M., Mascitti, A., Barrow, M., Marinoni, M., Biondi, A., ... & Kastner, R. (2022). *ARTe: Providing real-time multitasking to Arduino*. Journal of Systems and Software, 186.

Lampiran : Rangkaian Eksperimen