SinarFe7
Seminar Nasional Fortei Regional 7

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

SISTEM PEMBACAAN DATA POWER METER DENGAN KOMUNIKASI MODBUS SECARA TERPUSAT

[1]Jonathan Teng, [2]Julius Sentosa Setiadji, [3]Resmana Lim
[1 2 3] Teknik Elektro, Universitas Kristen Petra, Surabaya
[1]myblog.jonathan@gmail.com , [2] julius@petra.ac.id, [3] resmana@petra.ac.id

Abstract - Power meters are used to monitor the usage of electrical energy in a room or a building. To see in detail the usage of electricity in one room of a building, a power meter is deployed to several points. Because of the scattered location of the power meter, a power meter that has a data communication feature is needed. With that feature, readed data from the power meter can be sent and collected centrally.

The system is made using two devices. the first device is a gateway that reads data from scattered power meters. Gateway has a functions to convert Modbus RTU data from power meter to modbus TCP / IP data. The second device is the server that receives Modbus TCP / IP data from the gateway and displays it to the web page. The server will also send a notification to the smartphone if the power meter reading condition shows a certain value wich more than threshold value.

The system that has been created produces a gateway device that can send power meter data to the server, a web page to display power meter data, a database system to store data from the power meter, and a notification system via the Line app. The system that has been created is able to display power meter data to the web page with a time delay of 864.09 milliseconds to 943.35 milliseconds from the real value from the power meter.

Keywords — Data Acquisition, Power Meter, Raspberry Pi, Node-Red, Modbus, ESP8266

***Abstrak* - *Power meter* digunakan untuk memantau penggunaan energi listrik pada suatu ruangan atau gedung. Untuk melihat secara mendetail penggunaan listrik pada satu ruang pada suatu bangunan, digunakan *power meter* yang disebar pada beberapa titik. Karena letak *power meter* yang tersebar, maka diperlukan *power meter* yang memiliki fitur komunikasi data agar data pembacaan dapat dikirimkan dan dikumpulkan secara terpusat.**

Sistem dibuat dengan menggunakan dua perangkat. yang pertama adalah *gateway* yang membaca data dari *power meter* yang tersebar. *Gateway* berfungsi untuk merubah data *Modbus RTU* menjadi *modbus TCP/IP.* Perangkat kedua yaitu *server* menerima data *Modbus TCP/IP* dari *gateway* lalu menampilkannya menuju halaman *web. Server* juga akan mengirim notifikasi menuju *smartphone* apabila kondisi pembacaan *power meter* menunjukkan nilai tertentu yang melebihi nilai *threshold* dari *user*.

Sistem yang telah dibuat menghasilkan perangkat *gateway* yang dapat mengirimkan data *power meter* menuju *server*, halaman web untuk tampilan data *power meter* beserta halaman untuk konfigurasi, sistem *database* untuk menyimpan data dari *power meter*, dan sistem notifikasi melalui aplikasi *Line.* Sistem yang telah dibuat mampu menampilkan data *power meter* menuju halaman web *dengan* jeda waktu 864.09 milidetik hingga 943.35 milidetik dari nilai riil pada *power meter*.

***Kata Kunci - Data Acquisition, Power Meter, Raspberry Pi, Node-Red, Modbus, ESP8266*.**

I. Pendahuluan

Untuk mengukur besaran daya aktif yang digunakan dalam satu waktu, digunakan alat ukur yang bernama kWh meter. kWh meter memberikan informasi secara *real time* konsumsi daya aktif yang digunakan pada suatu sistem kelistrikan [1]. Secara umum jenis kWh meter dibagi menjadi dua jenis, yang pertama adalah kWh meter analog dan yang kedua adalah kWh meter digital [2]. Dewasa ini, kWh meter digital banyak digunakan karena pembacaan yang lebih mudah dan fitur yang lebih banyak dari kWh meter analog. Beberapa jenis kWh meter digital sudah dilengkapi dengan sistem komunikasi data sehingga memungkinkan untuk melalukan pengukuran dengan jarak jauh. Sistem pengukuran ini disebut dengan sistem telemetri [3]. Pada perkembangannya, istilah *power meter* digunakan untuk menyebut kWh meter digital dengan fitur komunikasi data dan telemetri.

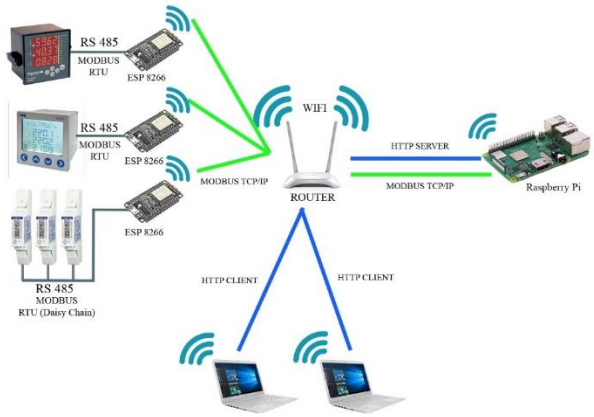
Sistem telemeteri pada *power meter* digital yang sudah jamak digunakan adalah komunikasi serial *RS485* dengan protokol *Modbus RTU*. Data *Modbus* dikirim melalui *port serial RS485* menggunakan sarana dua buah kabel untuk berkomunikasi dengan jarak maksimum hingga 1,2 Kilometer. Sistem komunikasi *Modbus* yang sudah ada lalu diperbarui dengan model komunikasi *TCP/IP* yang memanfaatkan protokol *TCP/IP* untuk dapat membawa data *Modbus*. Dengan menggunakan *Modbus TCP/IP* maka komunikasi *Modbus* dapat dilakukan dengan menggunakan sarana kabel Ethernet atau jaringan WiFi. Data *Modbus* dapat dikirim di dalam jaringan lokal maupun internet. Dengan digunakannya *Modbus TCP/IP* maka komunikasi dapat

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

dilakukan lebih jauh lagi dengan memanfaatkan protokol *TCP/IP*.

II. Metode Penelitian

Gambar 1 menggambarkan skema komunikasi mengenai penelitian yang akan dibuat. Data dari *power meter* akan dibaca menggunakan jalur komunikasi serial *RS485* dengan protokol *Modbus RTU*. Data *Modbus RTU* akan dibaca oleh perangkat *ESP 8266* yang berperan sebagai *Modbus gateway*. Beberapa *power meter* dapat dihubungkan secara *daisy-chain* dan dibaca oleh *gateway*. Data *Modbus RTU* yang telah didapatkan oleh *gateway* lalu diubah menjadi data *Modbus TCP/IP*. *ESP 8266* lalu mengirimkan data *Modbus TCP/IP* yang telah masuk melalui koneksi Wireless LAN menuju *server Raspberry Pi*. Data *Modbus TCP/IP* yang telah diterima dibaca oleh *Raspberry Pi* yang juga terkoneksi pada jaringan Wireless LAN yang sama. Data yang telah terbaca oleh *Raspberry Pi* lalu ditampilkan menuju *Web* dan disimpan kedalam *database*.



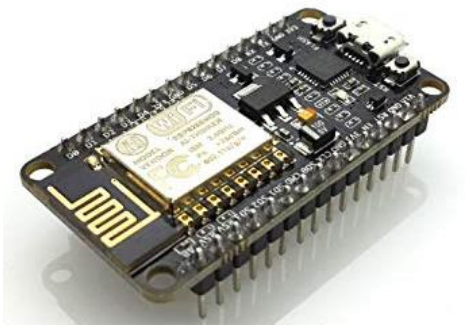
Gambar 1 Diagram system

A. *Desain Gateway*

Desain *gateway* yang dibuat terdiri atas dua bagian. Bagian pertama adalah hardware yang didalamnya terdapat 2 bagian *hardware* yaitu *RS485* to TTL *Converter* dan mikrokontroller *ESP 8266*. Bagian kedua merupakan software yang dieksekusi oleh mikrokontroller *ESP 8266*. Untuk pembuatan software menggunakan program dari *Arduino IDE* dengan tambahan beberapa *library*.

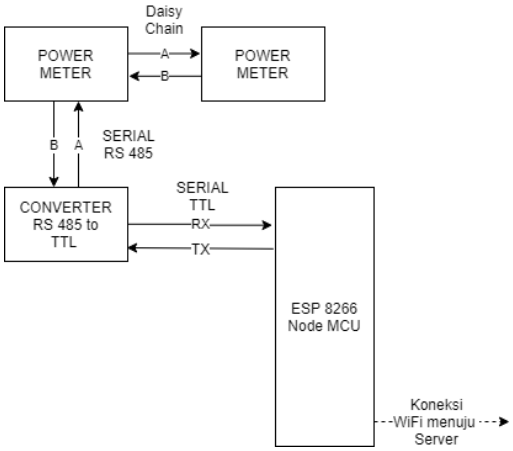
Hardware yang digunakan dalam pembuatan *gateway* ini adalah *converter RS485 to* TTL dan modul *ESP 8266* Node MCU. *Converter RS485 to* TTL berguna untuk mengkonversi data dari komunikasi serial *RS485* menjadi data serial untuk komunikasi serial TTL. Desain pada. perangkat *converter RS485 to* TTL ini menggunakan basis dari IC MAX485 yang memiliki fungsi dasar mengkonversi data *serial full duplex* dengan level tegangan TTL menjadi data *serial half duplex* dengan level tegangan sesuai dengan standar serial *RS485* dan sebaliknya. Terdapat fitur *Automatic flow control* yang berguna untuk mengatur aliran data *serial full duplex* TTL agar tidak menjadi kacau saat dikonversi menjadi data *serial half duplex RS485*.

Modul *ESP 8266 Node MCU* (Gambar 2) membaca data serial TTL, hasil dari konversi data serial *RS485* perangkat *converter RS485 to* TTL . Data yang telah dibaca oleh *ESP 8266* kemudian disimpan ke dalam sebuah variabel untuk dikirim melalui jaringan *local area network* dengan perantaraan *WiFi*.



Gambar 2. Desain *board* I/O [4]

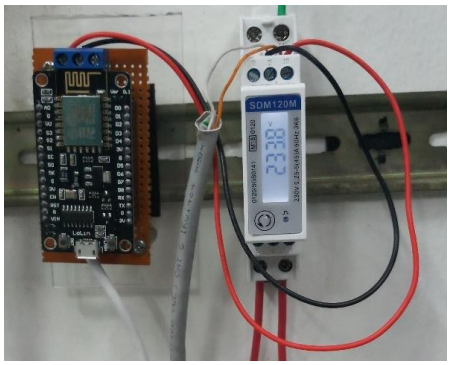
Gambar 3 menunjukkan skema koneksi *hardware* pada gateway. Koneksi data serial RS485 pada *converter RS 485 to TTL* menuju ke *power meter* dapat dilakukan secara *daisy chain*. Setelah data serial RS485 dari *power meter* terkonversi menjadi data serial TTL, data serial TTL dibaca oleh mikrokontroller *ESP 8266* melalui pin *hardware serial* "*Tx0*" dan "*Rx*0". Data yang telah terbaca oleh mikrokontroller *ESP8266* lalu diteruskan menuju ke *server* melalui koneksi *wifi*. Koneksi *wifi* merupakan fitur standar dari mikrokontroller *ESP8266*.



Gambar 3 Diagram Koneksi pada *Gateway*

Gambar 4 menunjukkan pemasangan *gateway*. Lokasi *gateway* dipasang sedekat mungkin dengan lokasi *power meter*. Hal ini dilakukan unutk menghindari *noise* yang kemungkinan terjadi pada komunikasi serial RS485 menuju ke *power meter*. Lokasi pemasangan juga harus terjangkau oleh jaringan *wifi* agar data yang diterima dari *power meter* dapat diteruskan menuju ke *server*.

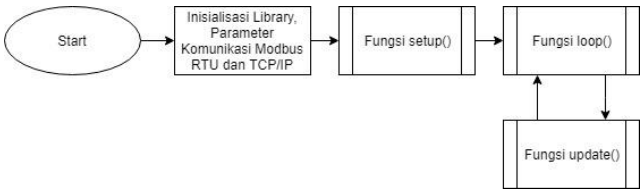
ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551



Gambar 4 Pemasangan *gateway* pada *power meter*

Pemrograman pada *gateway* dilakukan dengan *Software Arduino IDE* dengan tambahan *library SimpleModbusMaster*, *ModbusTCPSlave*, *Ticker*, dan *ESP8266WiFi*. Kode program yang sudah dibuat selanjutnya dilakukan proses kompilasi lalu program yang sudah terkompilasi dilakukan proses *upload* menuju ke mikrokontroller *ESP8266*.

Pada Gambar 5 alur jalan program diawali dengan inisialisasi. Proses inisialisasi diawali dengan inisialisasi *library* yang bertujuan untuk memasukkan *library* tambahan yaitu *library SimpleModbusMaster*, *ModbusTCPSlave*, *Ticker*, dan *ESP8266WiFi*. Setelah *library* ditambahkan, dilakukan inisialisasi komunikasi *Modbus RTU* dimana dalam proses inisialisasi ini dimasukkan parameter yang dibutuhkan untuk *library SimpleModbusMaster* agar dapat melakukan komunikasi *Modbus* menuju *power meter* sebagai *Modbus RTU master*. Setelah proses inisialisasi *Modbus RTU* dilakukan proses inisialisasi *Modbus TCP/IP slave* dengan melakukan konfigurasi *ip address*, *default gateway* dan *subnet mask*. Setelah itu dilakukan proses deklarasi objek untuk koneksi *Modbus TCP/IP*. Dalam hal ini objek dideklarasikan dengan nama "Mb". Proses deklarasi ini memunginkan bagi objek "Mb" untuk menggunakan *method-method* yang terdapat pada *library ModbusTCPSlave*.

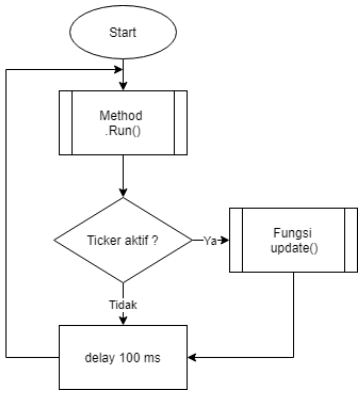


Gambar 5 *Flowchart* proses program utama pada *Arduino IDE*

Fungsi *setup()* merupakan fungsi dasar yang terdapat pada *Arduino IDE* yang dieksekusi sekali pada saat program berjalan. Fungsi ini berguna untuk melakukan inisialisasi variabel beserta fungsi-fungsi lain yang perlu diinisialisasi pada awal program berjalan. Fungsi *setup()* ini hanya dijalankan sekali saat mikrokontroller *ESP 8266* menyala.

Dalam fungsi *setup()* terdapat beberapa fungsi dari *library SimpleModbusMaster* yang digunakan untuk memulai proses inisialisasi komunikasi melalui protokol *Modbus RTU* menuju *power meter*.

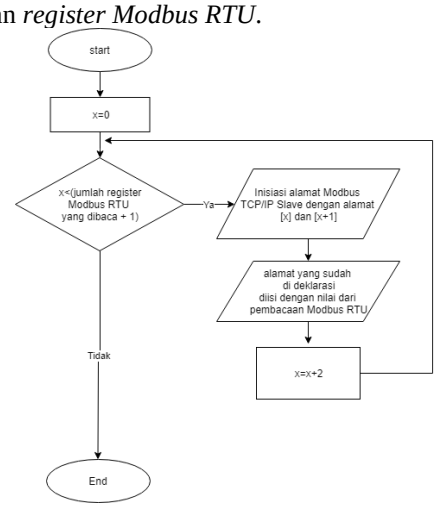
Fungsi *loop()* merupakan fungsi yang dieksekusi secara berulang selama mikrokontroller *ESP8266* menyala.



Gambar 6 *Flowchart* fungsi *loop()*

Di dalam fungsi *loop()* terdapat *method .Run()* yang merupakan *method* dari objek "Mb". *Method .Run()* berfungsi untuk meng*update* nilai dari *register* pada *Modbus TCP/IP Slave*. Setelah data terupdate, selanjutnya *method .attach_ms()* dijalankan. *Method .attach_ms()* merupakan *method* dari *library Ticker* yang digunakan untuk menjalankan fungsi *update()* setiap interval 20 milidetik sekali. Dengan *method* ini, maka fungsi *update()* yang berguna untuk meng*update* data *register Modbus TCP/IP* dapat dijalankan setiap 20 milidetik sekali. Sehingga nilai pada *register Modbus TCP/IP* dapat tetap sama dengan nilai dari data *register Modbus RTU*.

Fungsi *update()* merupakan fungsi yang dideklarasikan pada bagian awal dari program. Kegunaan dari fungsi ini adalah untuk melakukan inisialisasi alamat *Modbus TCP/IP Slave* dan mengisi alamat itu dengan nilai yang didapat dari pembacaan *register Modbus RTU*.



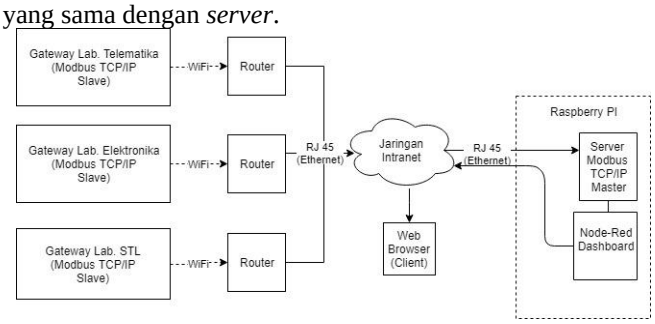
Gambar 7 *Flowchart* fungsi *update()*

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551

B. Desain Server

Perangkat *server* menggunakan sebuah *Raspberry Pi* yang adalah sebuah komputer berukuran kecil. *Server* terkoneksi pada jaringan *intranet* melalui *interface* dari *port ethernet* yang sudah terdapat pada *Raspberry Pi*.

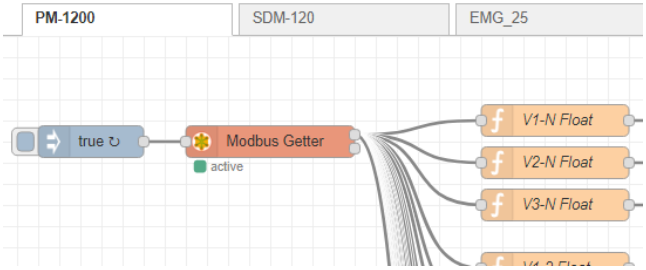
Pada gambar Gambar 8 koneksi pada *server* memanfaatkan jaringan *intranet* yang merupakan jaringan privat yang bersifat lokal. Jaringan ini menggunakan protokol *TCP/IP* sehingga masing-masing perangkat memerlukan pengenal berupa alamat IP. *Server raspberry PI*, terhubung menuju jaringan *intranet* melalui *interface* menggunakan *port ethernet* dengan konektor berjenis RJ 45. *Server* yang telah terhubung ke dalam jaringan *intranet*, dapat diakses melalui alamat IP yang telah di set sebelumnya atau yang telah diberikan oleh DHCP *server* dalam jaringan *intranet*. Antara *gateway* dengan *server* terdapat *router* yang memiliki dua fungsi. Fungsi pertama adalah sebagai *access point WiFi* yang digunakan oleh *gateway*. Fungsi kedua adalah sebagai perangkat *routing* atau perangkat penghubung agar koneksi perangkat yang terkoneksi melalui jaringan *wireless LAN* atau *WiFi* dapat terhubung kedalam jaringan *intranet*. *Client* merupakan pengguna yang juga terhubung menuju jaringan *intranet*. *Client* dapat mengakses halaman *web* yang memberikan tampilan status dari *power meter* melalui sebuah *web browser*. *Client* juga harus berada pada jaringan *intranet* yang sama dengan *server*.



Gambar 8 Skema komunikasi *gateway* menuju *server*

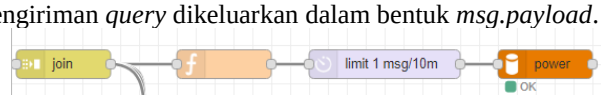
Node-Red berfungsi sebagai *platform* pemrograman dalam *server* ini. Di dalam *Node-Red* terdapat program untuk membaca data *Modbus TCP/IP* dari *gateway*, mengkonversi nilai dari *register* pada *gateway* menjadi nilai *float*, mengirim *SQL Query* menuju *database MySQL*, menampilkan nilai pembacaan dari *gateway* menuju halaman *web* yang disebut sebagai *Node-Red dashboard*, mekanisme pengiriman notifikasi menuju *smartphone*, dan menampilkan isi *database* ke dalam bentuk grafik.

Pembacaan data dari *gateway* dilakukan dengan sebuah *node* pada *Node-Red* yang bernama "*node-red-contrib-modbus*" (Gambar 9). *Node* ini dapat membaca data *Modbus* dalam bentuk serial ataupun *TCP/IP*. Dalam penelitian ini konfigurasi yang digunakan adalah untuk membaca data dari *Modbus TCP/IP*.



Gambar 9 *Node Modbus*

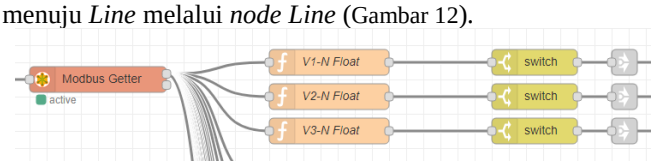
Pengiriman *query* menuju *SQL server* dilakukan menggunakan *node* yang bernama *mysql* (Gambar 10). Dalam *node mysql*, *query* yang dikirimkan merupakan *input* yang didapat dari *msg.topic*. Data dari *msg.topic* dibaca dan dikirimkan menuju *SQL server*. Output dari *node mysql* adalah hasil dari *query* yang telah dikirimkan. Hasil dari pengiriman *query* dikeluarkan dalam bentuk *msg.payload*.



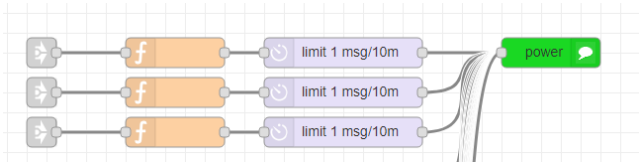
Gambar 10 Proses memasukkan data menuju *database*

Notifikasi menuju *smartphone* dikirim menuju aplikasi *chatting "Line"* yang sebelumnya telah terinstall pada *smartphone*. Notifikasi memberitahu kepada pengguna apabila terjadi pembacaan nilai dari *power meter* yang berada diluar nilai *threshold*.

Proses pengiriman notifikasi dilakukan dengan membandingkan nilai dari *node modbus* dengan nilai *threshold* yang berasal dari *database*. *Flow* dari proses ini terdapat pada Gambar 11. Proses perbandingan ini dilakukan menggunakan *node* yang bernama *node switch*. Di dalam *node switch*, nilai antara data yang terbaca melalui *node modbus* dilakukan perbandingan dengan nilai *threshold* yang dibaca dari *database*. Jika nilai yang didapati melampaui dari *threshold* yang ditentukan, maka *server* akan mengirim pesan menuju *Line* melalui *node Line* (Gambar 12).



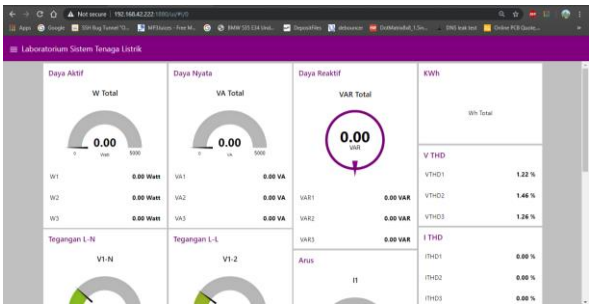
Gambar 11 Perbandingan nilai real dengan nilai *threshold*



Gambar 12 *Node Line*

Pada Gambar 13 merupakan tampilan *dashboard* pada *Node-Red*. Dalam *Node-Red dashboaord* terdpat tampilan dari parameter – parameter dari masing – masing *power meter*. Selain itu juga teradapat halaman untuk melakukan pengaturan pada nilai *threshold*.

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551



Gambar 13 Tampilan *dashboard Node-Red*

III. Hasil dan Pembahasan

Berikut mupakan hasil dari penelitian yang telah dilakukan. Hasil dari penelitiasn dijelaskan pada beberapa bagian berikut :

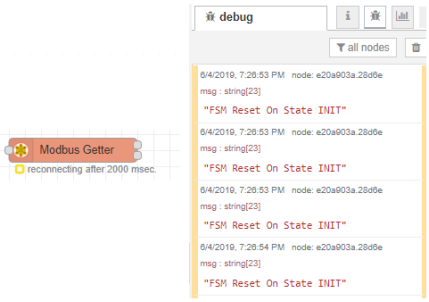
A. Pembacaan Data Modbus RTU

Pembacaan dilakuakn dengan melihat *debug* data dengan memanfaatkan pin *software serial* yang menggunakan pin 14 dan pin 12 sebagai pin *Tx* dan *Rx*. Data serial ini dikonversi menjadi data dalam bentuk *USB* agar dapat ditampilkan melalui *serial monitor* pada *software Arduino IDE*. Proses konversi data *serial* menggunakan perangkat *USB to TTL converter*.

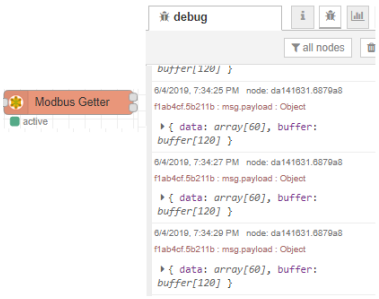
Hasil dari *debug* data yang telah dilakukan, terdapat kegagalan *request* sebagnyak 2 data dari 50 data yang di *request* pada *power meter PM-1200*. Pada *power meter* lainnya tidak terdapat kegagalan *request*.

B. Pembacaan Data Modbus TCP/IP

Hasil komunikasi dilihat dengan memantau *debug* data pada *node modbus* pada *Node-Red*. *Debug* ditampilkan pada *tab debug*. Pada *tab debug* akan tercatat *request* yang dilakukan oleh *Node modbus* pada *Node-Red* menuju *gateway*. *Request* yang berhasil diterima oleh *gateway* akan ditanggapi oleh *gateway* dengan mengirim data yang berisi nilai dari pembacaan *power meter*. Data yang diterima akan masuk menuju *debug* (**Error! Reference source not found.**Gambar 14). Jika *request* gagal, maka *node modbus* akan memberikan keluaran *debug* berupa peringatan *exception* (Gambar 15). Percobaan dilakukan sebanyak 50 kali *request*.



Gambar 14 *Debug* yang didapatkan saat *request* gagal



Gambar 15 *Debug* yang didapatkan saat *request* berhasil

Hasil dari ketiga percobaan yang telah dilakukan pada masing - masing *gateway* dapat diketahui tidak terdapat kegagalan *request.* Semua *request* yang diterima dapat diterima dengan baik dan dibalas dengan nilai dari masing – masing *power meter*.

C. Waktu Pengiriman Data

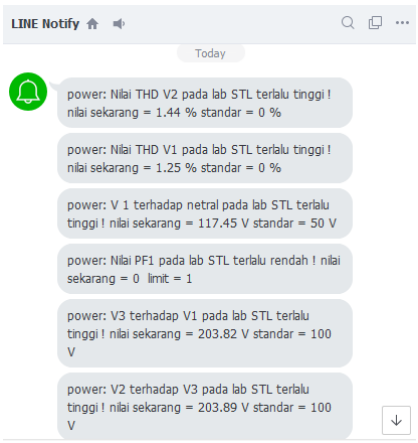
Hasil waktu pengirian data dapat dikatehui dengan melakukan pencatatan data *timestamp*. Data *timestamp* dicatat agar selisih jeda waktu dapat diketahui dengan tepat. Terdapat dua jenis data *timestamp* yang dicatat. Data timestamp yang pertama adalah data yang dicatat pada *gateway* saat melakukan *request* data dan pembacaan data dari *power meter.* Data *timestamp* kedua dicatat pada *server* saat melakukan pembacaan dan *request* data menuju *gateway*.

Dari tiga percobaan yang telah dilakukan, terdapat jeda waktu pengiriman data dari *power* menuju ke *gateway* dengan jeda waktu terbesar adalah 61.29 milidetik dan terkecil adalah 55.99 milidetik. Pada jeda waktu dari *gateway* menuju ke *server*, jeda waktu tercepat untuk pengujian tanpa beban *traffic* sebesar 58.1 milidetik dan terlama sebesar 118.6 milidetik. Untuk pengujian dengan beban *traffic* data, jeda waktu terlama adalah 134.2 milidetik dan jeda waktu tercepat adalah 70.1 milidetik. Total jeda waktu pengiriman data dari *power meter* menuju ke *server* adalah penjumlahan dari total jeda waktu dari *power meter* menuju ke *gateway* dan jeda waktu antara *gateway* menuju *server* ditambah dengan 750 milidetik. Penjumlahan dengan 750 milidetik merupakan jumlah total jeda waktu yang terdapat pada *library* dan jeda waktu *pooling server* menuju ke *gateway*. Sehingga didapati jeda waktu keseluruhan tercepat adalah sebesar 864.09 milidetik dan terlama sebesar 943.35 milidetik.

D. Notifikasi

Metode pengujian dilakukan dengan mengeset *threshold* pengaturan di bawah kondisi real pembacaan pada *power meter*. Data dari *power meter* yang masuk pada *server melalui gateway* akan dianggap melampaui *threshold* karena *threshold* yang di set lebih rendah dari kondisi real. Kondisi ini akan membuat notifikasi dari semua parameter yang ada akan terkirim menuju *smartphone* setiap notifikasi yang masuk akan di cek apakah mewakili data dari masing – masing parameter (Gambar 16).

ISSN (Print) : 2621-3540
ISSN (Online) : 2621-5551



Gambar 16 Notifikasi yang masuk pada aplikasi *Line*

Dari semua pengujian yang telah dilakukan pada masing – masing laboratorium, semua status notifikasi menunjukkan status terkirim dan tidak ada parameter yang gagal terkirim. Semua notifikasi dari parameter yang diujikan pada masing – masing Laboratorium dapat terkirim menuju *Line*.

IV. Kesimpulan

1. *Gateway* yang masing – masing terletak di Laboratorium Sistem Tenaga Listrik, Elektronika, dan Telematika dapat membaca data dari *power meter EMG 25*, *SDM 120*, *PM-1200* dan meneruskannya menuju *server*.
2. Terdapat jeda waktu pembacaan antara data yang dibaca oleh *power meter* dengan data yang tampil pada *dashboard* pada *server*. Jeda waktu bervariasi antara 864.09 milidetik hingga 943.35 milidetik. Jeda waktu menunjukkan respon yang lebih lambat saat pada *server* terdapat *traffic* data.
3. Dari keseluruhan pengujian yang telah dilakukan, keseluruhan sistem dapat bekerja dengan baik dalam membaca data dari *power meter*, menampilkan kepada pengguna melalui halaman *dashboard*, dan memberi notifikasi menuju ke *smartphone* pengguna.

Daftar Pustaka

[1] M. P. Panuntun, “PENGUJIAN KETELITIAN KWH METER ANALOG DAN KWH METER,” 2018.

[2] M. Ridho and R. Zuhri, “Perbandingan Akurasi Kwh Meter Digital Dan Kwh Meter,” 2017.

[3] M. Rovianto, B. Rahmat, and A. Rizal, “Desain dan realisasi sistem telemetri fsk (suhu, tekanan udara, kelembaban),” *Proceeding Semin. Nas. Pendidik. Tek. Elektro 2005*, 2005.

[4] “Lolin NodeMCU ESP8266 WIFI Serial Wireless Module.” [Online]. Available: https://www.amazon.in/Lolin-NodeMCU-ESP8266-CP2102-Wireless/dp/B010O1G1ES. [Accessed: 11-Jun-2019].