JuTI "Jurnal Teknologi Informasi"
Fakultas Teknologi Informasi
Universitas Teknologi Digital Indonesia (d.h STMIK AKAKOM)
vol.1,No.1, Agustus 2022

Node Sensor Pada Sistem *Monitoring* Tinggi Permukaan Air Sungai Berbasis FreeRTOS dan MQTT

Totok Budioko[1]

[1]Teknik Komputer, Universitas Teknologi Digital Indonesia
Yogyakarta, Indonesia
[1]budioko@utdi.ac.id

Ringkasan

Monitoring tinggi permukaan air sungai dapat dimanfaatkan untuk peringatan dini bahaya banjir maupun untuk pemantauan tingkat pendangkalan sungai. Salah satu bagian dari Sistem Monitoring tinggi permukaan air sungai adalah Node Sensor yang berfungsi untuk mengukur, menampilkan dan mengirim tinggi permukaan air sungai. Informasi tinggi permukaan air sungai dibutuhkan tidak hanya pada lokasi titik ukur namun juga digunakan ditempat lain sehingga dibutuhkan media transmisi dan protokol yang sesuai. Pada penelitian ini dirancang dan diimplementasi Node Sensor dari sistem Monitoring tinggi permukaan air sungai. Node Sensor menggunakan SoC ESP32, modul sensor ultrasonik HC-SR04, dan LCD. Perangkat lunak Node Sensor menggunakan RTOS FreeRTOS dan framework Arduino. Protokol komunikasi data antara Node Sensor dengan Node Monitor menggunakan protokol MQTT. Berdasarkan hasil verifikasi dengan menguji bagian-bagian Node Sensor dan pengujian pada sistem model, didapatkan bahwa semua task dieksekusi sesuai dengan rancangan dan memenuhi deadline masing-masing task. Node Sensor dapat terkoneksi dengan broker MQTT, mengirimkan alamat Node Sensor dan tinggi permukaan air dan dapat diterima pada node Monitor menggunakan aplikasi Android MyMQTT dan aplikasi desktop MQTT Explorer. Berdasrkan pengujian modul sensor ultrasonik HC-SR04 mampu mengukur dengan kesalahan rata-rata 0,17.

Kata kunci: *Node Sensor, FreeRTOS, ESP32, Arduino, MQTT*

1. Pendahuluan

Sistem *Monitoring* tinggi permukaan air sungai dapat digunakan pada sistem peringatan dini banjir atau pemantauan tingkat pendangkalan sungai. Pada sistem peringatan dini banjir, pengukuran ketinggian air sungai dapat menggunakan acuan tinggi permukaan air terhadap tinggi jagaan banjir. Tinggi jagaan merupakan tambahan tinggi pada tanggul untuk menampung loncatan arus dari permukaan air sungai yang sedang mengalir. Loncatan ini dapat terjadi akibat adanya ombak, gelombang, loncatan hidrolis pada saat terjadi banjir. [1]

Pada saat terjadi banjir, arus air bertambah dengan cepat dan kadang-kadang membawa material yang banyak dan berat. Terutama untuk sungai-sungai yang berhulu di gunung berapi yang masih aktif. Pengukuran tinggi permukaan air dengan metode non kontak mempunyai keunggulan karena peralatan *sensor* lebih aman dari kemungkinan terbawa arus atau terkena material berat. Metode non kontak yang banyak digunakan adalah menggunakan gelombang ultrasonik yang dipancarkan dari titik ukur menuju titik yang diukur sehingga gelombang ultrasonik dipantulkan. Jarak dihitung berdasarkan waktu tempuh gelombang ultrasonik antara alat ukur dengan obyek yang diukur. [2] [3] [4].

Posisi titik ukur tinggi permukaan air sungai dengan pengguna data hasil ukur biasanya terletak pada jarak yang jauh sehingga perlu ditransmisikan melalui media transmisi. Media transmisi yang banyak digunakan adalah media nirkabel dengan protokol Wifi IEEE 802.11.[2] [3] [4]. Selain menggunakan Wifi juga dapat menggunakan GPRS pada jaringan telepon seluler, seperti pada artikel [5] dan radio Lora seperti pada artikel [6]. Rancangan dan implementasi *Node Sensor* pada penelitian ini menggunakan protokol Wifi IEEE 802.11 untuk koneksi ke jaringan *internet*.

Node Sensor pada sistem *Monitoring* tinggi permukaan air sungai harus berupa sistem *realtime* karena informasi tinggi permukaan air sungai harus terukur tepat waktu. Oleh karena itu penggunaan *Real Time*

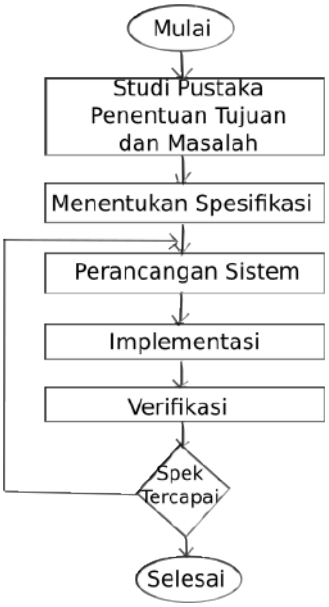
This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Operating System (RTOS) akan memudahkan dalam implementasi sistem yang *real time*. Hal ini karena pada RTOS sudah tersedia *Application Programming Interface* (API) untuk mengimplementasikan algoritma penjadwalan *Realtime*. Pada penelitian ini RTOS yang akan digunakan adalah FreeRTOS. FreeRTOS bersifat free dan open. [7][8] [9].

Protokol yang digunakan dalam aplikasi *internet of Things* (IoT) antara lain, Message Queue Telemetry Transport(MQTT), Constrained Application Protocol (COaP), Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP), dan WEBSOCKET. Berbagai protokol IoT tersebut protokol MQTT merupakan protokol yang banyak digunakan dalam aplikasi IoT [5]. Pada penelitian ini protokol MQTT akan digunakan dalam Sistem *Monitoring* Tinggi Permukaan Air Sungai untuk komunikasi data antar node pada sistem *Monitoring*.

2. Metode Penelitian

Tahapan penelitian yang digunakan ditunjukkan pada Gambar 1. Pada tahap awal dilakukan studi pustaka berkenaan dengaan perancangan dan implementasi sistem *Monitoring* tinggi permukaan air sungai pada *Node Sensor*. Pada tahap ini juga ditentukan tujuan dan masalah penelitian yaitu menghasilkan prototipe *Node Sensor* yang bersifat *realtime* dan memverifikasi rancangan *Node Sensor* pada sistem *Monitoring* tinggi permukaan air. Verifikasi *Node Sensor* terdiri atas pengujian kerja modul sensor ultrasonik HC-SR04[2][4], pengujian pengiriman data tinggi permukaan air pada *Node Monitor* dan *NodeDB*[2], serta pengujian eksekusi *task*[9]. Spesifikasi *Node Sensor* adalah sebagai berikut: 1. tinggi permukaan air diukur menggunakan ultrasonik, 2. *Lyquid Crystal Display* (LCD) untuk menampilkan hasil pengukuran, 3. kontroler menggunakan *System on Chip* (SoC) ESP32, 4. koneksi jaringan internet menggunakan *Wifi*, 5. program berbasis FreeRTOS untuk mengukur jarak dan mengirimkan hasil ukur ke *broker* MQTT menggunakan protokol MQTT



Gambar 1. Metode Penelitian

2.1. Perancangan *Node Sensor* Pada Sistem *Monitoring* Tinggi Permukaan Air Sungai

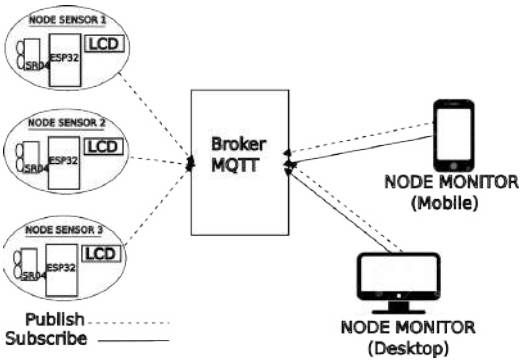
Protokol MQTT menggunakan arsitektur *Publish/Subcribe*. Komunikasi antar *client* dilakukan melalui *server* yang dikenal dengan istilah *broker*. *Client* pengirim data melakukan *Publish*. *Client* yang akan menerima data dari *client Publish* harus melakukan *Subcribe* terlebih dahulu dan selanjutnya setiap data yang *dipublish* akan diterima. Pemisahan dan penyaringan data *Publish* dan *Subcribe* menggunakan topik. Topik berupa karakter UTF-8 *case-sensitive* yang dapat disusun secara bertingkat menggunakan pemisah / (*slash*), sebagai contoh "Yogya/levelSungai".[10]

Arsitektur Sistem *Monitoring* Tinggi Permukaan Air Sungai terdiri atas *Node Sensor* dan *Node Monitor*. *Node Sensor* berfungsi untuk mengambil data tinggi permukaan air dan mengirimkanya ke *broker* MQTT. *Node Monitor* digunakan untuk menampilkan data tinggi permukaan air sungai secara *live*. *Diagaram* blok Sistem *Monitoring* Tinggi Permukaan Air Sungai secara lengkap diperlihatkan pada Gambar 2.

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Tabel 1. Spesifikasi ESP32[13]

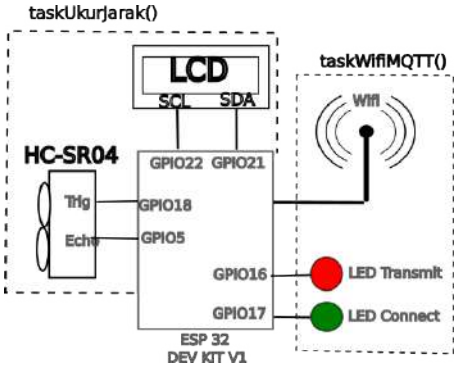
Spesifikasi	Keterangan
CPU	single/dual core Xtensa® 32-bit LX6
Internal Memory	448 KB ROM dan 520 KB SRAM
Wireless	Wifi dan Bluetooth
Interface	< UART, I2C, SPI
Peripheral	ADC, DAC, PWM



Gambar 2. Sistem *Monitoring* Tinggi Permukaan Air Sungai. [5]

2.1.1. Perancangan Perangkat Keras *Node Sensor*

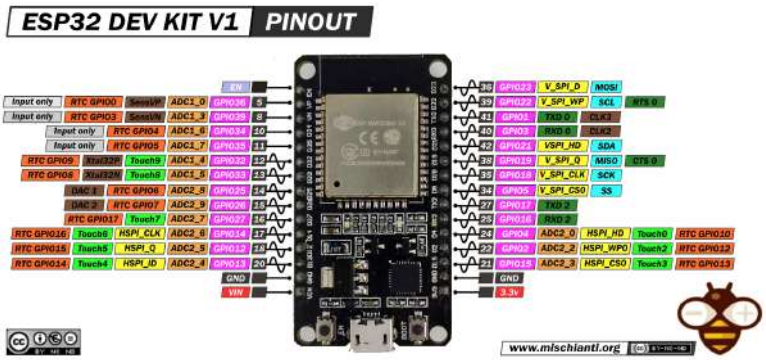
Perangkat keras *Node Sensor* menggunakan *System on Chip* ESP32 sebagai kontrollernya, modul ultrasonik HC-SR04 untuk mengukur tinggi permukaan air dan LCD untuk menampilkan hasil ukur secara lokal. *Diagaram* blok sistem *Node Sensor* seperti pada Gambar 3. Modul ultrasonik HC-SR04 dikoneksikan ke ESP32 melalui *General Port Input Output* (GPIO) 18 dan 5 sedangkan LCD melalui antarmuka *Inter Integrated Circuit* (I2C) pada GPIO 22 dan 21.



Gambar 3. *Diagaram* Blok *Node Sensor* [11][12]

ESP32 adalah *System on Chip* yang diproduksi oleh *Espressif Systems* (www.espressif.com) dengan spesifikasi secara singkat ditunjukkan pada Tabel 1. Untuk pengembangan perangkat keras maka digunakan papan pengembang DOIT Esp32 DevKit v1 seperti pada Gambar 4

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.



Gambar 4. Papan Pengembang ESP32[14]

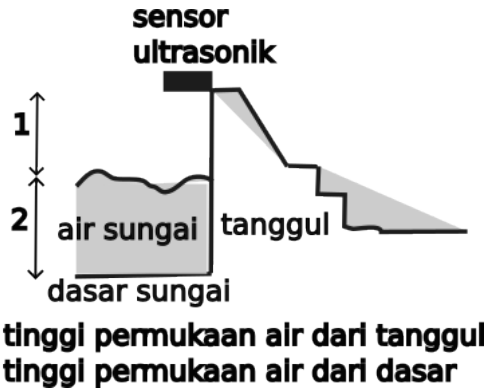
Pengukuran tinggi permukaan air dilakukan dengan menggunakan gelombang ultrasonik. Gelombang ultrasonik dipancarkan dari *transduser* pengirim yang diarahkan ke obyek yang akan diukur jaraknya. Karena mengenai obyek yang akan diukur maka gelombang ultrasonik akan dipantulkan kembali dan diterima oleh *transducer* penerima . Jarak antara modul ultrasonik dan obyek yang diukur dapat dihitung menggunakan Persamaan 1[11][4]. Kecepatan gelombang ultrasonik dipengaruhi oleh media dan suhu. Pada kondisi udara kering (kelembaban 0%) pengaruh suhu terhadap kecepatan suara dapat dihitung menggunakan Persamaan 2[15]. Pada penelitian ini perhitungan kecepatan rambat suara menggunakan suhu rata-rata tahunan di D.I. Yogyakarta tahun 2016 sebesar 26,7℃ [16], sehingga kecepatan rambat udara sebesar 332.98 m/s.

$$jarak = \frac{(V * waktuTempuh)}{2} \tag{1}$$

$$V = (331.3 + 0.606 * T) \quad \mathrm{m/s}, \tag{2}$$

ket:
V : kecepatan rambat suara di udara
T : suhu dalam derajat Celsius (℃)

Untuk tujuan peringatan dini atau tingkat pendangkalan, tinggi permukaan air dapat diukur relatif terhadap tinggi jagaan air (tanggul) sehingga nilainya akan berbanding terbalik dengan pengukuran jarak sensor dengan permukaan air. Unutk tujuan perhitungan volume air, diperlukan ketinggian air relatif terhadap dasar air[4]. Untuk memberikan gambaran pengukuran ketinggian air dengan acuan yang berbeda diperlihatkan pada Gambar 5.



Gambar 5. Ilustrasi Pengukuran Tinggi Permukaan Air

Node Sensor menggunakan modul ultrasonik HC-SR04 untuk mengukur jarak antara modul pengukur dengan permukaan air. Modul HC-SR04 sudah dilengkapi dengan rangkaian penguat gelombang dan logika. Bentuk modul HC-SR04 diperlihatkan seperti Gambar 6 dan fungsi pin pada modul HC-SR04 dapat dilihat pada Tabel .

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

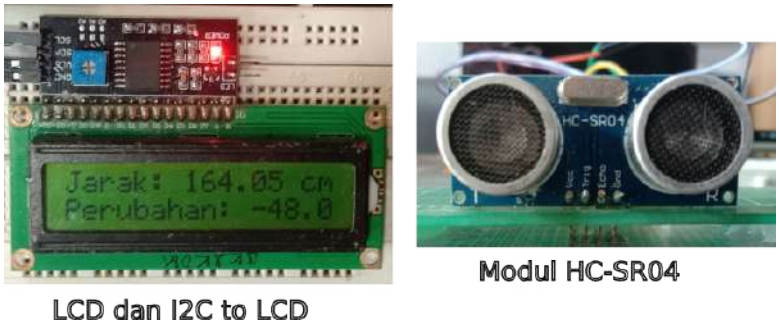
Tabel 2. Fungsi Kaki Modul HC-SR04[11]

No. PIN	Nama	Fungsi
1	VCC	+5V
2	Trigger	Mengaktifkan Kirim
3	Echo	Sinyal pantul (HIGH)
4	GND	Ground

Untuk memancarkan gelombang ultrasonik, pin Trigger diberi pulsa tinggi selama minimal 10uS maka kan dikirimkan sejumlah pulsa ultrasonik. Pulsa tersebut jika mengenai obyek maka akan dipantulkan kembali dan diterima oleh *transducer* penerima. Jika modul menerima gelombang ultrasonik pantulan maka kaki *Echo* akan berlogika tinggi. Waktu yang dibutuhkan gelombang ultrasonik dihitung untuk mendapatkan jaraknya[17].

Tabel 3. Spesifikasi Modul HC-SR04[11]

Spesifikasi	Nilai
Power Supply	+5V DC
Quiescent Current	<2mA
Working current	15mA
Effectual Angle	<15º
Resolution	0.3 cm
Measuring Angle	30º
Trigger Input Pulse width	10uS



Gambar 6. Modul Ultrasonik HC-SR04 (Foto sendiri)

Pada *Node Sensor* LCD digunakan untuk menampilkan hasil pengukuran. LCD yang digunakan adalah LCD karakter. Untuk menghemat kaki ESP32 maka LCD ditambahkan konverter I2C to LCD sehingga antarmuka ke ESP32 menggunakan *Inter Integrated Circuit* (I2C). Gambar LCD dan konverter I2C seperti Gambar 6.

2.1.2. Perancangan Perangkat Lunak *Node Sensor*

Perangkat lunak *Node Sensor* menggunakan sistem operasi *Realtime* (RTOS) FreeRTOS. FreeRTOS merupakan sistem operasi yang mendukung algoritma penjadwalan *Preemptive* berdasarkan prioritas *task* [18]. Aplikasi yang dibangun di atas FreeRTOS dibagi menjadi beberapa *task*. *task* pada FreeRTOS adalah potongan program kecil dalam bentuk fungsi yang diimplementasikan menggunakan bahasa C yang *task* mempunyai parameter masukan dan harus dengan nilai balik bertipe void. *Task* juga harus diimplementasikan dengan kalang tertutup yang tidak pernah selesai[19].

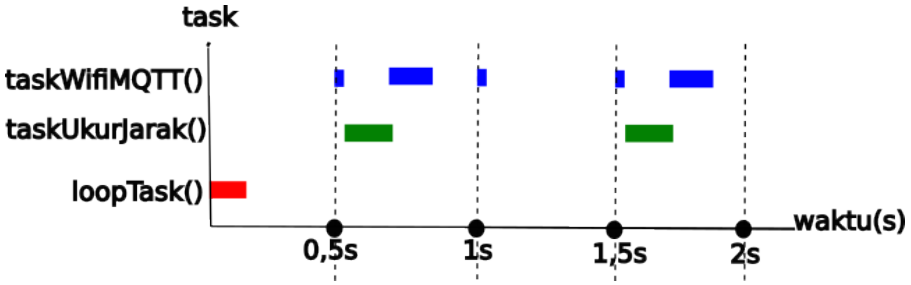
Perangkat lunak *Node Sensor* terdiri atas tiga buah *task* yang diberi nama *task* "loopTask()", "taskUkurJarak()" dan "taskWifiMQTT()". "loopTask()" merupakan *task* yang berfungsi memanggil fungsi setup() dan loop() dibuat agar sesuai dengan *framework* Arduino. *Task* pada ESP32 (*dual core*) dieksekusi pada dua CPU, yaitu CPU0 dan CPU1. Espressif menempatkan *support task* (mis. layanan untuk WiFi, TCP/IP, Bluetooth,dll) pada CPU0 dan *task* aplikasi pada CPU1[18].

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

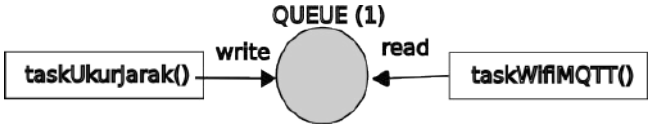
Rancangan prioritas masing-masing *task* seperti pada Tabel 4. loopTask() hanya dieksekusi sekali dengan prioritas 1. taskUkurJarak prioritas 2 dan dieksekusi setiap 1 detik. taskWifiMQTT() priortas 3 dan dieksekusi setiap 0,5 detik. Nilai besar pada prioritas menandakan prioritasnya tinggi.[19]. Komunikasi dan sinkronisasi data tinggi air antara taskUkurJarak() dengan taskWifiMQTT() menggunakan *queue*. *Queue* adalah memori yang mempunyai ukuran dan panjang. Normalnya *queue* bersifat *First Input First Output (FIFO)*. *Task* pengirim akan terblokir jika *queue* penuh dan *task* penerima akan terblokir jika *queue* kosong[19]. Prinsip kerja *queue* diperlihatkan pada Gambar 8. taskWifiMQTT() tergantung terhadap data dari taskUkurJarak(), untuk memastikan bahwa taskWifiMQTT() tidak terlewatkan dan menyebabkan terblokirnya taskUkurJarak(), maka periode eksekusi taskWifiMQTT() dibuat lebih cepat dan prioritas lebih tinggi. Ketiga *task* dieksekusi pada CPU1. *Diagaram* eksekusi berdasarkan prioritas masing-masing *task* diperlihatkan pada Gambar 7.

Tabel 4. Prioritas *Task*

task	Periode	Prioritas
loopTask	sekali	1
taskUkurJarak	1000ms	2
taskWifiMQTT	500	3



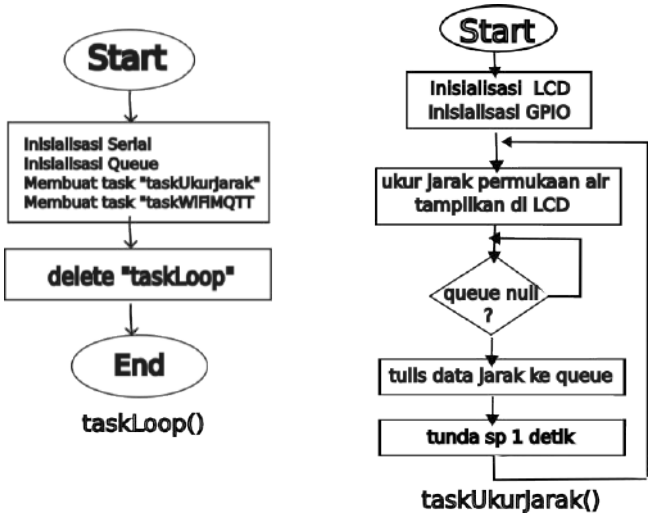
Gambar 7. *Diagaram* Penjadwalan *task*



Gambar 8. Prinsip Kerja *Queue*[19]

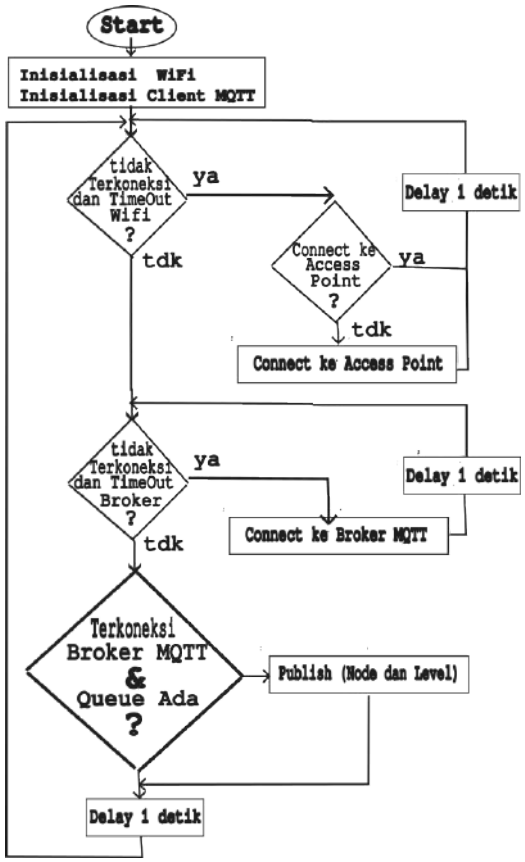
task loopTask() digunakan untuk memanggil fungsi setup() dan loop(). Pada *framework* Arduino fungsi setup() hanya dieksekusi sekali pada saat program dijalankan dan digunakan untuk melakukan proses inisialisasi termasuk pembuatan *task* taskUkurJarak() dan taskWifiMQTT(). Untuk menghemat sumber daya CPU, taskLoop() dihapus dari penjadwalan[18]. *Diagaram* alir loopTask() diperlihatkan pada Gambar 9

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.



Gambar 9. *Diagaram* Alir taskLoop() dan taskUkurJarak()

task taskUkurJarak() digunakan untuk mengukur jarak dari tanggul ke permukaan air, yang hasilnya ditampilkan di LCD dan dikirim ke *Queue*. *Diagaram* alir taskUkurJarak() diperlihatkan pada Gambar 9. *Task* taskWifiMQTT() digunakan untuk mengelola koneksi *Node Sensor* ke *Access Point*, *broker MQTT*, dan pengiriman data tinggi permukaan air ke *broker MQTT*. Data tinggi permukaan air didapatkan dari *Queue*. *Diagaram* alir taskWifiMQTT() diperlihatkan pada Gambar 10.



Gambar 10. *Diagaram* Alir taskWifiMQTT()

Format data yang dikirimkan ke *broker MQTT* terdiri atas alamat *Node Sensor* dan data tinggi permukaan air. Alamat *Node Sensor* pada penelitian ini menggunakan nama Node. Format data yang dikirim seperti pada Gambar 11. Untuk mengenali akhir dari data alamat digunakan karakter "pagar". Data setelah karakter

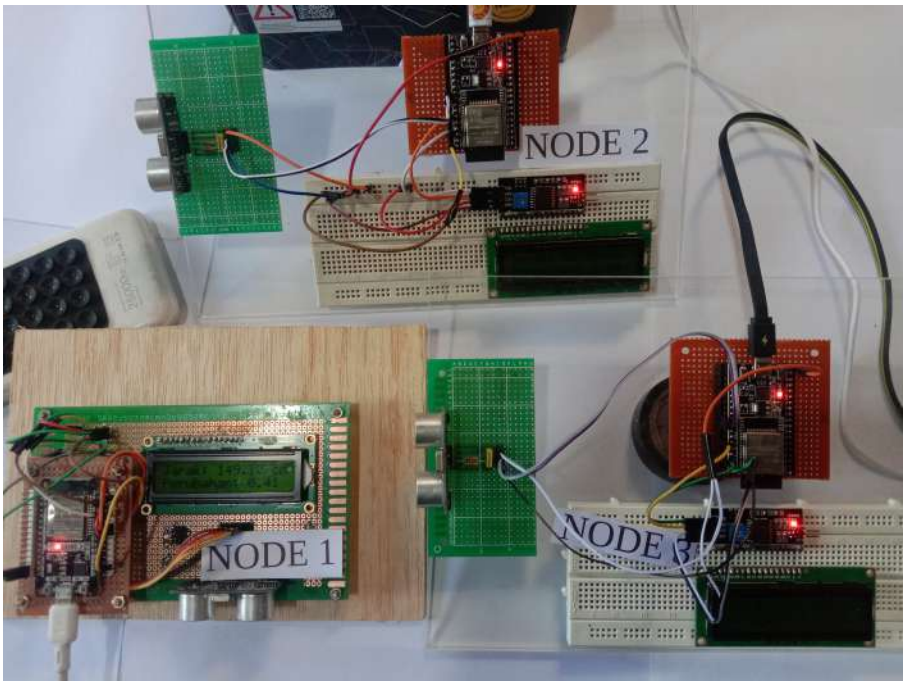
This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

"pagar" adalah data tinggi permukaan air yang bertipe float dengan lebar data 32 *bit* (4 *byte*).

Gambar 11. *Diagaram* Format Pengiriman Data

3. Hasil dan Pembahasan

Pada penelitian ini *Node Sensor* dibuat sebanyak tiga buah. Hasil prototipnya diperlihatkan pada Gambar 12.



Gambar 12. Prototip *Node Sensor*

Pemrograman HC-SR04 menggunakan *library* HCSR04[20], pemrograman LCD menggunakan *library* LiquidCrystal_I2C dan pemrograman *client MQTT* menggunakan *library* PubSubClient[22].

Verifikasi fungsi modul HC-SR04 dilakukan dengan menguji modul HC-SR04 untuk mengukur obyek yang diketahui jaraknya menggunakan penggaris. Hasil pengukuran diperlihatkan pada Tabel 5. Berdasarkan hasil pengukuran tersebut didapatkan rata-rata kesalahan sebesar 0,17.

Tabel 5. Pengujian Modul HC-SR04

Penggaris (cm)	HC-SR04 (cm))	Kesalahan
6	6,18	0,18
10	9,84	0,16
15	15,08	0,08
20	19,87	0,13
25	25,17	0,17
100	100,24	0,08
150	150,31	0.31
200	200,26	0,26

Verifikasi program pada *node Sensor* meliputi pengukuran waktu eksekusi tiap *task*, pengujian eksekusi *task*, dan pengujian pengiriman data ke *boker MQTT*.

Hasil pengukuran waktu eksekusi masing-masing *tasl* diperlihatkan pada Tabel 6. Waktu eksekusi taskUkurJarak() sebesar 65,522 milidetik. Waktu eksekusi ini merupakan waktu eksekusi paling lama dari taskUkur-

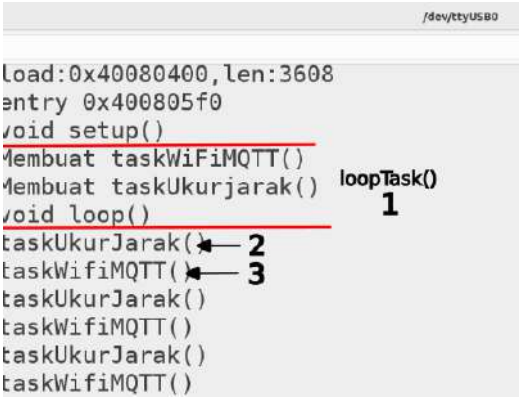
This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Jarak(), yaitu pada saat jarak ukurnya melebihi jarak maksimumnya yaitu 4 meter. Periode taskUkurJarak() 1 detik dan taskWifiMQTT() sebesar 0,5 detik sehingga semua *task* dapat dieksekusi.

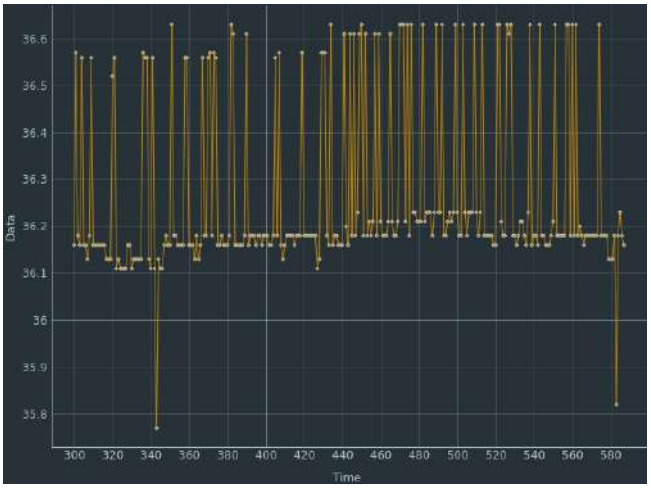
Tabel 6. Waktu Eksekusi *Task*

Task	Waktu Eksekusi (mS)
loopTask()	2,678
taskUkurJarak()	65,522
taskWifiMQTT()	38,433

Hasil eksekusi *task* dimulai dari kondisi *reset* diperlihatkan pada Gambar 13. Satu-satunya *task* pada CPU1 yang berjalan adalah taskLoop(). No 1 adalah bagian dari loopTask() yang hanya dieksekusi sekali. Pertama memanggil fungsi setup(), membuat taskWifiMQTT() dan taskUkurJarak(), kemudian memanggil fungsi loop(). Fungsi loop() hanya mengeksekusi instruksi menghapus loopTak(). Oleh karena itu pada periode berikutnya tinggal dua *task* yaitu taskUkurJarak() dan taskWifiMQTT(). No 2 adalah eksekusi dari taskUkurJarak(). taskUkurJarak() dieksekusi walaupun mempunyai prioritas lebih rendah dari taskWifiMQTT(). taskWifiMQTT() terblokir karena *Queue* masih kosong. Setelah taskUkurJarak() dieksekusi dan menuliskan data ke *Queue* maka taskWifiMQTT() dieksekusi. Pada implementasi *deadline* sama dengan periodenya yaitu 1 detik untuk taskUkurJarak() dan 0,5 detik untuk taskWifiMQTT(). taskWifiMQTT() tergantung dengan data dari taskUkurJarak() oleh karena itu taskWifiMQTT() selalu dieksekusi setelah taskUkurJarak(). Dengan demikian taskWifiMQTT akan mengirimkan data ke *broker MQTT* disinkronkan dengan kecepatan menulis taskUkurJarak() ke *Queue*.



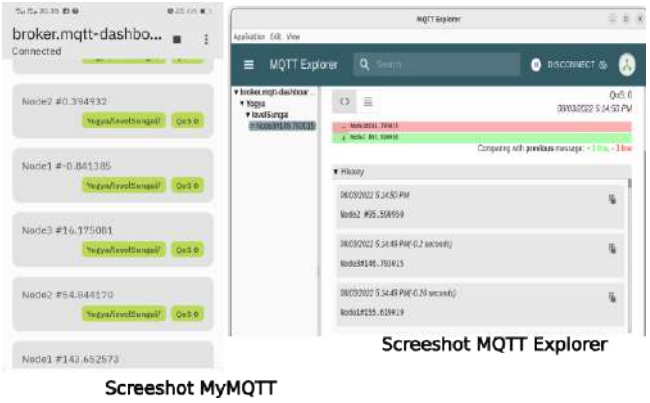
Gambar 13. Hasil Uji Eksekusi *Task*



Gambar 14. Hasil Pengujian Tampil Dalam Grafik

Pengujian *Node Sensor* dilakukan pada sungai kecil yang airnya mengalir. Data hasil pengukuran tinggi permukaan air diplot menggunakan aplikasi *Tauno-Serial Plotter* seperti pada Gambar 14 dan dimonitor meng-

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.



Gambar 15. Pengujian *Node Monitor* Menggunakan MyMQTT

gunakan aplikasi MyMQTT di *platform Android* dan aplikasi MQTT Explorer. Hasil *screenshot* diperlihatkan pada Gambar 15. Terlihat bahwa ketiga *Node Sensor* dapat diterima di aplikasi MyMQTT maupun MQTT Explorer.

4. Simpulan

Rancangan aplikasi *Node Sensor* yang bersifat *realtime* menggunakan aplikasi basis FreeRTOS dan protokol MQTT berhasil dibuat prototipnya dan diverifikasi. Semua *task* dapat dieksekusi tanpa melewati *deadlinenya* yaitu 1 detik untuk taskUkurJarak() dan 0,5 detik untuk taskWifiMQTT(). taskWifiMQTT() akan dieksekusi setelah eksekusi taskUkurJarak() karena taskWifiMQTT tergantung dengan data yang ditulis pada *Queue*. Hasil pengujian pada *Node Monitor* menunjukkan bahwa ketiga *Node Sensor* dapat mengirim data pada topik yang sama. Hasil pengujian modul HC-SR04 didapatkan rata-rata kesalahan sebesar 0,17.

Pustaka

[1] A. Fatharani and B. Sujatmoko, "ANALISIS TINGGI TANGGUL SEBAGAI BANGUNAN PENGENDALI BANJIR MENGGUNAKAN METODE HEC-RAS," *Jurnal Online Mahasiswa Fakultas Teknik*, vol. 5, p. 8, 2018. [Online]. Available: https://jom.unri.ac.id/index.php/JOMFTEKNIK/article/view/22041

[2] M. Kresna and K. E. Susilo, "Monitoring Level Air Pada Waduk Secara Realtime Berbasis IoT Memanfaatkan Aplikasi Telegram," *Jurnal SISKOM-KB (Sistem Komputer dan Kecerdasan Buatan)*, vol. 5, no. 1, pp. 30–37, Sep. 2021. [Online]. Available: https://jurnal.tau.ac.id/index.php/siskom-kb/article/view/223

[3] A. S. I. Nafik, A. Widodo, F. Baskoro, and R. Rahmadian, "RANCANG BANGUN PROTOTYPE MONITORING KETINGGIAN AIR PADA BENDUNGAN BERBASIS INTERNET OF THINGS," *Jurnal Teknik Elektro UNESA*, vol. 10, p. 7, 2021. [Online]. Available: https://ejournal.unesa.ac.id/index.php/JTE/article/view/36279

[4] D. N. Ramadan, S. Hadiyoso, and I. D. Irawati, "Sistem Monitoring Ketersediaan Air pada Perangkat Cuci Tangan Portable berbasis IoT," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 9, no. 2, p. 455, Apr. 2021. [Online]. Available: https://ejurnal.itenas.ac.id/index.php/elkomika/article/view/4444

[5] E. Sacoto-Cabrera, J. Rodriguez-Bustamante, P. Gallegos-Segovia, G. Arevalo-Quishpi, and G. Leon-Paredes, "Internet of Things: Informatic system for metering with communications MQTT over GPRS for smart meters," in *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. Pucon: IEEE, Oct. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8229598/

[6] M. Rosmiati, M. F. Rizal, F. Susanti, and G. F. Alfisyahrin, "Air pollution monitoring system using LoRa modul as transceiver system," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 2, p. 586, Apr. 2019. [Online]. Available: http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/11760

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

[7] 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

[8] 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

[9] D. I. Saputra and R. A. Permana, "PERANCANGAN DAN IMPLEMENTASI REAL TIME OPERATING SYSTEM PADA SISTEM KENDALI SUHU KANDANG AYAM SECARA CLOSED LOOP," *Journal of Energy and Electrical Engineering (JEEE) TE.Universitas Siliwangi*, vol. 03, no. 02, p. 7, 2022. [Online]. Available: https://jurnal.unsil.ac.id/index.php/jeee/article/view/4744/0

[10] G. C. Hillar, *MQTT Essentials - A Lightweight IoT Protocol*. Packt Publishin, 2017.

[11] R. Santos and S. Santos, "ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE | Random Nerd Tutorials," Jul. 2021. [Online]. Available: https://randomnerdtutorials.com/esp32-hc-sr04-ultrasonic-arduino/

[12] ——, "I2C LCD with ESP32 on Arduino IDE - ESP8266 compatible | Random Nerd Tutorials," Feb. 2019. [Online]. Available: https://randomnerdtutorials.com/esp32-esp8266-i2c-lcd-arduino-ide/

[13] Expressif, "ESP32 Datasheet V3.8," 2022.

[14] R. Mischianti, "DOIT ESP32 DEV KIT v1 high resolution pinout and specs – Renzo Mischianti." [Online]. Available: https://www.mischianti.org/2021/02/17/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/

[15] "Speed of sound - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Speed_of_sound

[16] D. BPS, "Suhu Rata-rata Tahunan D.I. Yogyakarta." [Online]. Available: https://yogyakarta.bps.go.id/indicator/151/150/1/suhu-udara-jumlah-hujan-dan-hari-hujan-per-bulan-di-d-i-yogyakarta.html

[17] E. J. Morgan, "HCSR04 Ultrasonic Sensor," 2014.

[18] W. Gay, *FreeRTOS for ESP32-Arduino: practical multitasking fundamentals*, 1st ed. London: Elektor International Media BV, 2020.

[19] R. Barry, *USING THE FREERTOS REAL TIME KERNEL A PRACTICAL GUIDE*. FreeRTOS.org, 2009. [Online]. Available: hreeRTOS.org

[20] M. Sosic, "GitHub - Martinsos/arduino-lib-hc-sr04: Arduino library for HC-SR04 ultrasonic distance sensor." [Online]. Available: https://github.com/Martinsos/arduino-lib-hc-sr04

[21] J. Rickman, "GitHub - johnrickman/LiquidCrystal_i2c: LiquidCrystal Arduino library for the DFRobot I2C LCD displays." [Online]. Available: https://github.com/johnrickman/LiquidCrystal_I2C

[22] N. O'Leary, "Arduino Client for MQTT." [Online]. Available: https://pubsubclient.knolleary.net/

This Article is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.