DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT* (RTU) SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER

SKRIPSI

Oleh :

Nabil Bachroin

171910201114

JURUSAN TEKNIK ELEKTRO STRATA 1

FAKULTAS TEKNIK

UNIVERSITAS JEMBER

2021

DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT* (RTU) SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER

SKRIPSI

Oleh :

Nabil Bachroin

171910201114

JURUSAN TEKNIK ELEKTRO STRATA 1

FAKULTAS TEKNIK

UNIVERSITAS JEMBER

2021

PERSEMBAHAN

Puji syukur kepada Allah SWT yang telah memberikan Rahmat dan Karunia-Nya, sehingga penulis dapat menyelesaikan skripsi ini. Penulis menyadari bahwa pengerjaan skripsi ini begitu banyak memperoleh semangat dari banyak pihak. Untuk itu penulis mempersembahkan ini kepada :

1. Kedua Orang Tua saya yaitu Ayahanda Agus Subandriyo dan Ibunda Mufidah tercinta, terimakasih atas semua pengorbanan, usaha, jerih payah, dorongan, nasehat, dan air mata yang menetes dalam untaian doa yang selalu mengiringi setiap langkah perjuangan serta keberhasilan penulis;
2. Bapak Dosen Pembimbing Utama Dr. Azmi Saleh, ST., MT serta Bapak Dosen Pembimbing Anggota Suprihadi Prasetyono, ST., MT atas keikhlasan dan kesabarannya dalam membimbing saya menyelesaikan skripsi ini;
3. Seluruh dosen Teknik Elektro Universitas Jember yang telah membekali ilmu pengetahuan dan bimbingan selama mengikuti pendidikan di Jurusan Teknik Elektro Fakultas Teknik Universitas Jember;
4. Guru - guru saya sejak taman kanak - kanak sampai perguruan tinggi;
5. Saudara kandung saya Nuriza Izroatus Sundus yang telah memberi semangat dan motivasi untuk terus berjuang dalam menyelesaikan tugas kuliah;
6. Almamaterku Universitas Jember yang saya cintai dan saya banggakan;
7. Keluarga Besar Organisasi Mahasiswa Robotika Fakultas Teknik Universitas Jember sebagai wadah penulis dalam memperoleh ilmu dan wawasan keorganisasian, dan mendapatkan keluarga baru dalam mengemban tugas bersama;
8. Asisten Laboratorium Sistem Tenaga Syahrijal Raja Jaya, Nabila Dinda Rahmania, dan Indra Sugma Widayanto yang telah memberi pengalaman yang sangat luar biasa dan tidak akan saya lupakan;
9. Sahabat tercintaku yang selalu ada dalam segala keadaan dan selalu memberikan semangat yaitu Nabila Dinda Rahmania;

10. Seluruh Dulur Elektro Universitas Jember angkatan 2017 yang sangat membantu saya dan menemani perjalanan masa perkuliahan yang selalu memberikan dukungan dan doa.

MOTTO

Maka sesungguhnya bersama kesulitan ada kemudahan.

(terjemahan Surat *Al - Insyirah* ayat 5)[1)]

atau

Saya sukses, karena saya telah kehabisan apa yang disebut dengan kegagalan.[2)]

atau

Tiada suatu usaha yang besar akan berhasil tanpa dimulai dari usaha yang kecil.[3)]

[1)] Departemen Agama Republik Indonesia. 1998. Al Qur'an dan Terjemahannya. Jember: Universitas Jember.

[2)] Thomas Alfa Edison

[3)] Joeniarto, 1967 dalam Mulyono, E. 1998. *Beberapa Permasalahan Implementasi Konvensi Keanekaragaman Hayati dalam Pengelolaan Taman Nasional Meru Betiri*. Tesis Magister Universitas Jember, tidak dipublikasikan.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Nabil Bachroin

NIM : 171910201114

Menyatakan bahwa karya ilmiah yang berjudul: "DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT (RTU)* SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER" adalah benar-benar karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya dan belum pernah diajukan pada institusi mana pun, serta bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata dikemudian hari pernyataan ini tidak benar.

Jember, 02 Desember 2021

Yang menyatakan,

Nabil Bachroin

171910201114

SKRIPSI

DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT (RTU)* SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER

Oleh
Nabil Bachroin
NIM 171910201114

Pembimbing

Dosen Pembimbing Utama : Dr. Azmi Saleh, S.T., M.T
Dosen Pembimbing Anggota : Suprihadi Prasetyono, S.T., M.T

PENGESAHAN

Skripsi berjudul "DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT (RTU)* SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER" karya Nabil Bachroin telah diuji dan disahkan pada:

hari, tanggal : Kamis, 02 Desember 2021

tempat : Fakultas Teknik Universitas Jember

Tim Penguji:

Pembimbing I,

Dr. Azmi Saleh, S.T., M.T

NIP 197106141997021001

Pembimbing II,

Suprihadi Prasetyono, S.T., M.T

NIP 197004041996011001

Penguji I,

Dr. Ir. Widjonarko, S.T., M.T.

NIP 197109081999031001

Penguji II,

Guido Dias Kalandro, S.ST., M.Eng.

NRP 760015734

Mengesahkan

Dekan,

Dr. Ir. Triwahju Hardianto, S.T., M.T.

NIP 197008261997021001

RINGKASAN

DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT (RTU)* SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER; Nabil Bachroin; 171910201114; 2021; 166 halaman; Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.

Bersamaan dengan pesatnya perkembangan teknologi di dunia maka diperlukan suatu sistem yang dapat mengontrol, mengawasi, dan mengumpulkan data pabrik. Dalam dunia industri, sistem SCADA (*Supervisory Control and Data Acquisition*) telah banyak digunakan terutama di bidang otomasi. Namun bagi masyarakat luas khususnya bagi kalangan mahasiswa, sistem SCADA masih kurang dikenal. Sebuah sistem SCADA dapat dibangun dengan berbagai macam cara komunikasi. Protokol komunikasi yang sering digunakan pada umumnya yaitu Modbus Serial atau Modbus RTU, Modbus TCP/IP yang masih membutuhkan kabel untuk komunikasi tersebut, seperti kabel RS485 maupun kabel LAN RJ45. Sehingga perlu adanya inovasi perancangan *Remote Terminal Unit* (RTU) dengan metode yang digunakan menggunakan protokol Modbus TCP/IP berbasis *wireless* yang sekaligus diimplementasikan pada Gedung B Fakultas Teknik Universitas Jember untuk melengkapi sistem *monitoring* dan kontrol pada Gedung tersebut. Untuk mendapatkan keakuratan dan kepresisian dalam pembacaan tegangan, arus, cos phi, dan daya maka digunakan sensor PZEM-004T dan YHDC SCT-013. Sistem minimum yang digunakan sebagai RTU dirancang dengan dua mikrokontroler yang diletakkan dalam satu board PCB untuk mendapatkan desain yang menarik dan mempunyai ukuran yang kecil. Pada pengujian performa sistem RTU yang telah dibangun menunjukkan performa yang cukup baik saat melakukan *monitoring* maupun pengontrolan terhadap komponen yang terkoneksi dengan perangkat RTU. Hasil pengujian jarak komunikasi secara *wireless* dari jarak 10 meter hingga 100 meter menunjukkan hasil yang lancar. Rata-rata dari eror pembacaan pada pengujian *monitoring* sebesar 1,37%.

SUMMARY

DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT (RTU)* SISTEM SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER; Nabil Bachroin; 171910201114; 2021; 166 pages; Department of Electrical Engineering, Faculty of Engineering, University of Jember.

Along with the rapid development of technology in the world, it is necessary to have a system that can control, monitor, and collect factory data. In the industrial world, the SCADA (Supervisory Control and Data Acquisition) system has been widely used, especially in the field of automation. However, for the wider community, especially for students, the SCADA system is still not well known. A SCADA system can be built in various ways of communication. Communication protocols that are often used in general are Modbus Serial or Modbus RTU, Modbus TCP / IP which still requires cables for communication, such as RS485 cables and RJ45 LAN cables. So there is a need for an innovative Remote Terminal Unit (RTU) design with the method used using the wireless-based Modbus TCP/IP protocol which is also implemented in Building B, Faculty of Engineering, University of Jember to complete the monitoring and control system in the building. To get accuracy and precision in reading voltage, current, cos phi, and power, PZEM-004T and YHDC SCT-013 sensors are used. The minimum system used as an RTU is designed with two microcontrollers placed on one PCB board to get an attractive design and has a small size. In testing the performance of the RTU system that has been built, it shows a fairly good performance when monitoring and controlling the components connected to the RTU device. The results of testing the distance of wireless communication from a distance of 10 meters to 100 meters show smooth results. The average reading error on the monitoring test is 1.37%.

PRAKATA

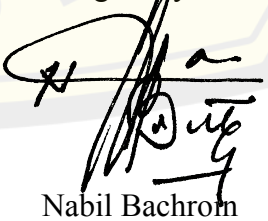
Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi berjudul "DESAIN DAN IMPLEMENTASI *REMOTE TERMINAL UNIT (RTU)* SISTEM SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Teknik Elektro Fakultas Teknik Universitas Jember. Penyusunan skripsi ini tidak lepas dari bantuan beberapa pihak. Oleh karena itu, penulis menyampaikan terimakasih kepada :

1. Bapak Dr. Triwahju Hardianto, S.T., M.T. selaku Dekan Fakultas Teknik Universitas Jember;
2. Bapak Dr. Bambang Sri Kaloko, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Jember;
3. Bapak Dr. Azmi Saleh, S.T., M.T selaku Dosen Pembimbing Utama serta Suprihadi Prasetyono, ST., MT selaku Dosen Pembimbing Anggota;
4. Bapak Dr. Ir. Widjonarko, S.T., M.T. selaku Penguji 1 serta Bapak Guido Dias Kalandro, S.ST., M.Eng. selaku Penguji 2;
5. Bapak Tama selaku Dosen Pembimbing Akademik;
6. Bapak Agus Subandriyo dan Ibu Mufidah selaku kedua orang tua penulis;
7. Semua pihak yang tidak dapat disebutkan satu per satu;

Penulis menyadari bahwa skripsi ini jauh dari kesempurnaan, untuk itu penulis menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini, terimakasih.

Jember, 02 Desember 2021

Yang menyatakan,



Nabil Bachrom

171910201114

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Bersamaan dengan pesatnya perkembangan teknologi di dunia maka diperlukan suatu sistem yang dapat mengontrol, mengawasi, dan mengumpulkan data pabrik. Dalam dunia industri, sistem SCADA (*Supervisory Control and Data Acquisition*) telah banyak digunakan terutama di bidang otomasi. Namun bagi masyarakat luas khususnya bagi kalangan mahasiswa, sistem SCADA masih kurang dikenal, walaupun dunia industri pada era sekarang ini sangat membutuhkan sistem tersebut untuk meningkatkan efisiensi dan efektifitas serta mengurangi beban kerja pekerja.

Salah satu cara untuk memperkenalkan dengan baik mengenai sistem tersebut kepada mahasiswa yaitu dengan adanya pembelajaran secara langsung pada alatnya, namun peralatan dan modul pembelajaran pada jurusan Teknik Elektro Universitas Jember masih belum memadai. Adapun 2 modul praktikum yang tersedia pada Laboratorium Sistem Tenaga, akan tetapi 1 modul tersebut masih belum bisa digunakan untuk keperluan praktikum dengan baik dan 1 modul lainnya sudah bisa digunakan untuk praktikum hanya saja modul yang telah dibuat oleh saudara Fahim Syah Reza Pahlevi mempunyai ukuran yang cukup besar sehingga kurang efektif.

Sebuah sistem SCADA dapat dibangun dengan berbagai macam cara komunikasi. Protokol komunikasi yang sering digunakan pada umumnya yaitu Modbus Serial atau Modbus RTU, sedangkan protokol komunikasi yang digunakan pada modul praktikum SCADA yang telah dibuat oleh saudara Fahim Syah Reza Pahlevi sudah menggunakan Modbus TCP/IP. Komunikasi tersebut disalurkan melalui kabel LAN RJ45. Akan tetapi komunikasi tersebut masih memiliki kekurangan seperti penggunaan kabel yang terkadang membuat terlihat tidak rapi dan keterbatasan jarak antara modul praktikum dengan komputer atau laptop yang digunakan untuk kontrol dan monitoring.

Berdasarkan masalah yang telah dipaparkan diatas maka penulis berinisiatif membuat tugas akhir dengan judul “DESAIN DAN IMPLEMENTASI

REMOTE TERMINAL UNIT (RTU) SCADA BERBASIS *WIRELESS* PADA GEDUNG B FAKULTAS TEKNIK UNIVERSITAS JEMBER" yang diharapkan mampu membantu proses pembelajaran praktikum di Laboratorium Sistem Tenaga, menambah popularitas sistem SCADA, menambah efektifitas dalam penerapan alat dengan penggunaan protokol Modbus TCP/IP berbasis wireless, serta uji coba pengimplementasian monitoring dan kontrol secara *real time* pada lantai 3 Gedung B Fakultas Teknik Universitas Jember yang didasari oleh adanya kesulitan dalam penyelesaian masalah kelistrikan pada gedung tersebut yang dikarenakan belum adanya sistem yang mampu melakukan monitoring dan pengontrolan yang baik.

1.2 Rumusan Masalah

Adapun beberapa rumusan masalah yang diperoleh berdasarkan latar belakang dan fokus masalah yang telah diuraikan antara lain :

1. Bagaimana mendesain dan merancang RTU sistem SCADA berbasis *wireless*?
2. Bagaimana kinerja dari RTU dengan protokol modbus TCP/IP berbasis *wireless*?

1.3 Batasan Masalah

Dalam proses penyelesaian masalah tersebut pasti ada masalah yang terjadi, agar pembahasan masalah tidak meluas, maka diberikan batasan-batasan masalah sebagai berikut :

1. Komunikasi yang digunakan adalah modbus TCP/IP dengan jarak maksimal komunikasi data dari RTU ke MTU tergantung pada kekuatan sinyal *wireless*;
2. Jumlah sensor yang digunakan adalah 8 yang terdiri dari 1 buah sensor PZEM-004T dan 7 sensor arus, sedangkan besaran yang dimonitoring hanya disertai dengan daya aktif ;
3. Tidak membahas software HMI Wonderware Intouch lebih dalam, software HMI utama yang digunakan sebagai pengujian alat adalah Modscan karena penulis berfokus pada hardware dan HMI Modscan sedangkan perancangan HMI Wonderware Intouch dikerjakan oleh Nabila Dinda Rahmania.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian tugas akhir ini adalah :

1. Dapat mendesain dan mengimplementasikan modul RTU dengan mikrokontroler berbasis *wireless*;
2. Dapat memberikan informasi beban pada MTU dengan protokol komunikasi Modbus TCP/IP secara *real time*;
3. Dapat memanfaatkan *wireless* untuk mengatasi penggunaan kabel sebagai jalur komunikasi sistem SCADA yang sering kali menjadi masalah jarak penempatan MTU dengan RTU menjadi berbasis *wireless*.

1.5 Manfaat penelitian

Hasil dari penelitian tugas akhir ini diharapkan mampu memberikan manfaat antara lain :

1. Bagi penulis diharapkan penelitian ini menjadi sarana untuk menerapkan sedikit ilmu yang telah diperoleh dan menjadi sarana untuk berlatih tanggung jawab;
2. Bagi mahasiswa lainnya diharapkan penelitian ini dapat digunakan sebagai referensi dan pembelajaran dalam mengembangkan ilmu pengetahuan dan teknologi;
3. Bagi teknisi gedung dapat mempermudah dalam pemantauan dan pengontrolan sistem kelistrikannya;
4. Mampu memberikan popularitas sistem SCADA khususnya dengan protokol TCP/IP berbasis *wireless*.

1.6 Sistematika Penulisan

Untuk mempermudah dalam penulisan tugas akhir ini, maka penulis membuat penataan penulisan sebagai berikut :

BAB 1. PENDAHULUAN

Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan

BAB 2. TINJAUAN PUSTAKA DAN LANDASAN TEORI

Berisi tentang uraian yang berkaitan dengan setiap komponen yang digunakan ataupun hasil penelitian terdahulu yang berkaitan dengan penelitian yang akan dilakukan.

BAB 3. METODOLOGI PENELITIAN

Berisi tentang paparan metode yang digunakan untuk menyelesaikan penelitian.

BAB 4. HASIL DAN PEMBAHASAN

Berisi hasil penelitian dan analisis hasil penelitian.

BAB 5. PENUTUP

Berisi kesimpulan dan saran dari penulis.

DAFTAR PUSTAKA

Berisi paparan referensi-referensi yang digunakan oleh penulis sebagai acuan, penunjang serta parameter yang mendukung terselesaikannya tugas akhir ini.

LAMPIRAN

Berisi gambar, tabel, maupun listing program yang menunjang proses penyelesaian tugas akhir ini.

BAB 2.TINJAUAN PUSTAKA DAN LANDASAN TEORI

Pada bab ini akan menguraikan landasan teori hal-hal yang dibutuhkan dalam penelitian serta membahas tinjauan pustaka yang berasal dari jurnal ataupun laporan tugas akhir yang telah ada yang digunakan sebagai referensi dalam proses penelitian.

2.1 Tinjauan Pustaka

Pada penelitian lain yang bersangkutan dengan penelitian ini adalah tugas akhir Universitas Jember 2020, oleh Fahim Syah Reza Pahlevi yang berjudul "RANCANG BANGUN HUMAN MACHINE INTERFACE (HMI) PADA SISTEM MONITORING DAN KONTROL KELISTRIKAN GEDUNG PERKANTORAN". Penelitian tersebut berfokus pada desain HMI yang diimplementasikan untuk melakukan pengontrolan serta pengawasan sistem kelistrikan pada gedung bertingkat. Namun pada penelitian tersebut masih menggunakan komunikasi berbasis kabel melalui kabel RJ-45.

Penelitian selanjutnya yang mendasari penelitian saat ini yaitu tugas akhir Politeknik Negeri Bandung, 2017, oleh Arvanida Feizal Permana yang berjudul "PINTU PEMBERITAHU KEGIATAN RUANGAN MENGGUNAKAN HMI SCADA BERBASIS MODUL MIKROKONTROLER (HARDWARE SISTEM ALARM DAN KUNCI OTOMATIS)". Penelitian tersebut berfokus pada desain RTU yang digunakan hanya untuk pengawasan pemakaian ruangan laboratorium yang sering terjadi selisih antara satu kegiatan dengan kegiatan yang lain dan masih belum bisa digunakan untuk pengontrolan. Namun mikrokontroler pada penelitian tersebut menggunakan NodeMCU sehingga ukuran dari hardware cukup besar dan sensor analog yang dapat digunakan hanya 1 pin dimana pada penelitian tersebut digunakan untuk sensor cahaya.

Pada penelitian ini penulis berfokus pada pengembangan sistem komunikasi, penambahan sistem pengontrolan dari MTU, dan desain hardware. Perubahan yang dilakukan yaitu merubah perantara komunikasi dari yang

awalnya menggunakan kabel menjadi wireless dan pengembangan pada desain hardware dengan merancang sebuah sistem minimum sendiri, sehingga dapat mewujudkan sebuah remote terminal unit yang cukup kecil dan dapat diimplementasikan pada sistem yang cukup kompleks dengan 8 sensor ataupun dapat pula digunakan hingga 16 sensor dengan menggunakan protokol komunikasi modbus TCP/IP. Penambahan sistem pengontrolan pada penelitian ini akan membuat hidup atau nyala beban dapat dikontrol oleh MTU.

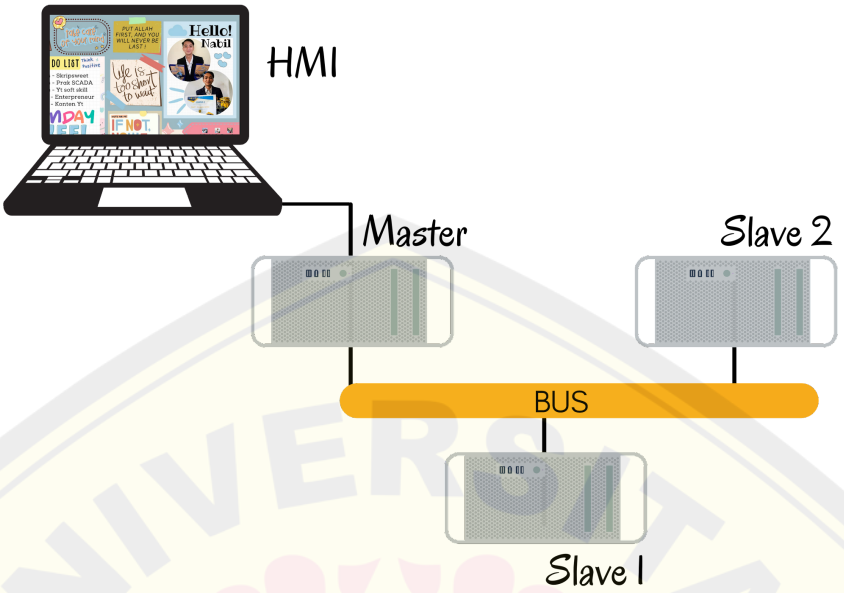
2.2 Landasan Teori

Beberapa teori yang mendukung dalam penyelesaian penelitian ini dijelaskan seperti berikut.

2.2.1 SCADA

Supervisory control and data acquisition (SCADA) adalah arsitektur sistem kontrol yang mencakup komputer, komunikasi data jaringan dan antarmuka pengguna grafis atau graphical user interfaces (GUI) untuk pemantauan dan manajemen proses lanjutan, serta perangkat periferal lainnya, seperti programmable logic controllers (PLC) atau dilengkapi dengan antarmuka pengontrol proporsional integral derivatif (PID) diskrit dengan pabrik atau mesin pemrosesan. Penggunaan SCADA pada umumnya juga dipertimbangkan untuk pengelolaan dan pengoperasian proses yang digerakkan oleh proyek dalam konstruksi.

Sistem komputer SCADA memproses antarmuka operator yang dapat memantau dan mengeluarkan perintah proses (seperti perubahan set point). Operasi tambahan, seperti logika kontrol waktu nyata atau kalkulasi pengontrol, dijalankan oleh modul jaringan yang terhubung ke sensor dan aktuator lapangan.



Gambar 2.1 Arsitektur SCADA

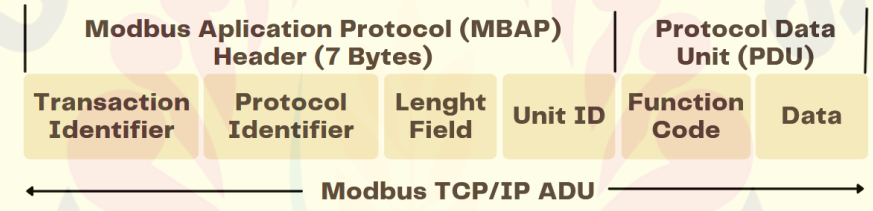
Pengembangan konsep SCADA yaitu metode universal untuk mengakses jarak jauh berbagai modul kontrol lokal, modul kontrol lokal ini mungkin berasal dari produsen yang berbeda dan memungkinkan akses melalui protokol otomatisasi standar. Faktanya, sistem SCADA skala besar telah menjadi sangat mirip fungsinya dengan sistem kontrol terdistribusi, menggunakan beberapa metode untuk berinteraksi dengan pabrik pada saat yang bersamaan. Mereka dapat mengontrol proses skala besar yang mungkin mencakup banyak situs, dan dapat bekerja pada jarak jauh dan pendek. Meskipun orang khawatir bahwa sistem SCADA rentan terhadap serangan *cyber war*/ *cyber terrorism*, sistem ini masih menjadi sistem kontrol industri yang paling umum digunakan.

Sistem SCADA memungkinkan operator di lokasi pusat proses yang didistribusikan secara luas, seperti ladang minyak atau gas, sistem pipa, sistem irigasi atau kompleks pembangkit listrik tenaga air, untuk membuat perubahan *set point* pada pengontrol proses yang jauh, untuk membuka atau menutup katup atau sakelar , untuk memantau alarm, dan untuk mengumpulkan informasi pengukuran. Ketika dimensi proses menjadi sangat besar - ratusan atau bahkan

ribuan kilometer dari satu ujung ke ujung lainnya - orang dapat menghargai manfaat yang ditawarkan SCADA dalam hal mengurangi biaya kunjungan rutin untuk memantau pengoperasian fasilitas. Nilai manfaat ini akan semakin berkembang jika fasilitasnya sangat terpencil dan memerlukan upaya ekstrim untuk berkunjung.

2.2.2 Modbus TCP/IP

Modbus merupakan tata cara komunikasi yang diaplikasikan ke PLC. Modbus dapat digunakan untuk berkomunikasi antar beberapa perangkat dalam satu jaringan, dalam contoh kasus sebuah sistem yang mengukur arus kemudian hasilnya dikirim ke komputer. Dalam penerapannya pada umumnya modbus digunakan untuk menghubungkan perangkat sebagai pemantau atau MTU dengan RTU pada sistem SCADA.



Gambar 2.2 *Frame* Data Modbus TCP/IP

Modbus mempunyai beberapa jenis antara lain Modbus ASCII, Modbus RTU, dan Modbus TCP/IP. Modbus RTU adalah salah satu jenis modbus yang singkat serta digunakan pada komunikasi serial. Modbus RTU inilah yang paling sering digunakan dalam implementasi protokol modbus. Modbus ASCII merupakan modbus yang dipakai pada komunikasi serial yang memanfaatkan karakter ASCII. Modbus TCP/IP adalah salah satu tipe modbus yang digunakan pada jaringan *Transmission Control Protocol/Internet Protocol* (TCP/IP).

2.2.3 LCD Oled

Organic Light Emitting Diode (OLED) merupakan semikonduktor yang memancarkan cahaya dengan berbahan dasar lapisan organik. *Organic Light Emitting Diode* (OLED) dimanfaatkan dalam teknologi elektroluminensi,

misalnya pada display atau tampilan layar. OLED terkenal fleksibel karena ketipisannya yang kurang dari 1 mm. Warna dari cahaya OLED bermacam-macam, ada OLED yang hanya dapat memancarkan satu warna dan ada pula OLED yang mampu memancarkan beberapa warna.

Gambar 2.3 LCD OLED

(Sumber : https://www.diymore.cc/)

Kemampuan memancarkan beberapa warna ini diperoleh hanya dengan membedakan tegangan yang diberikan pada perangkat OLED sehingga memiliki kemungkinan untuk menjadi perangkat alternatif seperti teknologi pada tampilan layar datar beralaskan kristal cair.

2.2.4 Sensor YHDC SCT 013

Gambar 2.4 Sensor Arus YHDC SCT013

(Sumber : https://i2.wp.com/www.nyebarilmu.com/)

YHDC SCT 013 adalah sensor yang berfungsi untuk mengukur arus AC. Sensor ini bisa mengukur arus listrik di sebuah bangunan, dengan pemilihan nilai perbandingan yang tertulis pada sensor akan membuat nilai pembacaan semakin bagus. *Split-core Current Transformer* merupakan sensor arus dengan konsep cara pembacaan pada trafo arus yang dirancang untuk memperoleh nilai arus yang lebih kecil nilainya dari kedua sisi yaitu pada sisi sekunder, karena sisi primer nilainya lebih besar dari sisi sekunder sehingga untuk dilakukan pengukuran lebih aman pada sisi sekunder.

2.2.5 Sensor PZEM-004T CT Split Core

PZEM-004T merupakan modul elektronik yang dapat digunakan untuk berbagai macam pengukuran, seperti pengukuran arus, tegangan, daya, frekuensi, energi, maupun faktor daya. Karena kemampuan berbagai macam pembacaan besaran oleh sensor ini sensor PZEM-004T juga disebut sebagai sensor yang mempunyai fitur lengkap dengan pembacaan yang sangat akurat. Berdasarkan kemampuannya dan keakuratan pembacaan tersebut yang membuat sensor ini cukup laris dipasaran untuk digunakan pada proyek pengukuran daya pada suatu jaringan listrik seperti gedung ataupun rumah.

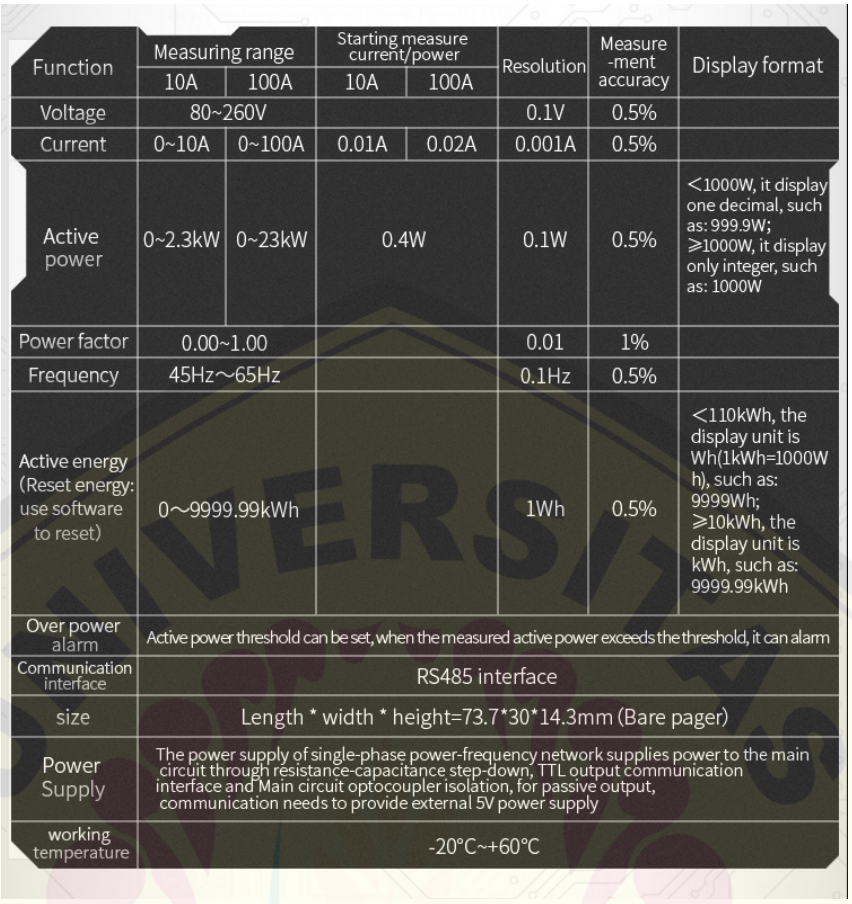
Gambar 2.5 Sensor PZEM-004T Split Core

(Sumber : https://www.nn-digital.com/2019/07/PZEM-004T-CT-Split-Core.jpg)

Tipe sensor PZEM yang digunakan pada penelitian ini yaitu tipe PZEM-004t yang merupakan versi keempat dari sensor PZEM. PZEM-004T mampu melakukan pengukuran arus hingga 100 ampere yang menggunakan *split core current transformer*. Karena menggunakan *split core* tentunya memiliki keunggulan tersendiri dalam kemudahan penggunaannya karena dapat langsung dipasangkan pada kabel jaringan listrik yang sudah terpasang tanpa harus melepas kabel tersebut. Meskipun terdapat beberapa perbedaan antara PZEM-004T V2.0 dan PZEM-004T V3.0, namun secara fungsional maupun fitur keduanya memiliki kemiripan. Berikut adalah fitur atau spesifikasi modul PZEM-004T :

A. Fungsi
    1. Fungsi pengukuran tegangan, arus, daya aktif, dll
    2. Tombol setel ulang energi (PZEM-004T V2.0)
    3. Fungsi penyimpanan data *power-down* (*power-down* kumulatif sebelum menyimpan)
    4. Komunikasi serial TTL
    5. Pengukuran daya : 0 ~ 9999 kW
    6. Pengukuran tegangan : 80 ~ 260 VAC
    7. Pengukuran arus : 0 ~ 100 A

B. Spesifikasi
    1. Tegangan kerja : 80 ~ 260 VAC
    2. *Rated power* : 100A/22000kW
    3. Frekuensi kerja : 46 - 65 Hz
    4. Akurasi pengukuran : 1.0



Function	Measuring range		Starting measure current/power		Resolution	Measure-ment accuracy	Display format
	10A	100A	10A	100A			
Voltage	80~260V				0.1V	0.5%	
Current	0~10A	0~100A	0.01A	0.02A	0.001A	0.5%	
Active power	0~2.3kW	0~23kW	0.4W		0.1W	0.5%	<1000W, it display one decimal, such as: 999.9W; ⩾1000W, it display only integer, such as: 1000W
Power factor	0.00~1.00				0.01	1%	
Frequency	45Hz～65Hz				0.1Hz	0.5%	
Active energy (Reset energy: use software to reset)	0～9999.99kWh				1Wh	0.5%	<110kWh, the display unit is Wh(1kWh=1000Wh), such as: 9999Wh; ⩾10kWh, the display unit is kWh, such as: 9999.99kWh
Over power alarm	Active power threshold can be set, when the measured active power exceeds the threshold, it can alarm						
Communication interface	RS485 interface						
size	Length * width * height=73.7*30*14.3mm (Bare pager)						
Power Supply	The power supply of single-phase power-frequency network supplies power to the main circuit through resistance-capacitance step-down, TTL output communication interface and Main circuit optocoupler isolation, for passive output, communication needs to provide external 5V power supply						
working temperature	-20°C~+60°C						

Gambar 2.6 Spesifikasi PZEM-004T

(Sumber : PZEM-004-specification-e1580011767396.jpg)

Berdasarkan gambar 2.6 Spesifikasi PZEM-004T tersebut, adapun beberapa informasi yang dapat dikerucutkan berdasarkan dari pembacaan masing-masing besaran oleh PZEM-004T sebagai berikut :

1. Tegangan :
   a. Rentang pengukuran : 80～260V
   b. Resolusi: 0.1V
   c. Akurasi pengukuran : 0.5%
2. Power factor :
   a. Rentang pengukuran : 0.00～1.00
   b. Resolusi: 0.01
   c. Akurasi pengukuran : 1%

3. Frekuensi :
   a. Rentang pengukuran : 45 Hz ～ 65 Hz
   b. Resolusi: 0.1 Hz
   c. Akurasi pengukuran : 0.5 %
4. Current :
   a. Rentang pengukuran : 0～100 A (PZEM-004T-100A)
   b. *Star*ting measure current: 0.02 A (PZEM-004T-100A)
   c. Resolusi: 0.001 A
   d. Akurasi pengukuran : 0.5 %
5. Active power :
   a. Rentang pengukuran : 0～2.3 kW (PZEM-004T-10A); 0～23 kW (PZEM-004T-100A)
   b. *Starting measure power*: 0.4 W
   c. Resolusi: 0.1 W
   d. Format tampilan:
      1000W,ditampilkan dalam desimal, seperti : 999.9W
      ≥1000W, hanya ditampilkan dalam integer, seperti : 1000W
   e. Akurasi pengukuran : 0.5 %
6. Active Energy :
   a. Rentang pengukuran : 0～9999.99 kWh
   b. Resolusi: 1 Wh
   c. Akurasi pengukuran : 0.5 %
   d. Format tampilan:
      10 kWh, *the display unit is* Wh(1kWh=1000Wh), seperti: 9999 Wh
      ≥10 kWh, *the display unit is* kWh, seperti: 9999.99 kWh

Modul PZEM-004T sangat mudah digunakan dalam pemrograman menggunakan berbagai jenis papan mikrokontroler seperti Arduino, ESP8266, STM32, WeMos, NodeMCU, Raspberry Pi dll karena menggunakan komunikasi serial TTL. Perlu diketahui bahwa protokol yang digunakan dalam PZEM-004T

V2.0 dan PZEM-004T V3.0 berbeda, sehingga pustaka dan pemrograman juga berbeda. Banyak orang mengira bahwa modul PZEM-004T rusak atau tidak berfungsi hanya karena mereka tidak tahu dan menggunakan pustaka yang salah.

No	Function	Head	Data1- Data5	Sum
1a	Voltage Req	B0	C0 A8 01 01 00 (Computer sends a request to read the voltage value)	1A
1b	Voltage Resp	A0	00 E6 02 00 00 (Meter reply the voltage value is 230.2V)	88
2a	Current Req	B1	C0 A8 01 01 00 (Computer sends a request to read the current value)	1B
2b	Current Resp	A1	00 11 20 00 00 (Meter reply the current value is 17.32A)	D2
3a	Active power Req	B2	C0 A8 01 01 00 (Computer sends a request to read the active power value)	1C
3b	Active power Resp	A2	08 98 00 00 00 (Meter reply the active power value is 2200w)	42
4a	Read energy Req	B3	C0 A8 01 01 00 (Computer sends a request to read the energy value)	1D
4b	Read energy Resp	A3	01 86 9f 00 00 (Meter reply the energy value is 99999wh)	C9
5a	Set the module address Req	B4	C0 A8 01 01 00 (Computer sends a request to set the address, the address is 192.168.1.1)	1E
5b	Set the module address resp	A4	00 00 00 00 00 (Meter reply the address was successfully set)	A4
6a	Set the power alarm threshold Req	B5	C0 A8 01 01 14 (computer sends a request to set a power alarm threshold)	33
6b	Set the power alarm threshold Resp	A5	00 00 00 00 00 (Meter reply the power alarm threshold was successfully set)	A5

Gambar 2.7 *Library* Protokol Komunikasi PZEM-004T

(Sumber : GitHub - olehs/PZEM004T: Arduino communication library for Peacefair PZEM-004T Energy monitor)

Berdasarkan gambar diatas dapat diketahui bahwa dalam melakukan komunikasi dengan sensor PZEM-004T mempunyai protocol seperti yang tertulis pada gambar tersebut. Misalkan untuk mengatur ambang batas alarm daya sebesar 20 kW maka mikrokontroler lain yang digunakan sebagai master akan mengirim perintah B5 C0 A8 01 14 33, dimana 14 dalam perintah tersebut adalah nilai alarm(14 merupakan representasi data heksadesimal yang diubah menjadi desimal yaitu 20). Hal yang perlu diperhatikan adalah nilai alarm daya modul ini berdasarkan satuan kW, artinya nilai alarm 1 kW dan nilai maksimal 22 kW. Dengan demikian sensor PZEM-004T akan mengirimkan data balasan yaitu A5 00 00 00 00 00 A5.

2.2.6 *Light Emitting Diode* (LED)

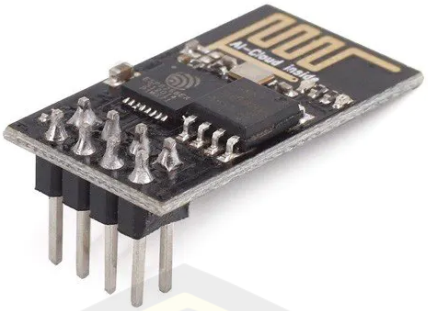
Gambar 2.8 LED SMD 0805

(Sumber : http://www.ledlight-components.com/)

Light Emitting Diode identik dengan bagian elektronika yang berbahan semikonduktor dan mampu mengeluarkan sinar monokromatik apabila dilewatkan tegangan bias maju. Warna dari pancaran cahaya LED ini dipengaruhi oleh jenis pada bahan semikonduktor yang digunakan. Komponen ini sering sekali digunakan sebagai indikator yang menandakan status dari rangkaian tersebut. Seperti contohnya pada suatu alat *rice cooker* terdapat LED indikator berwarna merah yang menunjukkan bahwa *rice cooker* dalam mode menanak nasi dan warna kuning yang menunjukkan bahwa *rice cooker* dalam mode *warm*.

2.2.7 Modul Wifi ESP8266-01

ESP8266-01 merupakan *system on a chip* (SOC) independen dengan bertumpuk protokol TCP/IP terintegrasi, yang bisa menyediakan akses mikrokontroler untuk jaringan WiFi. ESP8266 dapat menghosting aplikasi ataupun menghapus seluruh fungsi jaringan WiFi dari prosesor aplikasi lain. ESP-01 ESP8266 sudah diprogram dengan perangkat tegar atau firmware set perintah AT, dimana firmware sendiri mempunyai kemiripan dengan sistem operasi yang ada pada komputer, yang berarti pengguna hanya perlu mencocokkannya ke perangkat mikrokontroler utamanya misalnya Arduino untuk mendapatkan kemampuan WiFi yang disediakan oleh WiFi Shield.



Gambar 2.9 ESP8266-01

(Sumber : https://create.arduino.cc/projecthub/pratikdesai/)

ESP8266-01 memiliki banyak fungsi pemrosesan dan penyimpanan *on board*, dan dapat diintegrasikan dengan sensor dan perangkat khusus aplikasi lainnya melalui GPIO, sehingga meminimalkan jumlah pekerjaan pra-pengembangan dan beban paling sedikit saat runtime. Tingkat integrasi *on chip* ESP8266-01 yang cukup tinggi dapat memungkinkan jalur eksternal terkecil termasuk modul *front end* dirancang untuk menempati area PCB paling sedikit. ESP8266-01 mendukung APSD dengan aplikasi VoIP dan antarmuka koeksistensi Bluetooth, termasuk radio frekuensi yang melakukan kalibrasi sendiri yang dapat bekerja dalam semua kondisi pengoperasian dan tidak memerlukan komponen radio frekuensi eksternal.

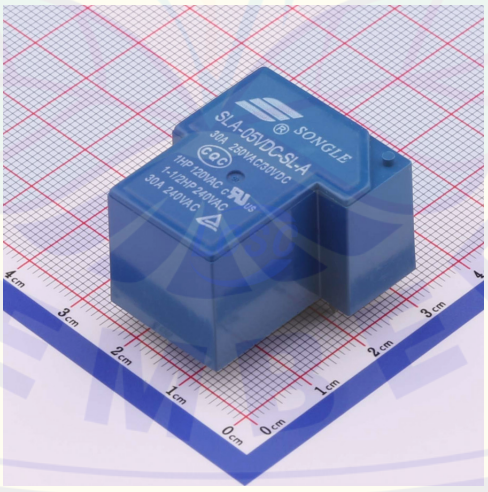
Fitur yang terdapat pada modul ESP8266-01 dapat dituliskan sebagai berikut :

1. 802.11 b/g/n
2. Wi-Fi Direct (P2P), soft-AP
3. Tumpukan protokol TCP / IP terintegrasi
4. Sakelar TR terintegrasi, balun, LNA, penguat daya, dan jaringan yang cocok
5. PLL terintegrasi, regulator tegangan, DCXO dan unit manajemen daya
6. +19.5dBm *output power in* 802.11b *mode*
7. *Power down leakage current of* <10uA
8. 1MB *Flash Memory*

9. Konsumsi daya rendah terintegrasi 32-bit CPU dapat digunakan sebagai prosesor aplikasi
10. SDIO 1.1 / 2.0, SPI, UART
11. STBC, 1×1 MIMO, 2×1 MIMO
12. A-MPDU & A-MSDU *aggregation* & 0.4ms interval penjaga
13. Waktu untuk menyala dan kirimkan paket masuk < 2ms
14. Konsumsi daya saat siaga < 1.0mW (DTIM3)

2.2.8 Relai

Relai identik dengan salah satu bagian elektronika yang digunakan sebagai *switch* atau sakelar elektrik yang dioperasikan dengan sistem kelistrikan dan terdiri dari 2 bagian utama yaitu elektromagnetik atau yang biasa disebut mekanikal dan koil yang merupakan satu setel kontak sakelar. Cara kerja relai memanfaatkan tindakan kerja elektromagnetik untuk menggerakan posisi sakelar sehingga aliran listrik yang sedikit bisa digunakan untuk menghantarkan listrik yang yang memiliki beda potensial lebih tinggi. Misalnya relai dengan elektromagnet 12V dan 75 mA sanggup menggerakan jangkar magnet relai yang difungsikan sebagai sakelarnya untuk menghantarkan listrik dengan tegangan yang lebih tinggi misalnya yaitu 220V dengan arus maksimal 10A.



Gambar 2.10 Relai Songle SLA-5VDC 30A

(Sumber : https://lcsc.com/product-detail/)

Relai terdapat 4 komponen dasar yakni *armature* atau jangkar magnet, elektromagnet atau coil, kontak sakelar, dan pegas. Relai memiliki dua jenis kontak antara lain *normally close* (NC) dan *normally open* (NO). NC merupakan kondisi kontak mula-mula yang posisinya tertutup. NO adalah kebalikan dari NC yaitu kondisi kontak awal yang posisinya terbuka.

2.2.9 *Push Button*

Push button identik dengan tombol tekan yang bertugas untuk penghubung atau pemutus arus listrik. Push button berbeda dengan sakelar *switch*, prinsip kerja dari push button yaitu tidak mengunci dengan artian setelah push button tidak ditekan lagi maka posisi kontak akan kembali pada posisi semula sedangkan sakelar *switch* mempunyai prinsip kerja mengunci posisi kontaknya.

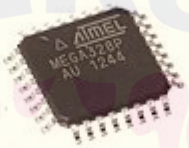
Gambar 2.11 *Push Button*

(Sumber : http://id.chinapushbuttons.com/metal-push-button/)

Push button mempunyai 3 macam, yaitu titik kontak *normally open* (NO), titik kontak *normally close* (NC), serta titik kontak *normally open* (NO) dan *normally close* (NC). Titik kontak *normally open* (NO) merupakan kondisi kontak sebelum ditekan dalam keadaan terbuka. Titik kontak *normally close* (NC) merupakan kondisi kontak sebelum ditekan dalam keadaan tertutup. Push button yang mempunyai model gabungan dari titik kontak *normally open* (NO) dan *normally close* (NC) bekerja dengan prinsip kedua titik kontak yang telah dijelaskan sebelumnya sehingga *push button* model ini memiliki terminal sebanyak tiga buah.

2.2.10 Atmega328AU

Secara umum ATmega328P merupakan mikrokontroler berbasis AVR CMOS 8 bit dengan arsitektur RISC. AVR dapat mengeksekusi instruksi dalam 1 clock cycle, sehingga ATmega328P dapat mencapai sekitar 1 MIPS per Hz. Oleh karena itu, ATmega328P mengoptimalkan konsumsi daya dan kecepatan pemrosesan, dimana tipe ATmega328P-AU ini adalah versi SMD dari ATmega328P-PU sehingga karena ATmega328P-AU ini adalah versi SMD dimana ukurannya lebih kecil dari yang ATmega328P-PU maka konsumsi dayanya sangat kecil dan rendah



Gambar 2.12 Atmega328P-AU

(Sumber : https://en.wikipedia.org/wiki/ATmega328)

Penggambaran secara singkat mikrokontroler berdasarkan pico Power 8-bit AVR RISC Microchip berkinerja tinggi menggabungkan memori flash ISP 32 KB dengan fungsi baca dan tulis, 1024B EEPROM, 2 KB SRAM, 23 jalur I / O, 32 fungsi umum register, tiga A pengatur waktu / penghitung fleksibel dengan mode pembanding, interupsi internal dan eksternal, serial USART yang dapat diprogram, antarmuka serial dua kabel berorientasi byte, port serial SPI, konverter A / D 10-bit 6 saluran (TQFP dan QFN) 8 saluran / paket dalam MLF), pengatur waktu pengawas yang dapat diprogram dengan osilator internal dan 5 mode hemat daya yang dapat dipilih perangkat lunak. Perangkat bekerja antara 1,8-5,5 volt. Dengan menjalankan instruksi yang kuat dalam satu siklus clock, perangkat dapat mencapai *throughput* yang mendekati 1 MIPS per MHz, sehingga menyeimbangkan konsumsi daya dan kecepatan pemrosesan.

Tabel 2.1 Karakteristik Atmega328P-AU

Name	**Value**
Program Memory Type	Flash
Program Memory Size (KB)	32
CPU Speed (MIPS/DMIPS)	20
SRAM (B)	2.048
Data EEPROM/HEF (bytes)	1024
Digital Communication Peripherals	2-UART, 2-SPI, 2-I2C
Capture/Compare/PWM Peripherals	3 Input Capture, 3 CCP, 10PWM
Timers	2 x 8-bit, 3 x 16-bit
Number of Comparators	1
Temperature Range (℃)	-40 to 105
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	32
Low Power	Yes
0-4 MHz (V)	1.8 -5.5
0-10 MHz (V)	2.7 - 5.5
0-20 MHz (V)	4.5 - 5.5
Active Mode (mA)	0.24
Power Down Mode (uA)	0.2
Power Save Mode (uA)	1.3 (including 32kHz RTC)

2.2.11 Crystal

Osilator kristal adalah rangkaian osilator elektronik yang menggunakan resonansi mekanis dari kristal getar bahan piezoelektrik untuk menghasilkan sinyal listrik dengan frekuensi konstan. Frekuensi ini sering digunakan untuk melacak waktu, seperti jam tangan kuarsa, untuk menyediakan sinyal jam yang stabil untuk sirkuit terintegrasi digital, dan untuk menstabilkan frekuensi untuk pemancar dan penerima radio. Jenis resonator piezoelektrik yang paling umum

digunakan adalah kristal kuarsa, sehingga rangkaian osilator yang menggabungkannya disebut osilator kristal, tetapi bahan piezoelektrik lainnya termasuk keramik polikristalin juga digunakan di rangkaian serupa.

Gambar 2.13 *Crystal* 16MHz SMD
(Sumber : https://www.machineryoffers.com/offer/)

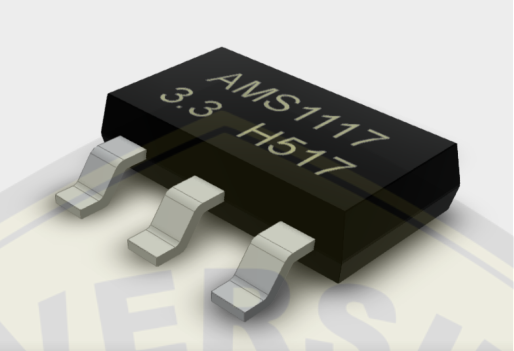
Osilator kristal mengandalkan perubahan halus pada bentuk kristal kuarsa dibawah aksi medan listrik. Karakteristik ini disebut elektrostriksi atau piezoelektrik terbalik. Tegangan yang diterapkan ke elektroda pada kristal mengubah bentuk kristal. Setelah tegangan dihilangkan, kristal akan secara elastis kembali ke bentuk aslinya, menghasilkan lebih sedikit tegangan. Kuarsa berosilasi pada frekuensi resonansi yang stabil, yang berperilaku seperti rangkaian RLC, tetapi memiliki faktor Q yang lebih tinggi (lebih sedikit energi yang hilang per siklus osilasi). Setelah menyetel kristal kuarsa ke frekuensi tertentu (dipengaruhi oleh kualitas elektroda yang terpasang pada kristal, arah kristal, suhu, dan faktor lainnya), frekuensi dapat menjaga stabilitas tinggi.

Frekuensi dari produksi kristal kuarsa berkisar dari puluhan kilohertz hingga ratusan megahertz. Ini terutama digunakan di perangkat konsumen seperti jam tangan, jam, radio, komputer dan telepon seluler. Kristal kuarsa juga muncul dalam peralatan pengujian dan pengukuran seperti penghitung, generator sinyal, dan osiloskop.

2.2.12 LM1117

LM1117 adalah regulator tegangan dropout rendah dengan penurunan tegangan 1.2V pada arus beban 800 mA. LM1117 memiliki versi yang dapat

disesuaikan, hanya dua resistor eksternal yang diperlukan untuk mengatur tegangan output antara 1,25 dan 13,8 V. Selain itu, ini juga menyediakan lima tegangan tetap: 1,8 V, 2,5 V, 3,3 V, dan 5V.



Gambar 2.14 Regulator LM1117 SMD

(Sumber : https://grabcad.com/library/ams1117-3v3-1)

LM1117 menyediakan fungsi batas arus dan penghentian *thermal*. Rangkaian ini menyertakan referensi celah pita level zener untuk memastikan bahwa akurasi tegangan keluaran berada dalam ± 1%. Setidaknya kapasitor tantalum 10µF diperlukan pada keluaran untuk meningkatkan respons transien dan stabilitas. Adapun fitur pada regulator LM1117 sebagai berikut:

1. Tersedia dalam 1,8 V, 2,5 V, 3,3 V, 5 V dan versi yang tegangannya dapat disesuaikan
2. Paket SOT-223 dan WSON yang hemat tempat
3. Pembatasan arus dan proteksi termal
4. Arus keluaran: 800 mA
5. Tingkat penyesuaian: 0,2% (maksimum)
6. Tingkat pengaturan beban: 0,4% (maksimum)
7. Kisaran suhu: 0 ° C hingga 125 ° C

2.2.13 EasyEDA

EasyEDA merupakan alat untuk merangkai berbagai macam rangkaian elektronika yang berbasis web yang memungkinkan seluruh orang untuk merancang, mensimulasikan, berbagi secara publik maupun pribadi,

mendiskusikan skematik, simulasi, dan papan sirkuit yang akan dicetak dalam rancangan yang dibuat.

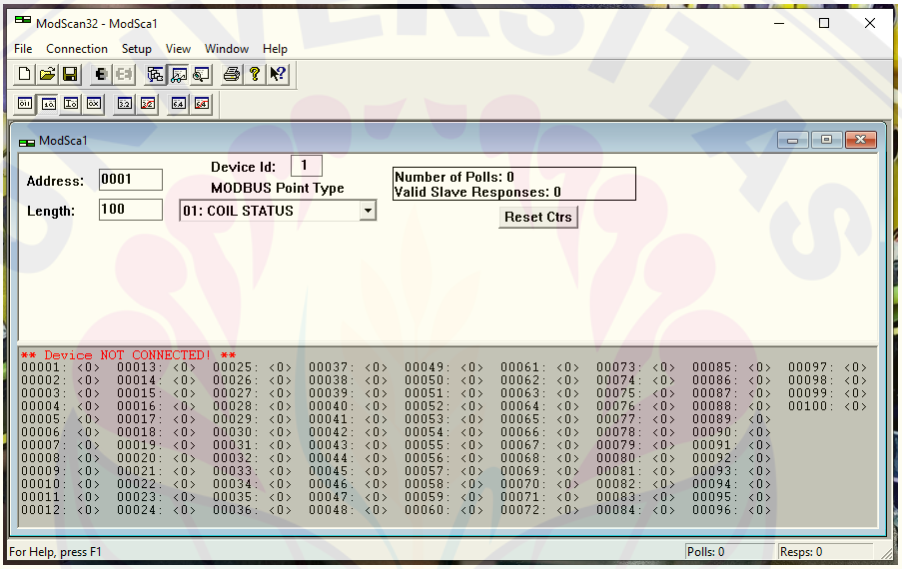
EasyEDA juga menyediakan fitur untuk membuat *bill of materiali*, file Gerber, file *pick and place*, dan keluaran dokumen dalam format PDF, PNG, dan SVG. EasyEDA juga memungkinkan pembuatan dan pengeditan diagram sirkuit dalam skematik, simulasi SPICE, dari sirkuit analog dan digital campuran, serta pembuatan maupun pengeditan tata letak papan sirkuit cetak, dan bahkan dapat membuat opsi papan sirkuit yang akan dicetak.

Gambar 2.15 Fitur EasyEDA

EasyEDA dapat digunakan oleh siapapun dengan gratis dan ini disediakan untuk umum dalam web maupun tautan untuk mengunduh sebagai software yang ingin di pasang pada laptop maupun komputer pengguna. EasyEDA juga menyediakan untuk pelanggan khusus yang bersedia untuk membayar bulanan dengan fitur khusus antara lain seperti proyek publik yang berkualitas tinggi, simbol skema dan desain PCB dengan fitur yang lebih lengkap, dan beberapa fitur lainnya. Selain itu EasyEDA juga terhubung dengan perusahaan layanan pembuatan PCB yang sangat terkenal yaitu JLCPCB dan EasyEDA juga terhubung dengan perusahaan penyedia komponen elektronika terlengkap yaitu LCSC. Layanan ini dapat digunakan dengan cukup mudah yaitu hanya dengan memasukkan file Gerber ataupun file BOM yang sudah disediakan pula oleh EasyEDA.

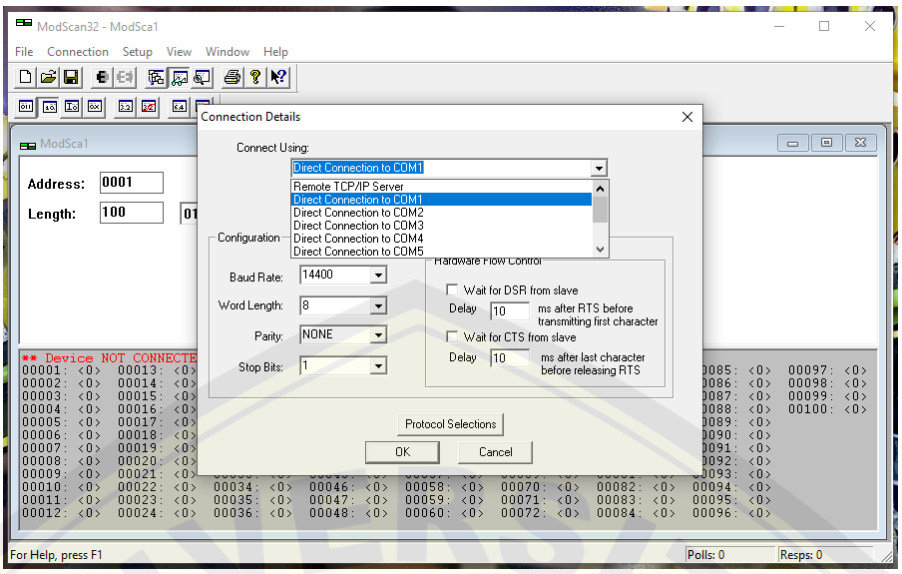
2.2.14 ModScan

ModScan merupakan *software* yang mampu melakukan komunikasi Modbus yang dijalankan pada PC/Laptop. ModScan adalah aplikasi pada OS Windows yang dirancang untuk beroperasi sebagai perangkat modbus master untuk mengakses titik data di perangkat pendukung kompatibel PLC yang terhubung. Software ini dirancang terutama sebagai perangkat pengujian untuk verifikasi operasi protokol yang benar dalam sistem baru atau yang sudah ada. Titik data modbus, coil dan register, dapat dibaca dan/atau ditulis dari aplikasi ModScan menggunakan perintah MODBUS 01-06.



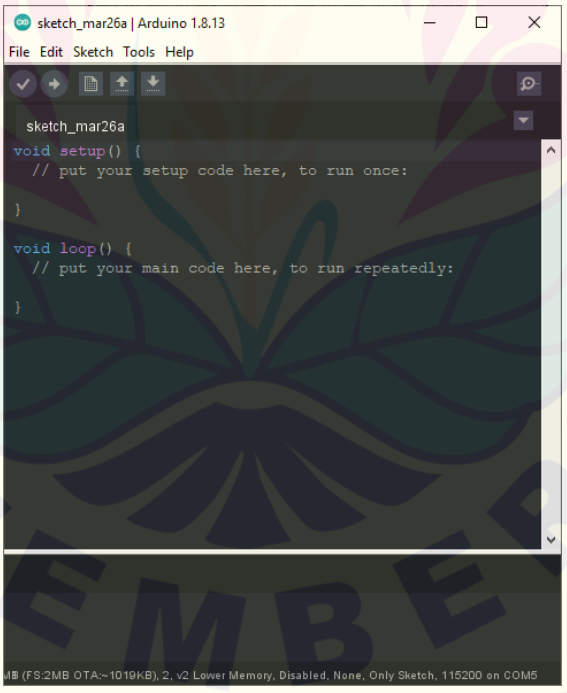
Gambar 2.16 Tampilan *Software* ModScan

Perangkat modbus slave yang ditentukan disurvei oleh *software* ModScan berdasarkan siklus yang ditentukan oleh pengguna. Setiap alamat yang didukung oleh perangkat dapat dipindai dan nilai yang terkait tertulis di bagian bawah tampilan. Kesalahan komunikasi atau tanggapan pengecualian yang dikembalikan oleh perangkat slave juga dicantumkan di baris status ModScan. Opsi menu memungkinkan nilai register untuk ditampilkan dalam notasi desimal atau heksadesimal. Selain itu juga terdapat pilihan untuk memilih mode transmisi modbus. Salah satu fitur tambahan ModScan adalah kemampuan untuk menampilkan lalu lintas data aktual antara perangkat master dan slave.



Gambar 2.17 Pilihan Mode Transmisi Modbus pada ModScan

2.2.15 Arduino IDE



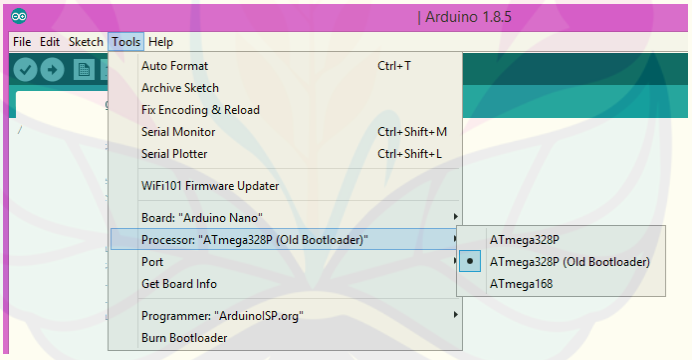
Gambar 2.18 Tampilan Arduino IDE

Arduino adalah platform elektronik *open source* yang didasari oleh perangkat keras dan perangkat lunak yang mudah digunakan. Papan

pengembangan Arduino dapat membaca berbagai macam sumber masukan seperti halnya input-cahaya pada sensor, jari pada tombol atau pesan dari Twitter atau Telegram dan kemudian mengubahnya dalam bentuk keluaran untuk menghidupkan motor, menyalakan LED dan memposting konten secara online. Anda dapat memberitahu Anda hal apa yang harus dilakukan dengan mengirimkan serangkaian instruksi ke mikrokontroler di papan tulis. Untuk ini, Anda dapat menggunakan bahasa pemrograman Arduino (berdasarkan Wiring) dan perangkat lunak Arduino (IDE) (berdasarkan Processing).

2.2.16 Bootloader

Secara singkat Bootloader adalah sepotong kecil kode (kode yang dapat dieksekusi dalam format .hex) yang berada di dalam memori mikrokontroler. Bootloader di Arduino memungkinkan kita memprogram Arduino melalui port serial, yaitu menggunakan kabel USB. Tugas dari Bootloader di Arduino adalah menerima kode dari komputer dan memasukkannya ke dalam memori mikrokontroler.



Gambar 2.19 Bootloader Arduino Nano Pada Arduino IDE

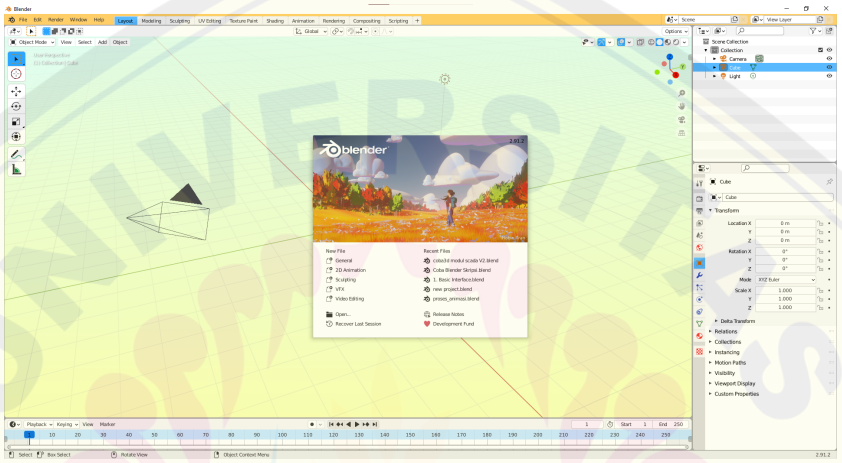
(Sumber : https://qastack.id/programming/19765037/)

Pada gambar diatas menunjukkan instalasi bootloader pada Arduino Nano yang terbaca oleh Arduino IDE. Secara tradisional, mikrokontroler seperti Atmel ATmega328 diprogram dengan bantuan programmer khusus, yang melibatkan beberapa koneksi khusus. Bootloader menghilangkan kerumitan ini dan memberi kami cara mudah untuk memprogram mikrokontroler hanya dengan menggunakan kabel USB. Bootloader terletak di lokasi khusus yang aman di memori flash yang

dapat diprogram dari mikrokontroler, yang biasanya membutuhkan kurang dari 1KB memori.

2.2.17 Blender

Blender adalah kit pembuatan 3D gratis dan open source. Ini mendukung seluruh pemodelan pipa 3D, perakitan, animasi, simulasi, rendering, pengkomposisian dan pelacakan gerak, pengeditan video dan pipa animasi 2D.



Gambar 2.20 Tampilan Blender V2.91.2

Adapun beberapa fitur penting pada Blender sebagai berikut:

1. Pemodelan
   a. Primitif

      Blender mendukung berbagai primitif geometris, termasuk mesh poligon, pemodelan permukaan subdivisi cepat, kurva Bezier, permukaan NURBS, metaballs, bola ico, teks, dan sistem pemodelan n-poligon yang disebut mesh B.

   b. Pengubah

      Blender juga menyediakan fitur mengubah yang menerapkan efek yang bersifat tidak merusak bentuk dari desain itu sendiri.

c. Patung

Blender juga memiliki patung digital dengan beberapa resolusi, termasuk topologi dinamis, pembuatan peta, pembentukan kembali, simetri ulang, dan ekstraksi.

2. Simulasi

Blender dapat digunakan untuk mensimulasikan asap, hujan, debu, kain, cairan, rambut, dan benda kaku. Simulator fluida dapat digunakan untuk mensimulasikan cairan, seperti air yang menabrak cangkir. Ia menggunakan metode Lattice Boltzmann untuk mensimulasikan fluida dan memungkinkan sejumlah besar penyesuaian pada jumlah partikel dan resolusi. Simulasi fluida fisika partikel menciptakan partikel yang mengikuti pendekatan "dinamika fluida partikel halus". Alat simulasi untuk dinamika perangkat lunak, termasuk deteksi tabrakan grid, dinamika fluida LBM, simulasi asap, dinamika benda kaku peluru, dan generator samudera dengan gelombang. Sistem partikel yang mencakup dukungan untuk rambut berbasis partikel. Kontrol waktu nyata selama simulasi fisik dan rendering. Di Blender 2.82, sistem simulasi fluida baru yang disebut mantaflow ditambahkan untuk menggantikan sistem lama.

3. Animasi

Alat animasi bingkai utama meliputi kinematika terbalik, angker (kerangka), kait, deformasi berdasarkan kurva dan kisi, animasi bentuk, animasi nonlinier, pembatas, dan bobot titik. Alat *Grease Pencil* Blender dapat melakukan pemrosesan animasi 2D dalam pipa 3D lengkap, dan masih banyak fitur lainnya pada Blender.

BAB 3. METODOLOGI PENELITIAN

Pada penelitian ini untuk mencapai hasil penelitian yang sesuai dengan tujuan yang diharapkan, tahapan yang direncanakan dalam penelitian ini dijelaskan dalam bagan berikut ini.

3.1 Tempat Penelitian

Penelitian ini dilaksanakan di Laboratorium Sistem Tenaga Jurusan Teknik Elektro, Fakultas Teknik, Universitas Jember.

3.2 Waktu Penelitian

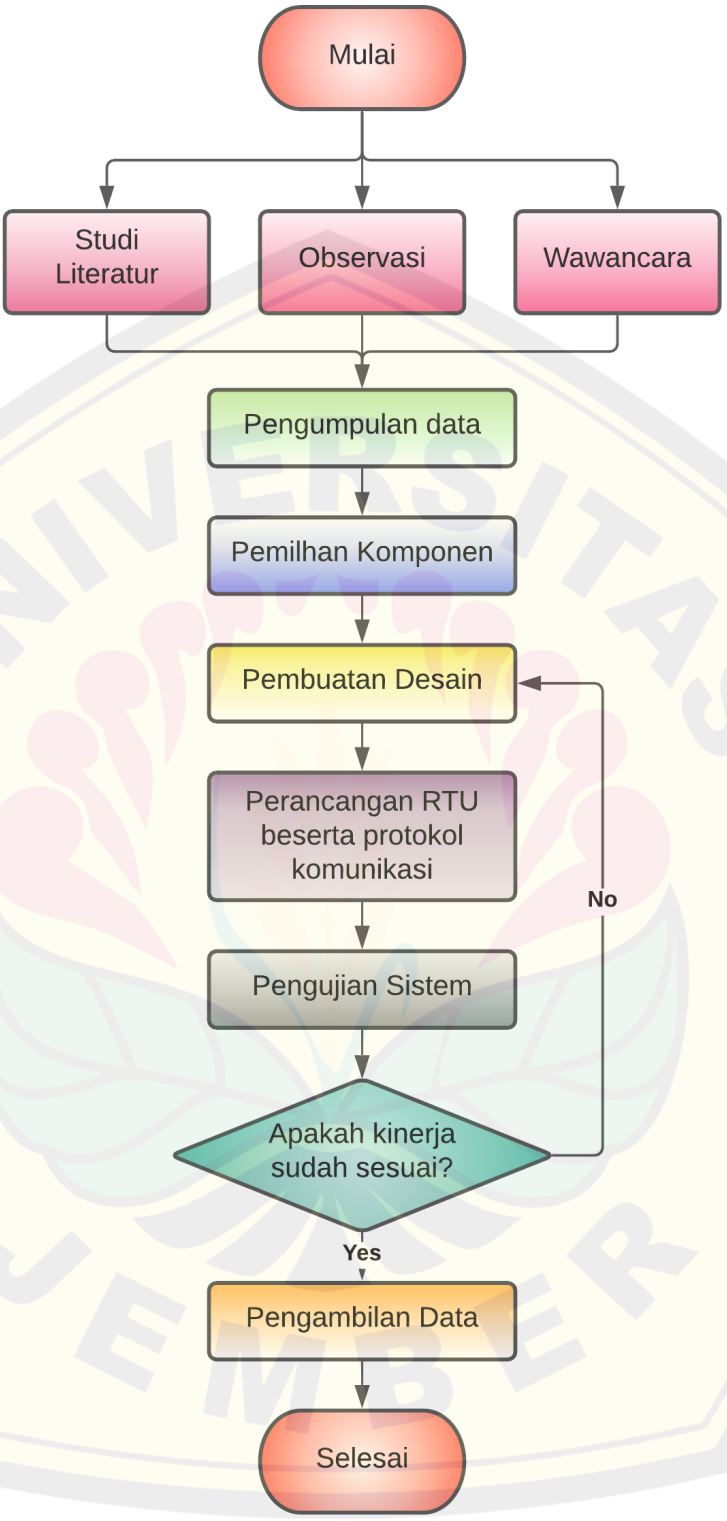
Penelitian ini dimulai pada bulan Februari 2021 hingga selesai, dengan rincian waktu sebagai berikut:

Tabel 3.1 Rencana Jadwal Penelitian

No.	Kegiatan	Bulan ke-1				Bulan ke-2				Bulan ke-3			
		1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Literatur												
2	Rancangan Sistem												
3	Pengambilan Data												
4	Analisis Data dan Pembahasan												
5	Penyusun Laporan												

3.3 Tahapan Penelitian

Tahapan yang akan dilakukan pada penelitian ini digambarkan dalam gambar berikut:



Gambar 3.1 *Flowchart* penelitian

Berdasarkan diagram tahapan yang akan dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Tahapan ini merupakan tahapan bagi penulis untuk memunculkan ide-ide baru pada penelitian yang akan dilakukan sekaligus mempertajam ide yang telah didapatkan dengan menelusuri dan mencari sumber yang dapat digunakan sebagai referensi untuk penelitian yang akan dilakukan.

2. Observasi

   Pada tahapan ini penulis mencatat data informasi untuk penelitian yang akan dilakukan. Proses yang dilakukan pada tahapan ini meliputi pengamatan serta peninjauan secara langsung pada lokasi penelitian dengan tujuan untuk mengetahui kondisi lapangan sehingga penelitian yang akan dilakukan dapat direncanakan dengan tepat.

3. Wawancara

   Tahapan ini diperlukan penulis untuk melengkapi data kondisi pada lapangan yaitu Gedung B Fakultas Teknik Universitas Jember dan memperdalam pemahaman pada penelitian yang sudah ada yaitu penelitian yang dilakukan oleh saudara Fahim melalui proses tanya jawab.

4. Pengumpulan Data

   Tahapan ini bertujuan untuk mengumpulkan data dan informasi yang terdapat di lapangan sehingga diharapkan penulis dapat memvalidasi data penelitian dan informasi yang diberikan oleh penulis diharapkan tidak menyesatkan bagi peneliti lainnya.

5. Pemilihan Komponen

   Faktor - faktor yang mempengaruhi keberhasilan dari penelitian salah satunya yaitu ketepatan pemilihan komponen, tahapan ini dipergunakan oleh penulis untuk menganalisis dan mempersiapkan komponen yang akan digunakan dalam perancangan sistem. Dengan demikian proses penelitian diharapkan dapat berjalan dengan lancar salah satunya yaitu mempermudah proses pengujian sistem.

6. Pembuatan Desain

   Pada tahapan ini penulis melakukan perancangan desain pada skematik rangkaian maupun penutup dari RTU sehingga penelitian yang dilakukan menghasilkan sebuah modul yang berkualitas dan layak untuk digunakan sebagai bahan pembelajaran maupun diproduksi secara masal.

7. Perancangan RTU beserta protokol komunikasi

   Pada tahapan ini penulis melakukan perancangan pada sistem yang akan dibangun seperti perancangan program, perancangan kode fungsi, perancangan protokol komunikasi, dan lain-lain.

8. Pengujian Sistem

   Tahapan ini diperlukan bagi penulis agar proses pengujian dapat dilakukan secara terstruktur untuk mengetahui keberhasilan, keandalan, keakuratan, dan kepresisian dari sistem yang telah dibangun.

9. Pengambilan Data

   Seperti penelitian pada umumnya bahwa penelitian yang dilakukan diperlukan proses pengambilan data dengan tujuan salah satunya yaitu informasi yang diberikan mampu dipahami dengan mudah dan data yang telah diperoleh akan dilakukan analisis yang dapat digunakan sebagai kesimpulan maupun saran pada penelitian ini.

3.4 Alat dan Bahan Penelitian

Alat dan bahan yang digunakan pada penelitian ini antara lain sebagai berikut:

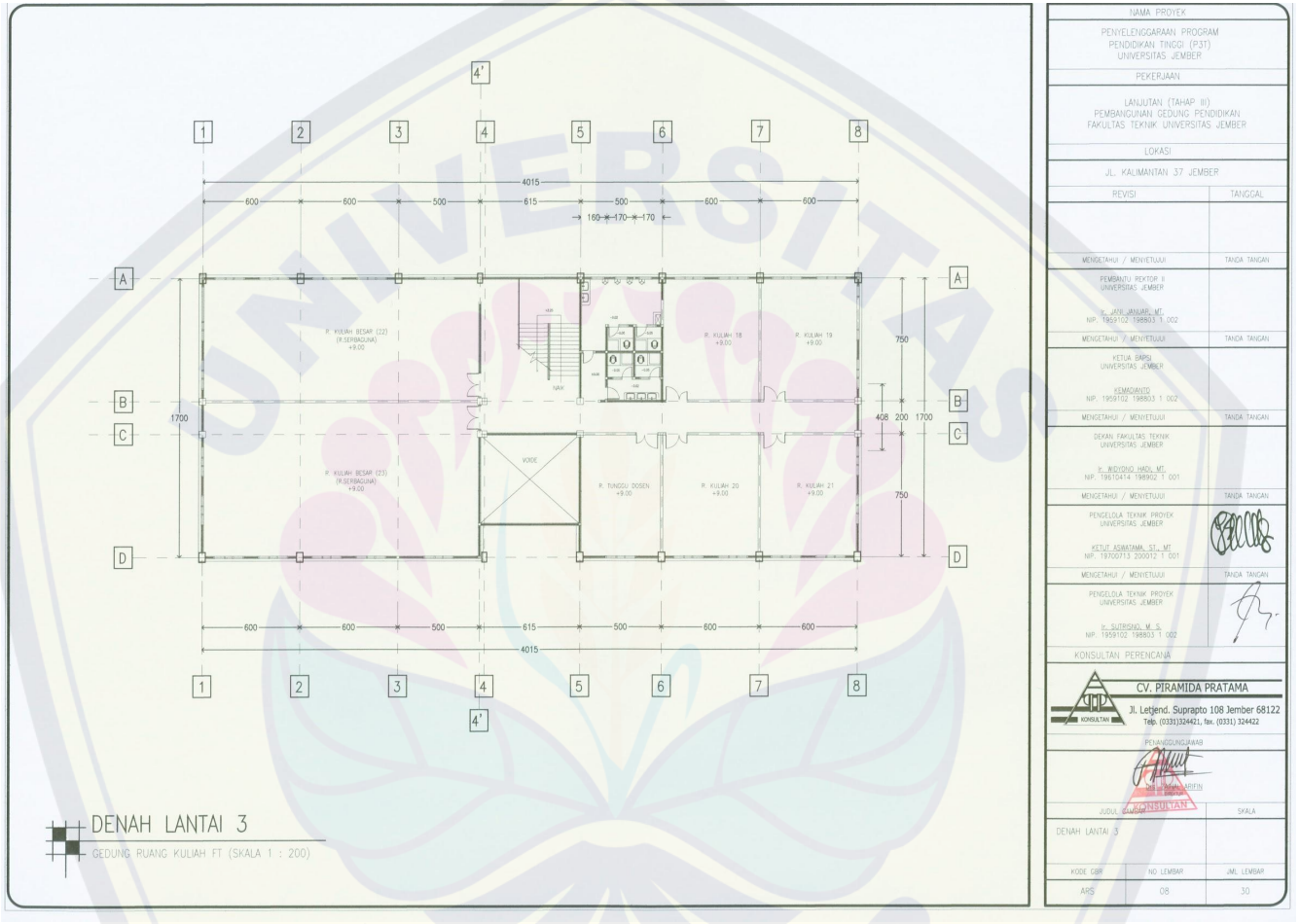
1.	Komputer/Laptop	1 buah
2.	*Software* ModScan	1 buah
3.	*Software* Arduino IDE	1 buah
4.	*Software* Blender	1 buah
5.	*Software* CorelDraw/Inkscape	1 buah
6.	EasyEDA (*Electronic Design Automation*)	*Online*
7.	*Router*	1 buah
8.	Kabel power AC	1 buah

9. Sakelar sumber AC	1 buah
10. Sekring 1A	1 buah
11. Hi-Link *Rectifier* (220V-5V)	1 buah
12. Dioda 1N4007M7 SMD	2 buah
13. AMS1117-3,3V SMD	1 buah
14. Kapasitor 22µF SMD	1 buah
15. Kapasitor 100nF SMD	2 buah
16. Kapasitor 22pF SMD	2 buah
17. Kapasitor 1000µF SMD	1 buah
18. Lampu LED SMD 1206	Secukupnya
19. Resistor 300 Ohm SMD 1206	4 buah
20. Resistor 10K Ohm SMD 1206	17 buah
21. Resistor 100 Ohm SMD 1206	7 buah
22. Resistor 33 Ohm SMD 1206	7 buah
23. Induktor 47 uH SMD	1 buah
24. *Crystal/Oscillator* 16Mhz	1 buah
25. Atmega328P-AU	1 buah
26. USB *to* TTL *converter* FT232RL	1 buah
27. Modul Wifi ESP8266-01	1 buah
28. Relai 5V 30A	2 buah
29. LCD Oled 128×64 - 2,42”	1 buah
30. Sensor arus Clamp YHDC SCT013	7 buah
31. Sensor PZEM-004T	1 buah
32. Push Button	2 buah
33. Push Button SMD 4P	1 buah
34. Socket XH-2,54mm 3P	1 pasang
35. Socket XH-2,54mm 4P	2 pasang
36. Terminal Block 3P	1 buah
37. Terminal Block 2P	2 buah
38. Pin Header 2x4 Female	1 buah
39. Pin Header 2x3 Male	1 buah

40. Pin Header (Male & Female)	Secukupnya
41. Banana Plug & Socket	5 pasang
42. Socket Audio Jack 5P	7 buah
43. Transistor BC547B	2 buah
44. Kabel Jumper	Secukupnya
45. Kabel NYM/NYAF 1,5mm^2	Secukupnya
46. *Acrylic* bening	Secukupnya
47. Solder, Timah, dan bahan lainnya	Secukupnya
48. Mur & baut 3mm	Secukupnya
49. *Spacer* 3mm	Secukupnya
50. Kit peralatan	Secukupnya

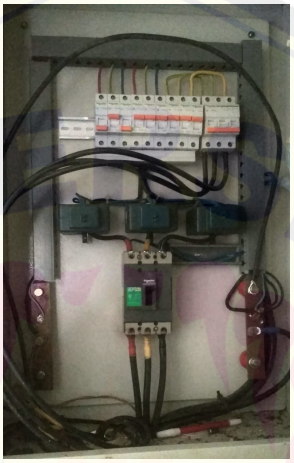
3.5 Rancangan Penelitian

Penelitian ini dilaksanakan di Gedung B Fakultas Teknik Universitas Jember. Gedung tersebut memiliki 25 ruangan yang terbagi menjadi 3 lantai untuk ruang kuliah. Sistem kelistrikan gedung B dibagi menjadi beberapa kelompok instalasi, di antaranya pembagiannya dirancang untuk mencegah kematian semua instalasi bangunan bila ada bagian tertentu dari instalasi yang gagal. Selain itu, pembagian grup instalasi digunakan untuk mempermudah perawatan operator atau teknisi. Pembagian grup instalasi juga dapat digunakan untuk mengelompokkan atau memisahkan beban sesuai dengan jenis beban yang ada.



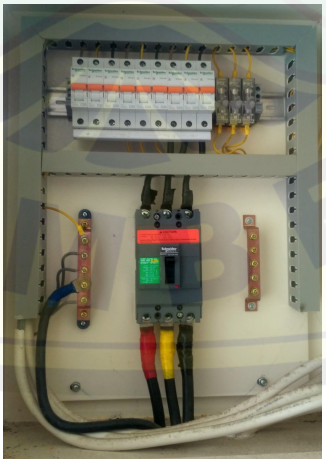
Gambar 3.2 Denah Gedung B Fakultas Teknik Universitas Jember

Pada sistem kelistrikan, setiap lantai pada Gedung B Fakultas Teknik Universitas Jember terbagi menjadi dua jenis panel sesuai dengan jenis pembebanannya, dan masing-masing panel terdiri dari beberapa kelompok. Panel utama di lantai satu dibagi menjadi panel beban AC dan panel beban lainnya, beban lainnya yang dimaksud seperti kotak kontak dan penerangan yang tersebar di setiap lantai.



Gambar 3.3 Panel Utama Gedung B Fakultas Teknik Universitas Jember

Tujuan dari penelitian ini difokuskan pada Gedung B lantai tiga Fakultas Teknik Universitas Jember yang dibagi menjadi 8 kelompok beban, yaitu 5 kelompok beban AC dan 3 kelompok beban lainnya, panel beban lainnya yang dimaksud seperti penerangan dan kotak kontak.



Gambar 3.4 Panel Beban AC Lantai 3

Gambar 3.5 Panel Beban Penerangan dan Kotak Kontak Lantai 3

Gedung B Fakultas Teknik Universitas Jember adalah gedung pusat perkuliahan mahasiswa teknik yang terdiri dari banyak beban. Berikut ini adalah data dari berbagai beban pada lantai 3:

Tabel 3.2 Data Beban Setiap Grup Panel Beban Lain

Beban	Grup 1 (16A)	Grup 2 (16A)	Grup 3 (10A)	Grup 4 (10A)	Grup 5 (10A)	Grup 6 (10A)
Lampu TL 20W	-	-	-	-	2 x 20	2 x 10
Lampu CFL 40W	-	-	-	-	19	5
Lampu CFL 20W	-	-	-	-	13	7
PC	5	-	-	-	2	1
Monitor	3	-	-	-	-	-
Proyektor	2	-	-	-	2	1

Tabel 3.3 Data Beban Setiap Grup Panel AC

Beban	Jumlah AC	Beban	Jumlah AC	Beban	Jumlah AC
Grup 1	1	Grup 4	1	Grup 7	1
Grup 2	1	Grup 5	1	Grup 8	1
Grup 3	1	Grup 6	1	Grup 9	1

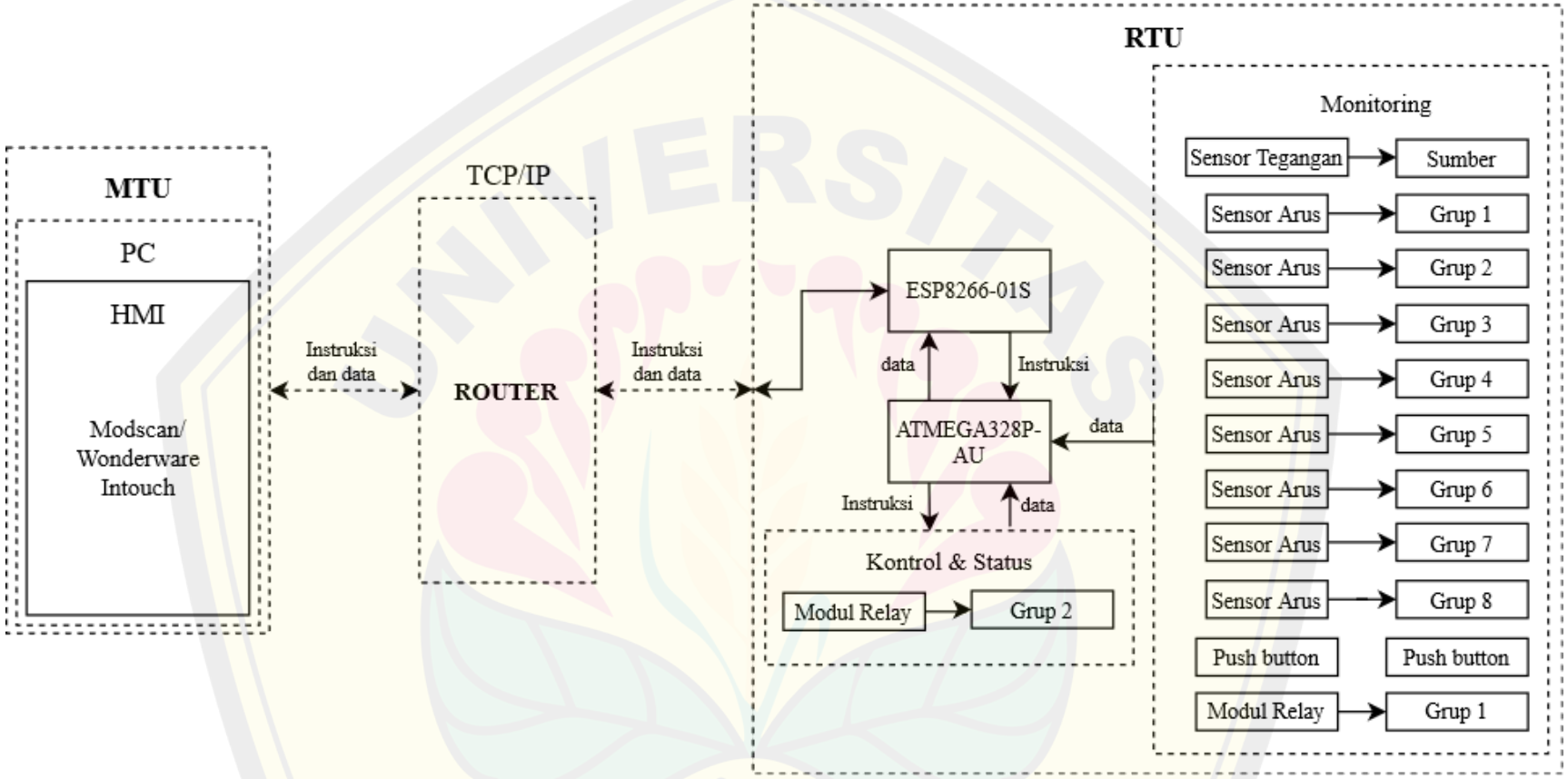
Tabel 3.4 Data Beban Setiap Setiap Ruangan

	Ruang 21	**Ruang 22**	**Ruang 23**	**Ruang 24**	**Ruang 25**	**Ruang 26**	**Ruang 27**	**Ruang 28**	**Kamar Mandi**	**Koridor**
Lampu TL 20W	2 x 8	2 x 4	-	-	-	2 x 4	2 x 4	2 x 4	-	2 x 6
Lampu CFL 20W	-	-	6	2	2	-	-	-	6	4
Lampu CFL 40W	-	-	6	4	10	-	-	-	2	2
PC	1	1	1	1	1	1	1	1	-	-
Monitor	-	-	-	-	-	1	1	1	-	-
Proyektor	1	1	1	1	1	-	-	-	-	-
AC	1	1	2	1	1	1	1	1	-	-

Oleh karena adanya pandemi covid-19 yang menerapkan *work from home* serta membatasi adanya kegiatan di kampus, maka pengujian penelitian ini tidak dapat dilakukan secara langsung pada beban yang ada di Gedung B Fakultas Teknik Universitas Jember. Pengujian pada penelitian ini menggunakan beban yang tersedia di Laboratorium Sistem Tenaga Fakultas Teknik.

3.5.1 Blok Diagram dan Cara Kerja

Sistem terdiri dari 1 MTU dan 1 RTU yang didalamnya terdapat mikrokontroler dan komponen lainnya, juga terdapat beberapa bagian yang dapat dikontrol oleh MTU secara jarak jauh. Berikut adalah blok diagram sistem secara keseluruhan :



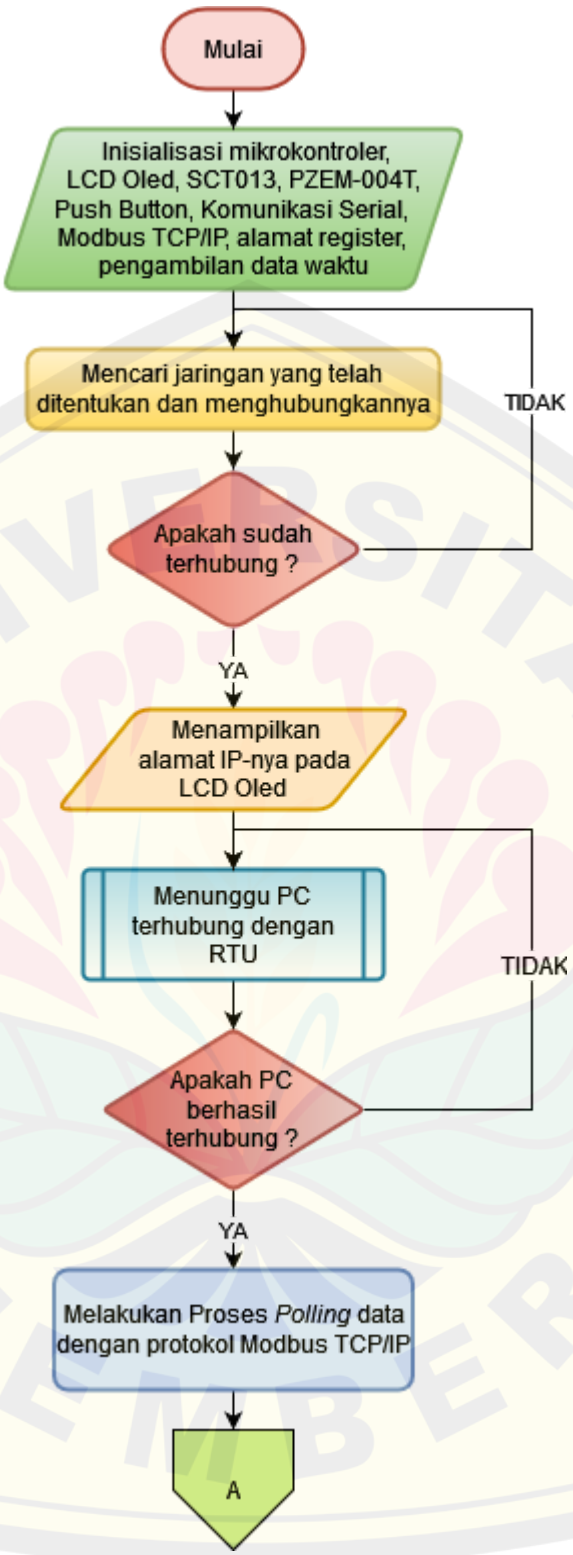
Gambar 3.6 Blok Diagram Sistem

Berdasarkan Gambar 3.6 diketahui sistem kerja dari perancangan pengontrolan dan pengawasan pada setiap grup menggunakan sistem SCADA. Dalam blok diagram tersebut terdiri dari satu *Master Terminal Unit* (MTU) yang digunakan yaitu *Personal Computer* (PC) atau laptop dimana didalamnya terdapat HMI yang digunakan yaitu ModScan maupun Wonderware Intouch. Pada saat awal pengujian komunikasi, HMI yang digunakan yaitu ModScan kemudian untuk pengujian selanjutnya direncanakan dilakukan pengujian dengan HMI yang dirancang pada Wonderware Intouch yang dikerjakan oleh Nabila Dinda Rahmania. Dalam blok diagram tersebut juga diketahui terdapat 1 buah *Remote Terminal Unit* (RTU) yang merupakan sekelompok komponen yang menjadi satu sistem. Dalam RTU tersebut terdiri dari 2 mikrokontroler yaitu ESP8266-01S dan ATmega328P-AU yang merupakan versi SMD dari ATmega328. Pada RTU tersebut terdapat 2 relai, dimana 1 relai untuk kontrol dan monitoring sedangkan 1 relai lainnya hanya untuk dimonitoring, dan komponen lainnya pada RTU juga dimanfaatkan sebagai monitoring. Koneksi antara MTU dengan RTU menggunakan protokol TCP/IP menggunakan jaringan *wireless* sehingga dalam komunikasi ini tidak diperlukan sebuah kabel. RTU akan berkomunikasi dengan MTU melalui protokol Modbus TCP/IP dengan media *wireless* yang dipancarkan oleh sebuah router.

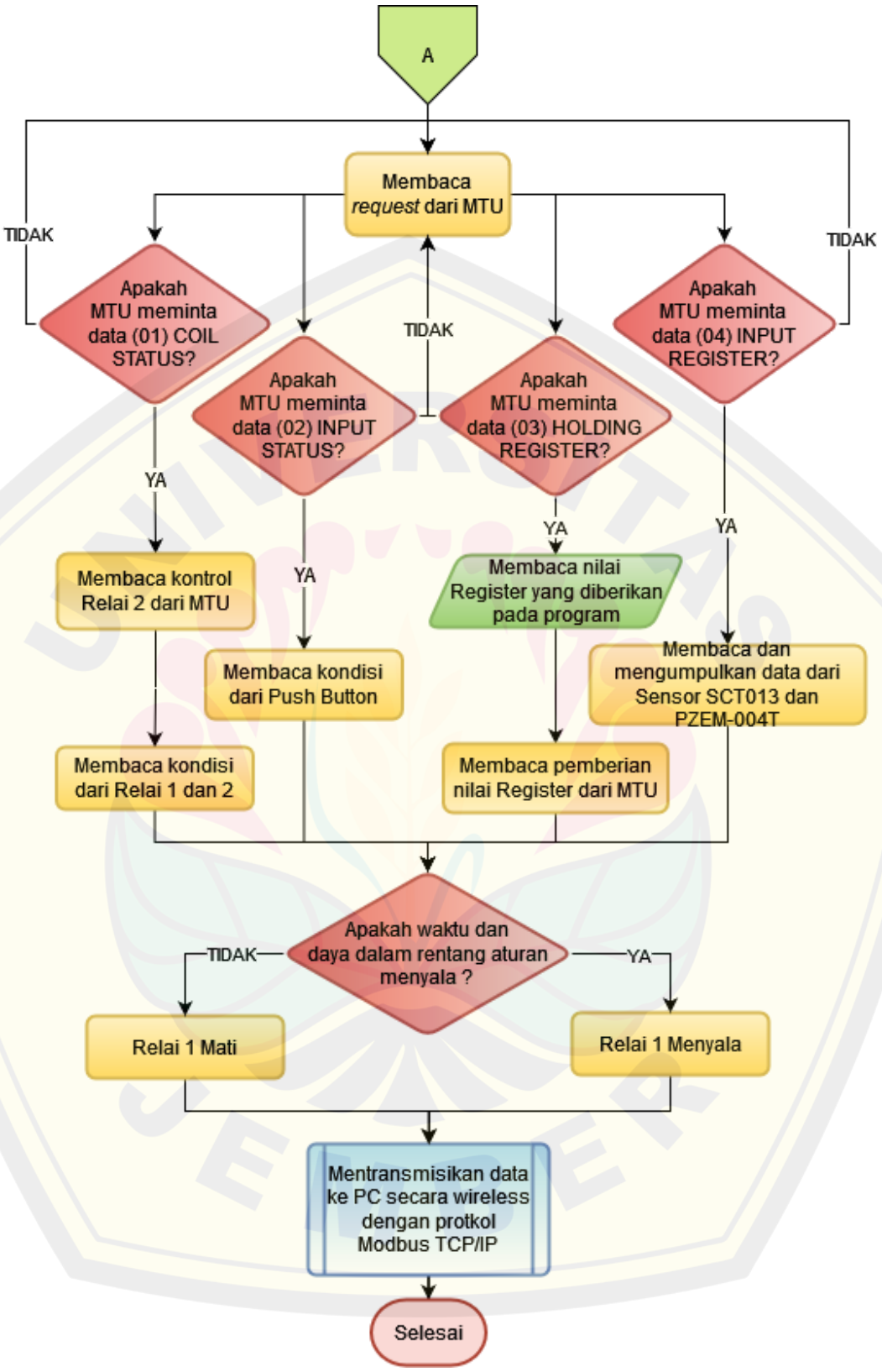
Komunikasi antar mikrokontroler yang terdapat pada RTU menggunakan komunikasi serial. Pembagian sensor pada mikrokontroler ATmega328P-AU dibagi sedemikian rupa untuk mempermudah dalam membuat perancangan program dan mempermudah dalam perancangan rangkaian pada papan sirkuit cetak. Pembagian *monitoring* sensor arus berdasarkan setiap grup pada panel yang terdapat di lantai 3 Gedung B Fakultas Teknik Universitas Jember, sedangkan pembagian relai dibagi dua yaitu untuk pengawasan dan pengontrolan, jadi terdapat satu buah relai yang nyala dan hidupnya dapat dikontrol oleh MTU serta satu buah relai yang nyala dan hidupnya hanya dikontrol oleh RTU sehingga dalam satu buah relai ini MTU hanya bisa mengawasi kondisi dari relai tersebut.

HMI berfungsi sebagai *master* yang berarti sebagai pusat pengontrolan terhadap sistem, *master* akan melakukan *polling* data terhadap *slave* yaitu

mikrokontroler yang terdapat RTU untuk melakukan permintaan data pengukuran sensor arus, data *coil status*, maupun sensor tegangan yang akan selanjutnya akan dikirim kembali ke MTU dan ditampilkan pada HMI dalam bentuk informasi data yang dikirim oleh RTU. Dalam proses pengontrolan HMI akan memberi perintah untuk RTU, kemudian mikrokontroler pada RTU akan mengeksekusi sesuai permintaan dari HMI.



Gambar 3.7 Flowchart Cara Kerja



Gambar 3.8 Flowchart Cara Kerja Lanjutan

Berdasarkan Gambar 3.7 dan Gambar 3.8 dapat diketahui cara kerja dari RTU. Pada Gambar 3.7 terlihat ketika proses pertama RTU mendapat sumber daya maka hal pertama yang dilakukan oleh RTU adalah menginisialisasi beberapa komponen yang digunakan seperti alamat I2C dari LCD Oled, sensor arus, sensor tegangan, *push button*, komunikasi serial antar mikrokontroler, dan pengaturan komunikasi Modbus TCP/IP. Setelahnya maka mikrokontroler ESP8266-01S akan berusaha untuk menghubungkan dengan jaringan wifi yang telah ditentukan sebagai media komunikasi antara MTU dengan RTU. Setelah ESP8266-01S berhasil terhubung dengan wifi tersebut maka LCD oled pada RTU akan menampilkan alamat IP dari ESP8266-01S sebagai alamat IP dari RTU.

Setelah RTU terhubung dengan media komunikasi, maka RTU akan menunggu hingga MTU terhubung ke media komunikasi yaitu jaringan *wireless* yang digunakan oleh RTU. Dalam blok diagram ini menggunakan bentuk *predefined process* pada flowchart karena terdapat proses yang lebih rinci didalamnya dan berada diluar proses dari RTU. Dalam proses ini PC atau komputer sebagai MTU diharuskan membuka software HMI, bisa menggunakan software yang sederhana seperti halnya ModScan kemudian mengatur pada *connection* untuk memasukkan alamat IP dari RTU. Setelah HMI pada PC atau komputer sudah berhasil terhubung dengan RTU maka RTU akan menampilkan alamat IP dari MTU yang terhubung dengan dirinya dan HMI siap melakukan *polling* data terhadap RTU.

Dalam proses *polling* data, MTU akan mengirimkan *request* kepada RTU. Terdapat empat macam *Modbus Point Type* yaitu 01 untuk coil status, 02 untuk input status, 03 untuk holding register, 04 untuk input register. Ketika HMI membuka halaman coil status maka HMI akan meminta RTU mengirimkan data coil status berdasarkan alamat register yang dibutuhkan, dengan demikian RTU akan membaca kondisi dari relai dan ketika HMI meminta untuk mengaktifkan relai berdasarkan alamat register yang diinginkan, maka HMI akan mengirimkan *request* berupa instruksi untuk mengaktifkan relai pada RTU dan dalam waktu ini RTU akan merespon untuk mengaktifkan relai yang terdapat pada RTU dan akan menuliskan data dari coil status ini pada LCD oled yang terdapat pada RTU

sekaligus mengirimkan data respon ke MTU secara *wireless* dengan protokol TCP/IP.

Ketika HMI membuka halaman yang berisikan data dari input status maka MTU akan mengirimkan *request* data pada RTU untuk mengirimkan data input status berdasarkan alamat register yang dibutuhkan oleh MTU. Input status disini adalah kondisi dari input pada RTU yang digunakan yaitu *push button*. Pada saat yang sama RTU akan membaca kondisi dari *push button* yang diinginkan kemudian nilai dari *push button* ini akan ditempatkan pada alamat register yang sudah ditentukan yang kemudian alamat register dan nilainya ini akan ditampilkan pada LCD oled dan akan dikirim ke MTU secara *wireless* dengan protokol TCP/IP.

Ketika HMI membuka halaman yang berisikan data dari holding register maka MTU akan mengirim *request* data pada RTU untuk mengirimkan data dari holding register berdasarkan alamat register yang diinginkan nilainya oleh MTU. Pada saat inilah RTU akan membaca nilai sebelumnya terlebih dahulu, ketika pada saat pertama sistem dihidupkan lagi maka RTU akan membaca pemberian nilai register berdasarkan nilai pemberian awal yang telah didefinisikan di dalam program sesuai alamat yang digunakan, kemudian alamat register dan nilainya akan ditampilkan pada LCD oled dan dikirim ke MTU secara *wireless* dengan protokol TCP/IP. Ketika MTU memberikan nilai pada alamat holding register yang dibutuhkan maka RTU akan merubah nilai sebelumnya yang didefinisikan oleh program menjadi nilai yang diberikan oleh MTU. Namun ketika MTU tidak merubah nilai, maka nilai pada holding register yang dimaksud akan tetap dengan nilai sebelumnya. Kemudian nilai dan alamat dari register ini akan ditampilkan pada LCD oled kemudian dikirim ke MTU secara *wireless* dengan protokol TCP/IP.

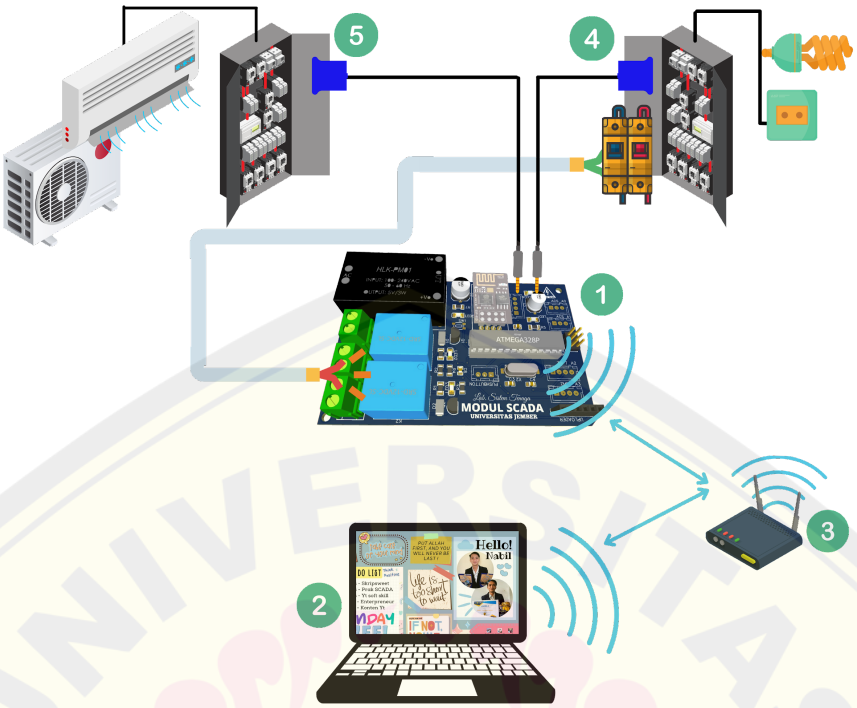
Pada saat HMI membuka halaman yang berisikan data dari input register maka MTU akan mengirimkan sinyal *request* data pada RTU untuk mengirimkan data input register sesuai alamat yang diinginkan oleh MTU dan sesuai alamat yang digunakan pada sistem ini. Input register disini adalah nilai dari input sebuah register dengan alamat register yang digunakan pada sistem ini yaitu nilai dari

sensor-sensor yang digunakan, diantaranya 8 buah sensor arus dan 1 buah sensor tegangan yang nilainya ditempatkan pada input register sesuai dengan alamat yang digunakan. Sehingga pada saat ini RTU akan membaca nilai pembacaan sensor-sensor tersebut kemudian nilainya akan diproses sesuai perhitungan yang dibutuhkan dan hasilnya akan ditempatkan pada alamat register yang sudah ditentukan. Setelah itu RTU akan melihat data waktu dan daya pembacaan yang dibandingkan dengan nilai aturan menyala pada relai 1 dan melakukan pengontrolan pada relai tersebut. Kemudian mengirimkan datanya ke MTU sebagai sinyal respon data dari RTU secara *wireless* dengan protokol TCP/IP.

Seluruh proses tersebut akan secara terus menerus dilakukan berulang sebagai *polling* data oleh MTU terhadap RTU hingga MTU ataupun RTU sudah tidak terhubung lagi dengan media komunikasi yang digunakan yaitu sinyal wifi yang telah ditentukan sebagai media transmisi data dengan protokol TCP/IP. Pada saat ini juga dapat dilihat dari HMI bahwa HMI sudah tidak lagi melakukan *polling* data terhadap RTU dengan salah satu indikatornya yaitu terdapat *error* atau selisih pada jumlah permintaan data oleh MTU dengan jumlah data yang direspon oleh RTU tidak sesuai atau tidak sama dengan yang diminta oleh MTU. Pada RTU juga akan memberikan indikator bahwa RTU dengan MTU sudah tidak lagi melakukan komunikasi dengan adanya tampilan alamat IP dari ESP8266-01S pada LCD oled yang menunjukkan RTU telah melakukan reset otomatis sehingga pada posisi ini RTU akan kembali menunggu MTU terhubung dengan jaringan yang telah ditentukan tersebut.

3.5.2 Rancangan Plant

Pada Gedung B Lantai 3 Fakultas Teknik Universitas Jember terdapat 2 buah panel yaitu panel untuk *air conditioner* (AC) atau mesin penyejuk udara dan panel untuk beban lainnya seperti lampu, *personal computer* (PC), proyektor, dan lainnya. Rancangan plant yang akan dibangun dapat dilihat seperti gambar dibawah ini:



Gambar 3.9 Ilustrasi Sistem

Keterangan:

1. Modul *Remote Terminal Unit* (RTU)
2. *Master Terminal Unit* (MTU)
3. Router WiFi
4. Panel untuk penerangan dan kotak kontak
5. Panel untuk *air conditioner* (AC)

Berdasarkan gambar 3.9 diketahui bahwa MTU terhubung dengan RTU melalui router yang memancarkan sinyal WiFi sebagai media komunikasi. Kemudian RTU dihubungkan dengan panel yang terdapat di lantai 3 Gedung B Fakultas Teknik Universitas Jember. Panel yang sebelah kanan atau yang tertulis dengan nomor empat pada Gambar 3.9 merupakan panel penerangan dan beban lainnya yang terhubung dengan kotak kontak sedangkan panel sebelah kiri merupakan panel untuk AC. Pada panel penerangan dan beban kotak kontak tersebut akan diletakkan sensor tegangan beserta sensor arus pada setiap grup. Pada panel penerangan dan kotak kontak tersebut terdapat tiga grup yang masih

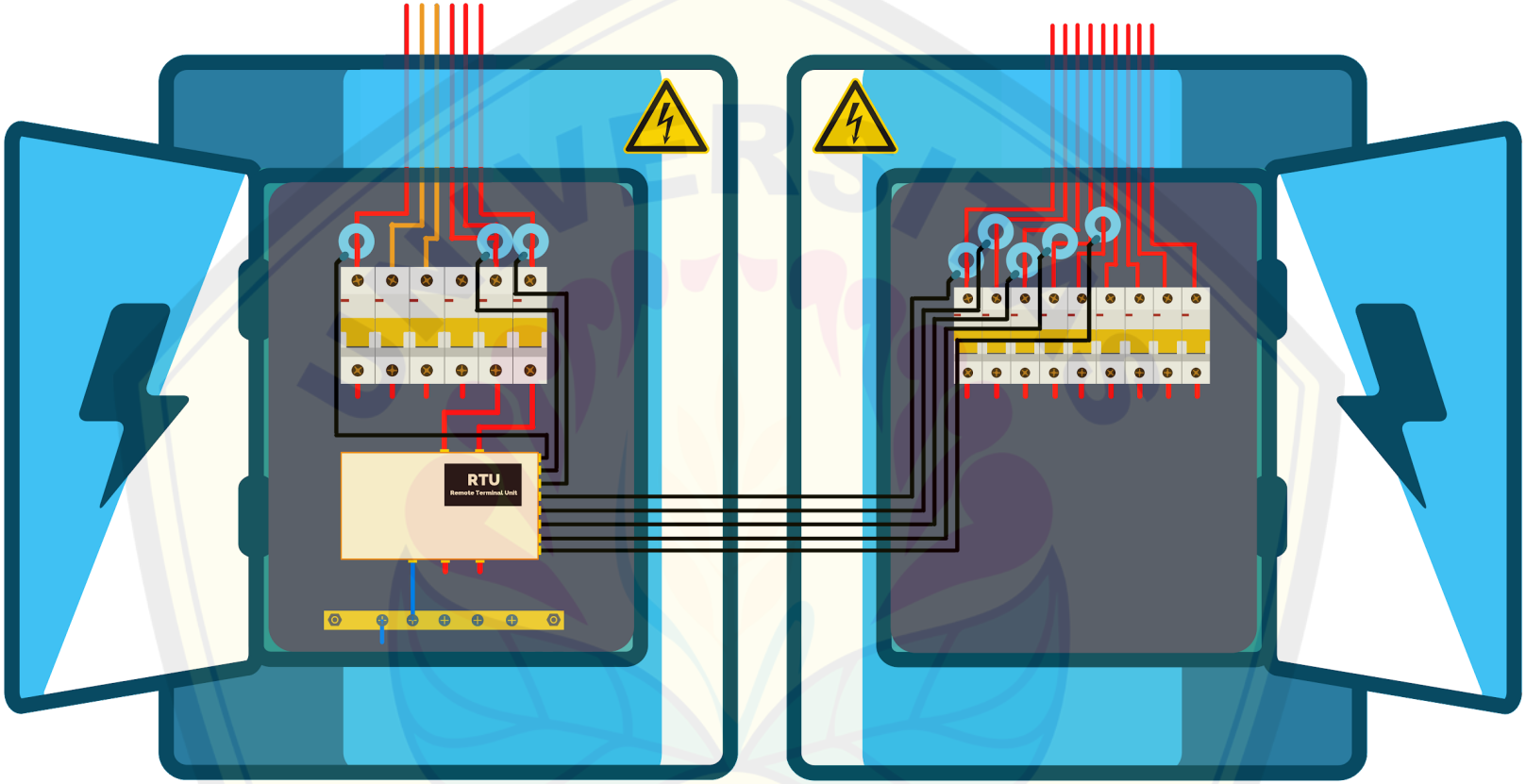
berfungsi sehingga akan ditempatkan sensor arus sebanyak tiga buah dan pada panel tersebut juga akan dipasangkan dua buah relai yang digunakan sebagai pengontrolan dalam sistem SCADA ini. Berhubungan dengan jumlah sensor arus yang akan dihubungkan dengan RTU sebanyak delapan buah maka pada panel untuk beban AC akan diletakkan sensor arus sebanyak lima buah. Data yang diperoleh oleh RTU akan diproses terlebih dahulu oleh mikrokontroler utama kemudian akan dikirim kepada MTU secara *wireless* dengan protokol Modbus TCP/IP.

3.5.3 Rancangan *Remote Terminal Unit* (RTU)

Dalam perancangan *Remote Terminal Unit*(RTU) terdapat beberapa blok subsistem yang harus dipersiapkan dengan tujuan untuk mempermudah dalam proses realisasinya serta dapat meminimalisir terjadinya error pada RTU.

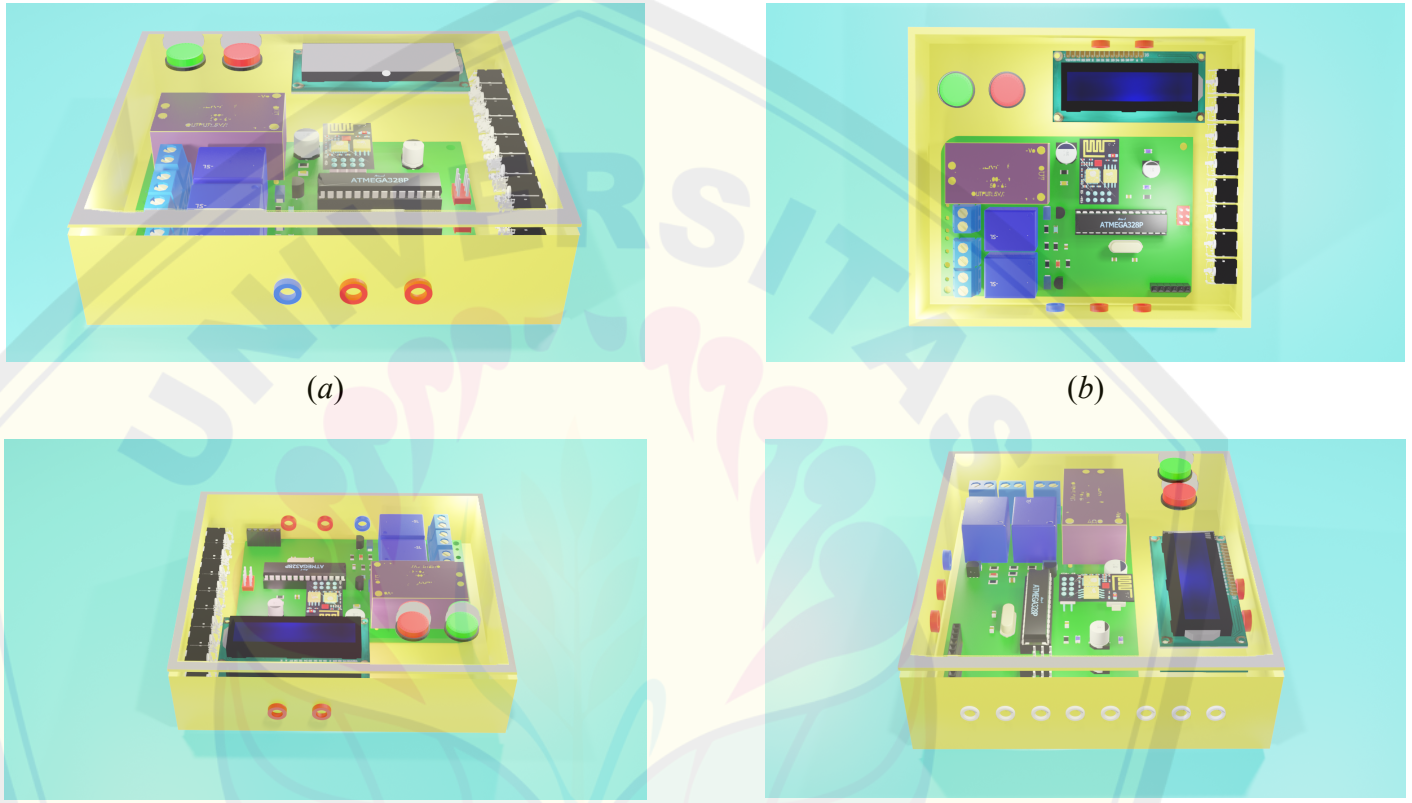
a. Rancangan Desain *Remote Terminal Unit* (RTU)

Remote Terminal Unit (RTU) ini dirancang dilengkapi dengan mikrokontroler sebagai pusat kendalinya yang dilengkapi dengan 8 sensor arus dan 1 sensor tegangan sebagai masukan data hasil pengukuran dari beban setiap grup. Modul RTU yang dibangun ini menggunakan protokol komunikasi Modbus TCP/IP berbasis *wireless*. Berikut merupakan wiring RTU ketika dipasang pada beban panel lantai 3 Gedung B Fakultas Teknik Universitas Jember:



Gambar 3.10 Wiring RTU Pada Panel

Modul RTU didesain berbentuk box yang dicetak menggunakan 3D *printing*, sedangkan untuk penutup atas box akan menggunakan akrilik putih transparan. Hal ini bertujuan agar bagian dalam RTU dapat terlihat dari sisi atas. Pada bagian atas terdapat 2 tombol tekan yang digunakan untuk mengontrol serta 1 LCD Oled yang berfungsi untuk menampilkan data. Bagian samping kanan modul terdapat 8 socket female yang digunakan untuk menghubungkan sensor arus dengan setiap grup pada panel. Pada bagian belakang modul terdapat 2 socket female yang digunakan sebagai output yang terhubung dengan input MCB. Sedangkan pada bagian depan modul terdapat 3 buah socket female yang digunakan sebagai power RTU tersebut dan digunakan untuk keperluan pengukuran pada sensor tegangan serta digunakan untuk pengontrolan pada grup tertentu yang akan dihubungkan dengan relai. Adapun lebih jelasnya dapat dilihat pada gambar rancangan 3D modul RTU berikut:



(*a*) (*b*)

(*c*) (*d*)

(a) Tampak Depan; (b) Tampak Atas; (c) Tampak Belakang;
(d) Tampak Samping

Gambar 3.11 Rancangan Desain 3D dengan Blender

b. Rancangan Kode Program RTU

Mikrokontroler diprogram supaya mampu menjalankan fungsinya sebagai RTU, kode program ini meliputi pembuatan perpustakaan program, program inisialisasi, program konfigurasi komunikasi, pembacaan sensor dan pengolahan data sensor yang nantinya akan dikirimkan kepada MTU, serta program pengalamatan pengiriman data.

1) Rancangan *Library*

Library atau jika diubah dalam bahasa Indonesia yang berarti perpustakaan. Dalam pemrograman Arduino, *library* merupakan sekumpulan program yang ditata untuk mempermudah dalam pemberian perintah terhadap suatu komponen atau aktuator ataupun proses agar dapat bekerja sesuai dengan fungsi yang diinginkan.

Adapun fungsi perancangan *library* pada penelitian ini yaitu sama seperti pada umumnya untuk mempercepat dan mempermudah penulis dalam menyusun program pada Arduino IDE. Sehingga dapat mempersingkat kode program pada Arduino IDE dan mempermudah orang lain untuk memahami kode program yang telah ditulis. Dengan adanya perancangan *library* secara individu inilah diharapkan proses RTU untuk mengolah data dapat dilakukan dengan lebih cepat, karena isi dari *library* dituliskan khusus hanya untuk proses yang dibutuhkan dalam pemrosesan data pada RTU. Dalam perancangan *library* pada penelitian ini dilakukan dengan membuat 3 buah file, yaitu file dengan format *.cpp*, *.h*, dan *keywords.txt*.

a) Library Untuk Sensor PZEM-004T

(1) *Source file* (*.cpp*)

```
#include "Nabil_PZEM.h"
#include <stdio.h>
```

```
Nabil_PZEM::Nabil_PZEM(HardwareSerial*  port,  uint8_t
addr)
{
     port->begin(PZEM_BAUD_RATE);
     this->_serial = port; this->_isSoft = false;
     init(addr);
}
```

(2) *Header File* (*.h)*

```
class Nabil_PZEM
{
public:
     Nabil_PZEM(HardwareSerial*     port,     uint8_t
addr=PZEM_DEFAULT_ADDR);
     ~Nabil_PZEM();
     float voltage(); float current(); float power();
     float pf();
```

(3) File Kata Kunci (*keywords.txt)*

```
Nabil_PZEM    KEYWORD1
Nabil    KEYWORD1
PZEM    KEYWORD1
voltage    KEYWORD2
current    KEYWORD2
power    KEYWORD2
pf    KEYWORD2
PZEM_DEFAULT_ADDR LITERAL1
```

b) Library Untuk Sensor YHDC-SCT013

(1) *Source file* (*.cpp*)

```
#include "Nabil_SCT.h"
void ArusLibNabil::current(unsigned int _inPinI, double
_ICAL)
{inPinI = _inPinI; ICAL = _ICAL; offsetI =
ADC_COUNTS>>1;}
```

(2) *Header File (.h)*

```
class ArusLibNabil
{
  public:
     void current(unsigned int _inPinI, double _ICAL);
     double calcIrms(unsigned int NUMBER_OF_SAMPLES);
     long readVcc();
     double Irms;
};
```

(3) File Kata Kunci (*keywords.txt*)

```
ArusLibNabil    KEYWORD1
current    KEYWORD2
currentTX    KEYWORD2
calcIrms    KEYWORD2
Irms    LITERAL1
```

Kode program *library* diatas merupakan rancangan untuk 2 buah sensor yaitu sensor PZEM-004T dan sensor YHDC-SCT013. Setiap pembuatan *library* diatas terdiri dari 3 file yang digunakan, yaitu *source file* (.cpp), *header file* (.h), dan kata kunci (*keyword.txt*). Kode program diatas dirancang khusus untuk memenuhi tugas dari RTU, sehingga dengan rancangan kode tersebut diharapkan

RTU dapat lebih fokus dalam melaksanakan tugasnya, dapat menghemat memori dari mikrokontroler, dan dapat mempercepat tugas dari mikrokontroler tersebut.

2) Program Inisialisasi

Pada bagian pertama, sebuah program diawali dengan inisialisasi terlebih dahulu yang dibutuhkan untuk pembacaan sensor, pengolahan data serta komunikasinya. Program rancangan program disini terdapat inisialisasi untuk sensor PZEM-004T yang digunakan untuk pembacaan nilai tegangan, arus pada beban pertama, dan cos phi beserta inisialisasi untuk sensor YHDC-SCT013 yang digunakan untuk pembacaan arus. Selain inisialisasi sensor, pada kode program tersebut juga terdapat inisialisasi untuk komunikasi serial dengan mikrokontroler ESP8266-01, dimana komunikasi dilakukan via Serial dan melalui Software Serial. Untuk *library* komunikasi Serial tidak menggunakan SoftwareSerial namun *library* yang digunakan yaitu NeoSWSerial, karena library ini mampu melakukan transmisi dan menerima data secara bersamaan tanpa mematikan jalur komunikasi lainnya. Hal ini dibutuhkan karena pada mikrokontroler ATmega disini terdapat 2 komunikasi Serial yang harus dilakukan, yaitu komunikasi Serial dengan sensor PZEM-004T dan komunikasi Serial dengan mikrokontroler lainnya. Dikarenakan ATmega328P-AU hanya menyediakan 1 *Hardware Serial* dan jalur tersebut juga digunakan untuk transfer kode program, selain itu *Software Serial* juga hanya dapat melakukan 1 buah pasang *Software Serial* saja, sehingga digunakan model komunikasi seperti yang sudah dijelaskan sebelumnya. ArusLibNabil merupakan kode program inisialisasi dari *class* fungsi yang telah dibuat pada library untuk sensor YHDC-SCT013 yang digunakan untuk membaca arus. Jumlah dari inisialisasi ArusLibNabil sesuai dengan jumlah sensor arus yang digunakan pada penelitian ini yaitu 7 buah sensor arus clamp YHDC-SCT-013 dan pin analog yang digunakan yaitu sesuai pada program inisialisasi diatas yaitu dimulai dari pin A0 hingga A6. Hasil pembacaan dari masing-masing sensor diatas akan disimpan pada variabel dan disesuaikan dengan nama besarannya yang ditunjukkan pada program inisialisasi diatas dengan tipe data yang digunakan yaitu *double* untuk menyimpan bit data yang lebih banyak.

Pada program inisialisasi pada mikrokontroler ESP8266 pada baris pertama dan kedua yaitu inisialisasi Wire dengan SPI yang merupakan inisialisasi dari komunikasi I2C dengan SPI yang digunakan untuk komunikasi mikrokontroler dengan LCD Oled. Kemudian inisialisasi untuk LCD Oled yang digunakan yaitu menggunakan *library* dari Adafruit yaitu Adafruit_GFX dan Adafruit_SSD1306. Selain itu adapun *library* program yang digunakan untuk komunikasi dengan protokol Modbus yaitu *library* bawaan dari ESP8266WiFi dan ModbusIP_ESP8266. Inisialisasi selanjutnya yaitu inisialisasi untuk variabel-variabel yang digunakan untuk keperluan pemrograman selanjutnya. Kemudian inisialisasi terakhir pada bagian awal dari program yang digunakan pada mikrokontroler ESP8266 tersebut adalah inisialisasi untuk berbagai pin dan setting model yang diperlukan untuk LCD Oled.

3) Program Konfigurasi Komunikasi

Supaya RTU dapat berkomunikasi dengan MTU maka dibutuhkan sebuah konfigurasi untuk mengatur jalannya komunikasi sesuai dengan protokol yang digunakan. Pada program konfigurasi yang utama disini merupakan pembuatan program untuk mengatur koneksi dan identitas agar dapat terdeteksi pada satu jaringan yang sama. Kode program yang dirancang sebagai berikut:

```
bool cbConn(IPAddress ip_master)
    {
      Serial.write(terhubung);
      oled.clearDisplay();
      oled.setTextColor(SSD1306_WHITE);
      oled.setTextSize(1);
      oled.setCursor(13,15);
      oled.println(F("Berhasil Terhubung"));
      oled.setCursor(35,25);
      oled.println(F("IP Master ="));
      oled.setCursor(20,35);
      oled.println(ip_master);
```

```
      oled.display();
      delay(2750);
      oled.clearDisplay();
      tampilkan_logo_lab();
      return true;
    }
WiFi.begin("MODUL SCADA", "tenagasukses");
    while (WiFi.status() != WL_CONNECTED)
      { delay(250);}
mbTCPWifi.onConnect(cbConn);
```

Kode program diatas merupakan program untuk melakukan konfigurasi awal untuk keperluan komunikasi antara RTU dengan MTU. Sebuah fungsi dengan tipe data *boolean* yang diberi nama `cbConn` berfungsi untuk mengembalikan nilai "*true*' ketika RTU sudah berhasil terhubung dengan router yang telah ditetapkan sebagai media komunikasi. Kemudian WiFi.begin merupakan program untuk memerintahkan mikrokontroler menghubungkan dengan wifi yang mempunyai SSID "`MODUL SCADA`" dengan kata sandi wifi yaitu "`tenagasukses`". Kode program while merupakan kode program yang akan terus diulang ketika mikrokontroler masih tidak terhubung dengan wifi yang dimaksud. Kemudian program bagian akhir tersebut merupakan program yang memanggil fungsi yang telah dijelaskan sebelumnya ketika mikrokontroler telah terhubung dengan jaringan wifi yang telah ditentukan.

4) Program Pembacaan Sensor dan Pengolahan Data Sensor

Bagian terpenting selain konfigurasi komunikasi pada mikrokontroler adalah perancangan program untuk melakukan pembacaan sensor beserta proses pengolahan data dari hasil pembacaan tersebut. Sehingga hasil dari data yang telah diolah tersebut telah siap digunakan untuk proses selanjutnya. Dalam melakukan hal tersebut, program dirancang sebagai berikut:

```
{
```

```
voltage = pzem.voltage();
pf = pzem.pf();
      currentpzem = pzem.current();
      power = pzem.power();
//Clamp
  ArusRms1=kalibrasi(simpanArusRms1);
  DayaAktifClamp1 = ArusRms1*voltage*pf;
double kalibrasi(double nilai)
  {
    double hasil = ((nilai*1.084)-1.036)+1;
    return hasil;
  }
```

Dari program diatas merupakan program perancangan proses pembacaan sensor beserta fungsi untuk melakukan pengolahan data sensor. Nilai dari hasil pembacaan tegangan akan disimpan pada variabel "`voltage`", untuk hasil pembacaan arus pada sensor PZEM-004T akan disimpan pada variabel "`currentpzem`", untuk hasil pembacaan faktor daya pada sensor PZEM-004T akan disimpan pada variabel "`pf`", sedangkan hasil pembacaan daya pada sensor PZEM-004T akan disimpan pada variabel "`power`". Setelah program pembacaan sensor untuk PZEM-004T, dibawahnya adalah program untuk pembacaan sensor YHDC-SCT013. Nilai yang didapatkan dari hasil pembacaan sensor tersebut disimpan dalam variabel dengan tipe data *double*. Konstanta yang bernilai `1480` pada program diatas adalah nilai kalibrasi dari konversi nilai tegangan pada pembacaan ADC dari mikrokontroler tersebut. Nilai yang didapatkan dari proses pembacaan arus tersebut kemudian akan dikalibrasi yang ketiga kalinya yaitu dengan memanggil fungsi dengan nama "`kalibrasi`" yang kemudian nilainya akan digunakan dalam proses mencari besar daya pada jalur tersebut. Kemudian seluruh nilai setelah proses pengolahan data tersebut tersebut akan dikirimkan pada MTU.

5) Program Pengiriman dan Penerimaan Data

Proses setelah pembacaan sensor dan pengolahan data sensor yaitu pengiriman nilai-nilai tersebut kepada MTU. Nilai yang akan dikirim dari hasil pengolahan data yaitu nilai untuk besaran tegangan, arus, daya, beserta nilai kondisi dari tombol tekan yang digunakan sebagai kode fungsi *input status*. Dalam melaksanakan proses tersebut, program dirancang sebagai berikut:

```
mbTCPWifi.Ireg(SENSOR_IREG,  tegangan);
mbTCPWifi.Ireg(SENSOR_IREG2,  arusPZEM);
mbTCPWifi.Ireg(SENSOR_IREG3,  dayaPZEM);
mbTCPWifi.Ireg(SENSOR_IREG4,  arusClamp1);
mbTCPWifi.Ireg(SENSOR_IREG5,  dayaClamp1);

mbTCPWifi.Ists(SWITCH_ISTS ,k);
mbTCPWifi.Ists(SWITCH_ISTS2,l);
```

Setelah diketahui bahwa nilai hasil dari pengolahan data sensor disimpan pada variabel yang telah dijelaskan sebelumnya, maka nilai tersebut yang akan dikirim pada MTU. Dalam proses pengiriman data, maka terdapat alamat yang menjadi tujuan. Alamat tujuan tersebut akan dijelaskan pada subbab lain yaitu pada pengalamatan pengiriman data. Pada kode program diatas terdapat `mbTCPWifi.Ireg`, kode program tersebut merupakan kode untuk komunikasi modbus dan datanya akan dikirimkan pada kode fungsi *input register* dengan pola penulisan `mbTCPWifi.Ireg(alamat, nilai)`. Sehingga pada perancangan kode program diatas akan mengirim nilai hasil pembacaan sensor pada kode fungsi *input register*, sedangkan nilai hasil pembacaan tombol tekan akan dikirimkan pada kode fungsi *input status* yang ditunjukkan pada program `mbTCPWifi.Ists(alamat ,nilai)`. Program ini akan mengirimkan nilai pada MTU yang bernilai 0 atau 1, nilai 1 menunjukkan bahwa kondisi tombol tidak sedang ditekan sedangkan nilai 0 menunjukkan bahwa kondisi tombol sedang ditekan.

6) Program Kontrol RTU

Seperti pada idealnya pada sebuah RTU, terdapat *output* yang nyala atau matinya hanya dapat dibaca oleh MTU dan terdapat *output* lainnya yang nyala maupun matinya dapat dikontrol oleh MTU. Dikarenakan pada RTU terdapat 2 mikrokontroler yang mempunyai tugas masing-masing berbeda, sehingga perancangan kode program pada proses ini sebagai berikut:

```
int save=0;
uint8_t pin_coil2[] = {valrelay2};
#define banyak_data_coil2
sizeof(pin_coil2)/sizeof(uint8_t)
//Fungsi callback, yg ini MASTER BISA MENULIS
uint16_t cobaRead_coil2(TRegister* reg, uint16_t val)
    {
      oled.print(F("       "));
      oled.print(reg->address.address);
      oled.print(F("    "));
      if(val>0) save=1;
      else if(val==0) save=0;
      oled.println(save);
      oled.display();
      Serial.write
        (
          save   //Untuk Coil Status
        );
      return COIL_VAL(val);
    }

//Fungsi Callback, yg ini MASTER BISA MENULIS
uint16_t cobaWrite_coil2(TRegister* reg, uint16_t val)
    {
      Serial.write
        (
```

```
        kontrol    //Untuk Coil Status
      );
    return COIL_VAL(val);
  }
```

Pada kode program diatas merupakan program untuk penerimaan data kontrol, setelah data kontrol oleh MTU diterima maka proses selanjutnya yaitu mengeksekusi *output* atau aktuator untuk dinyalakan maupun dimatikan sesuai dengan data kontrol yang diberikan oleh MTU. Program penerimaan data kontrol dilakukan oleh fungsi yang mempunyai nama "`cobaRead_coil2`", kemudian nilai yang diperoleh disimpan pada variabel yang bernama "`val`". Kemudian nilai dari "`val`" ini dikonversi menjadi nilai 0 atau 1 dan selanjutnya disimpan pada variabel "`save`". Kemudian nilai pada variabel "`save`" inilah yang digunakan untuk mengeksekusi sebuah *output* pada RTU dengan cara mengirimkan data nilainya melalui komunikasi serial dengan kode program yang digunakan yaitu `Serial.write(nilai)`.

7) Program Parsing Data

Dalam proses komunikasi dilakukan pengiriman maupun penerimaan data yang jumlahnya cukup banyak, sehingga untuk mempercepat proses komunikasi yang mewakili proses pengiriman maupun penerimaan data tersebut penulis merancang sebuah kode program untuk melakukan parsing data. Kode program dirancang seperti berikut:

```
void kirim_data()
 {
   DataESP.println
    (
      String('^')
      + String(voltage)
      + String(" ")
      + String(currentpzem)
```

```
        + String(" ")
        + String(power)
        + String(" ")
        + String(ArusRms1)
        + String(" ")
        + String(DayaAktifClamp1)
        + String(" ")
        + String(ArusRms2)
      );
  }
```

Program diatas merupakan fungsi dari program untuk komunikasi data pada mikrokontroler ATmega828P-AU. Pada fungsi "`kirim_data()`" merupakan fungsi yang digunakan untuk mengirimkan data, data yang dikirimkan menggunakan tipe data *string*. Pemilihan tipe data ini bertujuan untuk mempertahankan angka dari nilai yang akan dikirimkan, sehingga tidak ada kekeliruan dalam proses pengiriman data. Pengiriman data dilakukan secara bersamaan dalam satu kali pemanggilan fungsi tersebut sehingga dapat mempersingkat waktu yang dibutuhkan untuk mengirim banyak data. Masing-masing data dipisahkan oleh spasi "`String(" ")`". Data yang dikirim merupakan data tegangan, arus, dan daya dari sensor pertama hingga ke 7. Selain itu terdapat data lainnya yang dikirimkan yakni data pembacaan tombol dan data dari kondisi relai. Sedangkan pada fungsi "`ambil_data()`" merupakan program pada mikrokontroler ATmega untuk melakukan pengecekan data ketika terdapat perintah yang diberikan oleh MTU pada RTU. Data yang akan diambil pada fungsi ini meliputi data untuk eksekusi kontrol relai dan data untuk menghidupkan buzzer, dimana buzzer hanya aktifkan sejenak ketika terdapat perintah pengontrolan oleh MTU. Kemudian adapun kode program untuk parsing data pada mikrokontroler ESP8266 sebagai berikut:

```
void parsing_data()
  {
```

```
if (Serial.available())
  {
    if(Serial.find('^'))
      {
        delay(25);
        tegangan=Serial.parseFloat();
        arusPZEM=Serial.parseFloat();
        dayaPZEM=Serial.parseFloat();
          arusClamp1=Serial.parseFloat();
          dayaClamp1=Serial.parseFloat();
      }}}
```

Program diatas adalah program parsing data pada mikrokontroler ESP8266. "`if (Serial.available())`" merupakan program untuk mengetahui ketika ketika terdapat data yang masuk melalui serial. Kemudian "`if(Serial.find('^'))`" merupakan kode program yang digunakan untuk melihat ketika data yang masuk pada serial terdapat tanda "^" yang telah diatur untuk menunjukkan bahwa tanda tersebut adalah tanda awalan untuk data yang dikirim oleh ATmega, dimana ketika terdapat tanda tersebut maka mikrokontroler akan melanjutkan proses selanjutnya yaitu mengambil data dengan cara parsing datanya sesuai dengan tipe data yang digunakan. "`parseFloat()`" merupakan parsing data untuk tipe data *float*, sedangkan "`Serial.parseInt()`" merupakan kode program parsing data untuk tipe data *integer*.

8) Pengalamatan Pengiriman Data

Pengiriman data antara MTU dengan RTU menggunakan komunikasi protokol Modbus dengan format pengiriman data yang telah dijelaskan pada bab sebelumnya. Setelah konfigurasi komunikasi antara RTU dengan MTU terhubung, maka proses selanjutnya yaitu proses pembacaan sensor dan kondisi relai. Pembacaan sensor pada RTU dilakukan pada pin serial RX-TX dan pin analog, sedangkan untuk relai berada pada pin digital. Nilai yang telah dikumpulkan

selanjutnya akan dikirim oleh RTU, alokasi pengalamatan untuk menjalankan fungsi-fungsi yang dimaksud dapat dilihat pada tabel berikut.

Tabel 3.5 Alokasi Pengalamatan pada RTU

Data	Alamat RTU	Alamat Modbus	Aksi
Relai1	pin_coil[]	00001	Read
Relai2	pin_coil2[]	00002	Write
Push Button1	SWITCH_ISTS = 0	10001	Read
Push Button2	SWITCH_ISTS2 = 1	10002	
Tegangan	SENSOR_IREG = 0	30001	Read
Arus PZEM	SENSOR_IREG2 = 1	30002	Read
Daya PZEM	SENSOR_IREG3 = 2	30003	
Arus Clamp1	SENSOR_IREG4 = 3	30004	
Daya Clamp1	SENSOR_IREG5 = 4	30005	
Arus Clamp2	SENSOR_IREG6 = 12	30013	
Daya Clamp2	SENSOR_IREG7 = 13	30014	
Arus Clamp3	SENSOR_IREG8 = 14	30015	
Daya Clamp3	SENSOR_IREG9 = 15	30016	
Arus Clamp4	SENSOR_IREG10 = 16	30017	
Daya Clamp4	SENSOR_IREG11 = 24	30025	
Arus Clamp5	SENSOR_IREG12 = 25	30026	
Daya Clamp5	SENSOR_IREG13 = 26	30027	Read
Arus Clamp6	SENSOR_IREG14 = 27	30028	
Daya Clamp6	SENSOR_IREG15 = 28	30029	
Arus Clamp7	SENSOR_IREG16 = 36	30037	
Daya Clamp7	SENSOR_IREG17 = 37	30038	
Holding Register1	Hreg[0]	40001	Write
Holding Register2	Hreg[1]	40002	

Data	Alamat RTU	Alamat Modbus	Aksi
Holding Register3	Hreg[2]	40003	Write
Holding Register4	Hreg[3]	40004	
Holding Register5	Hreg[4]	40005	
Holding Register6	Hreg[5]	40006	

Pada tabel 3.5 dapat diketahui alokasi pengalamatan tiap parameter, sehingga rancangan alokasi data pada tabel itulah yang digunakan sebagai acuan dalam pembuatan *address tag* pada pemrograman RTU maupun HMI agar data yang ditampilkan oleh HMI sesuai dengan nilai yang diberikan oleh RTU. *Address tag* mempunyai fungsi untuk mengorganisir pengalamatan yang digunakan pada komponen-komponen yang dibangun pada HMI. Sehingga perancangan kode program pada RTU dapat dilihat sebagai berikut:

```
//Penempatan alamat register untuk masing2 data dari
sensor
    const int SENSOR_IREG   = 0;      //tegangan
    const int SENSOR_IREG2  = 1;      //arus PZEM
    const int SENSOR_IREG3  = 2;      //daya PZEM
  long TimerSampling;
    #define banyaknya_Ireg 100
    #define nilai_offset_Ireg 0
      uint16_t proses_Ireg(TRegister* regIreg, uint16_t
valIreg) {return valIreg; }
//>>>>SETTINGAN UNTUK COIL STATUS
  for (uint8_t i = 0; i <
banyak_data_coil+banyak_data_coil2; i++)
//Setting Coil yg digunakan
  mbTCPWifi.addCoil(COIL_BASE, COIL_VAL(false),
banyak_data_coil+banyak_data_coil2);
```

```
//>>>>>SETTINGAN UNTUK INPUT STATUS
  mbTCPWifi.addIsts(SWITCH_ISTS);
  mbTCPWifi.addIsts(SWITCH_ISTS2);

//>>>>>COBA MEMBUAT SETTINGAN CALLBACK UNTUK INPUT
STATUS
mbTCPWifi.onGetIsts(nilai_offset, proses_Ists,
banyaknya_Ists);

//>>>>>SETTINGAN UNTUK HOLDING REGISTER
        if (!mbTCPWifi.addHreg(0, 0xF0F0,
banyak_data_HoldingRegister)) Serial.println("Error");
        mbTCPWifi.onGetHreg(0, cbRead_HoldingRegister,
banyak_data_HoldingRegister);
//>>>>>SETTINGAN UNTUK INPUT REGISTER
//Tambahkan alamat register sensor - pakai addIreg()
untuk Analog Input
  mbTCPWifi.addIreg(SENSOR_IREG);
  mbTCPWifi.addIreg(SENSOR_IREG2);
  mbTCPWifi.addIreg(SENSOR_IREG3);
```

Program diatas merupakan perancangan kode program untuk pengalamatan pengiriman data. "`mbTCPWifi`" merupakan inisialisasi untuk komunikasi alamat Modbus. Penginisialisasian untuk kode fungsi *input register* pada program tersebut menggunakan "`SENSOR_IREG`", untuk menambahkan alamat yang digunakan menggunakan kode program "`mbTCPWifi.addIreg`". Penginisialisasian untuk kode fungsi *input status* pada program tersebut menggunakan "`SWITCH_ISTS`", untuk menambahkan alamat yang digunakan menggunakan kode program "`mbTCPWifi.addIsts`". Penginisialisasian untuk kode fungsi *holding register* pada program tersebut menggunakan "`banyak_data_HoldingRegister`" yang menunjukkan bahwa alamat pada *holding register* ditempatkan secara berurutan dan tidak terpisah dengan nilai

yang telah diberikan, untuk menambahkan alamat yang digunakan menggunakan kode program "`mbTCPWifi.addHreg`". Penginisialisasian untuk kode fungsi *coil status* pada program tersebut menggunakan "`banyak_data_coil`", untuk menambahkan alamat yang digunakan menggunakan kode program "`mbTCPWifi.addCoil`".

9) Perancangan Program Kontrol Otomatis

Supaya RTU dapat menjalankan sebuah kinerja yang diharapkan mampu mematikan maupun menyalakan sebuah beban secara otomatis maka dibutuhkan sebuah parameter untuk menjalankan otomatisasi ini. Parameter yang diterapkan pada RTU adalah parameter waktu setempat dan daya maksimum. Dengan adanya fitur ini diharapkan mampu mempermudah pengguna dalam manajemen energi yang digunakan. Kode program dirancang sebagai berikut:

```
#include <NTPClient.h>
#include <WiFiUdp.h>
WiFiUDP ntpUDP;
NTPClient ambilWaktu(ntpUDP, "pool.ntp.org", 25204);
  //Waktu Indonesia bagian Barat = UTC+7, dikonversi
detik = 25200

if(millis()-kirimJam>=5000)
     {
     Serial.println(String('*')+
     String(jam)+String(" ")+String(menit)+String("
")+String(detik));
     kirimJam=millis();
     }
```

Program diatas merupakan perancangan kontrol otomatis yang terdapat pada ESP8266. Pengambilan data waktu pada RTU dirancang dengan mengambil data waktu pada server NTP (*network time protocol*) sehingga dapat dilihat bahwa pada baris pertama program diatas memasukkan inisialisasi client dari NTP. Pada

penelitian ini tidak menggunakan RTC (*real time clock*) untuk mengambil data tanggal maupun waktu, dikarenakan chip pada RTC tidak akurat sehingga penggunaan RTC ini menyebabkan pengguna harus menyesuaikan waktu secara manual secara terus menerus dengan cara mengupload ulang program dengan penyesuaian waktu yang baru. Sehingga pada penelitian ini penulis memilih untuk menggunakan pengambilan data waktu menggunakan server dari NTP, dengan demikian jika akan dikembangkan menjadi banyak modul yang harus tersinkronisasi waktunya antara modul satu dengan modul lainnya akan menjadi mudah untuk dikembangkan. Kemudian pada baris selanjutnya yaitu ambil waktu disitu terdapat penggunaan waktu UTC, dimana UTC (*Universal Time Coordinated*) adalah waktu standar dari seluruh dunia sehingga tidak ada perbedaan waktu. Sehingga pada program tersebut NTP mengatur sebagai zona waktu yang diatur berdasarkan UTC. Penelitian ini dilakukan bertempat di Jember sehingga zona waktu yang digunakan pada penelitian ini yaitu zona waktu dari UTC+7, yang artinya waktu di Jember yaitu waktu dari UTC ditambah dengan 7 jam. Nilai 7 jam ini dikonversikan menjadi detik untuk dijalankan dalam Arduino IDE yang menjadi 25200 seperti pada program diatas. nilai yang dimasukkan pada program menjadi 25204 dimana nilai ini adalah nilai kalibrasi pada RTU karena RTU membutuhkan waktu sekitar 4 detik ketika RTU pertama kali diberi daya. Kemudian pada baris selanjutnya yaitu mengenai program holding register tersebut merupakan struktur program untuk pengontrolan berdasarkan daya maksimum, dimana program tersebut akan mengambil nilai daya maksimum yang diberikan pada register alamat 00001 yang pada kode program berarti 0 seperti program diatas, kemudian nilai yang didapatkan tersebut akan dikirimkan melalui komunikasi serial untuk melanjutkan eksekusi pada relai. Kemudian untuk pengiriman atau update data waktu pada atmega berada pada baris program selanjutnya yaitu setiap 5 detik sekali, update data pada atmega dilakukan dengan cara mengirimkan data waktu melalui komunikasi serial dengan format data menggunakan tipe data string seperti pada program yang ditunjukkan diatas. Program pada atmega dapat dilihat sebagai berikut:

```
if(DataESP.find('*'))
          {
          jam=DataESP.parseFloat();
          menit=DataESP.parseFloat();
          detik=DataESP.parseFloat();
          }
if((jam>=21 && menit>=59 && detik>=10) ||
power>=kontrolDaya || jam>=22 || (jam<05||jam<5))
     {
     digitalWrite(Relay1, LOW);
     kondisiR1=0;
     }
  else if((jam>=05 || jam>=5) && (jam<21 || (jam==21 &&
menit<59 && detik<=59)) && power<kontrolDaya)
     {
     digitalWrite(Relay1, HIGH);
     kondisiR1=1;
     }
```

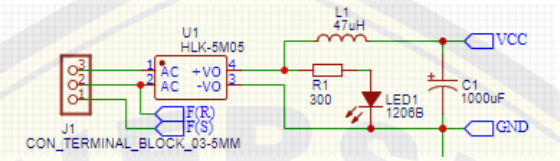
Data komunikasi serial akan dilihat dan dibaca pada program `DataESP.find('*')`, tanda bintang tersebut adalah kode bahwa data waktu telah masuk. Setelah itu data yang masuk berjumlah 3 dan diparsing sesuai variabel yang ditunjukkan pada program diatas. Kemudian sub program yang terakhir yaitu aturan program dalam perancangan kontrol berdasarkan waktu maupun daya maksimum.

c. Perancangan Skematik Sistem Minimum

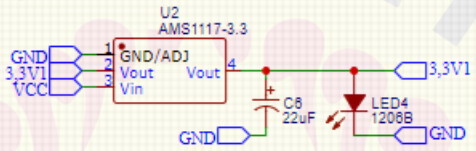
1) Rancangan *Supply* Tegangan

Supply tegangan merupakan peran utama dari bekerjanya sebuah sistem minimum. Kestabilan dari tegangan yang dikeluarkan akan mempengaruhi umur dari mikrokontroler dan komponen yang terhubung lainnya. Bahkan, kurang maupun lebihnya nilai tegangan yang diberikan juga dapat mempengaruhi kinerja

dari perangkat *input* berupa sensor maupun perangkat *output*. Pada penelitian ini terdapat dua macam *supply* tegangan yang dibutuhkan, yaitu sumber tegangan untuk subsistem 5 VDC dan 3,3 VDC. Pada penelitian ini menggunakan komponen dengan model ukuran SMD supaya tidak memakan ruang. Sehingga untuk meminimalisir terjadinya eror dalam penyelesaian tugas akhir ini maka dilakukan perancangan *supply* tegangan sebagai berikut.



Gambar 3.12 Rangkaian Skematik *Supply* Tegangan 5V



Gambar 3.13Rangkaian Skematik *Supply* Tegangan 3,3V

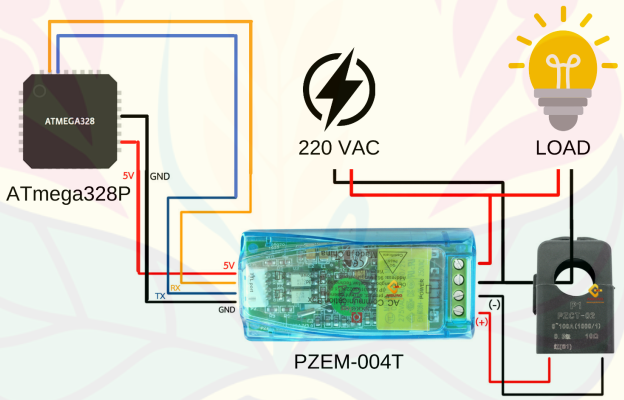
2) Rancangan Sensor PZEM-004T

Sensor yang digunakan untuk membaca tegangan, arus pada beban pertama, dan cos phi adalah sensor PZEM-004T. Rentan tegangan yang dapat diukur yaitu 80V hingga 260V. Pemilihan sensor PZEM-004T dikarenakan penggunanya yang cukup mudah dapat langsung membaca tegangan, arus, cos phi dan besaran lainnya dengan komunikasi data melalui Serial. Berikut tabel spesifikasi sensor PZEM-004T-100A:

Tabel 3.6 Spesifikasi Sensor PZEM-004T-100A

Besaran	Rentang Pengukuran	Resolusi	Akurasi Pengukuran
Tegangan	80 - 260 V	0.1 V	0.5%
Faktor Daya	0.00 - 1.00	0.01	1%
Frekuensi	45 - 65 Hz	0.1 Hz	0.5%
Arus	0 - 100 A	0.001 A	0.5%
Daya Aktif	0 23 kW	0.1 W	0.5%
Energi	0 - 9999.99 kWh	1 Wh	0.5%

Penggunaan sensor PZEM-004T yaitu dengan menghubungkan power berupa 5 VDC dan ground, kemudian untuk pin komunikasi Serial yaitu RX dan TX. Berikut gambar visualisasi untuk lebih jelasnya:



Gambar 3.14 Visualisasi Rangkaian Sensor PZEM-004T

3) Rancangan Sensor YHDC SCT-013

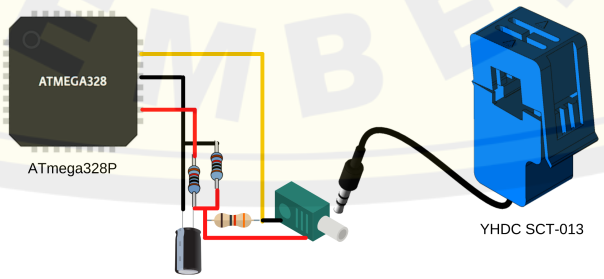
Sensor arus yang digunakan yaitu YHDC SCT-013 dengan maksimal pembacaan arus sebesar 30A. Pemilihan sensor arus YHDC-SCT-013-030 dikarenakan arus yang mengalir pada panel yang digunakan pada penelitian ini mencapai kurang lebih 25A. MCB dengan nilai tertinggi yang terdapat pada panel Lantai 3 Gedung B Fakultas Teknik Universitas Jember yaitu 30A. Sehingga pemilihan sensor arus yang tepat untuk mendapatkan pembacaan yang maksimal

yaitu YHDC SCT-013-030. *Output* tipe dari sensor yang digunakan yaitu 0V hingga 1V. Nilai tegangan inilah yang digunakan untuk perancangan program sensor arus. Spesifikasi dari sensor arus dapat dilihat pada tabel berikut:

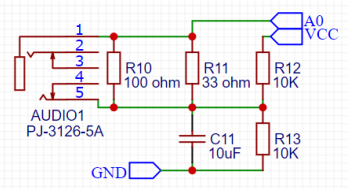
Tabel 3.7 Spesifikasi Sensor YHDC-SCT013-030

Input Current	0 - 30 A
Output Type	0 - 1 V

Dalam perancangan rangkaian untuk sensor arus terdapat beberapa hal yang perlu diperhatikan agar nilai pembacaan yang didapatkan sesuai dengan yang diharapkan. Hal yang perlu diperhatikan antara lain pemilihan resistor untuk pembagi tegangan, resistor burden yang digunakan untuk mewakili nilai *output* tegangan dari sensor, pemilihan jumlah lilitan pada sensor yang digunakan untuk perancangan pada program, serta yang diperhatikan untuk mendapatkan nilai yang mendekati dengan multimeter yaitu kode program untuk kalibrasi hasil konversi nilai ADC pada pembacaan sensor tersebut. Pada penelitian ini resistor yang digunakan untuk pembagi tegangan yaitu resistor 10K Ohm sebanyak 2 buah, kemudian resistor burden yang digunakan yaitu 100 Ohm atau 33 Ohm, dan kapasitor yang digunakan yaitu 10uF. Seluruh komponen yang digunakan pada penelitian ini menggunakan komponen dengan model SMD. Komponen yang digunakan lainnya yaitu socket audio sehingga pada penelitian ini untuk menghubungkan sensor dengan perangkat RTU cukup memasangkan kabel sensor arus yang sudah berbentuk jack audio. Berikut gambar visual beserta rangkaian skematik untuk sensor arus YHDC-SCT-013:



Gambar 3.15 Visualisasi Rangkaian Sensor YHDC SCT-013



Gambar 3.16 Rangkaian Skematik Sensor YHDC SCT-013

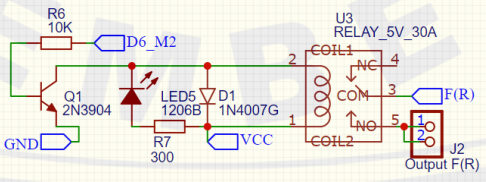
4) Rancangan Relai

Fungsi kontrol dilakukan dengan menggunakan relai, relai yang digunakan pada penelitian ini yaitu Songle 5V-30A. Pada penelitian ini tidak menggunakan modul relai yang telah siap pakai melainkan perancangan rangkaian *switching* relai digabung menjadi satu dengan sistem minimum pada RTU.

Tabel 3.8 Spesifikasi Relai Songle SLA-05VDC

Coil Voltage	5VDC
Coil Power	L 0.9W
Max Switching Current	30A
Max Switching Power	7200VA

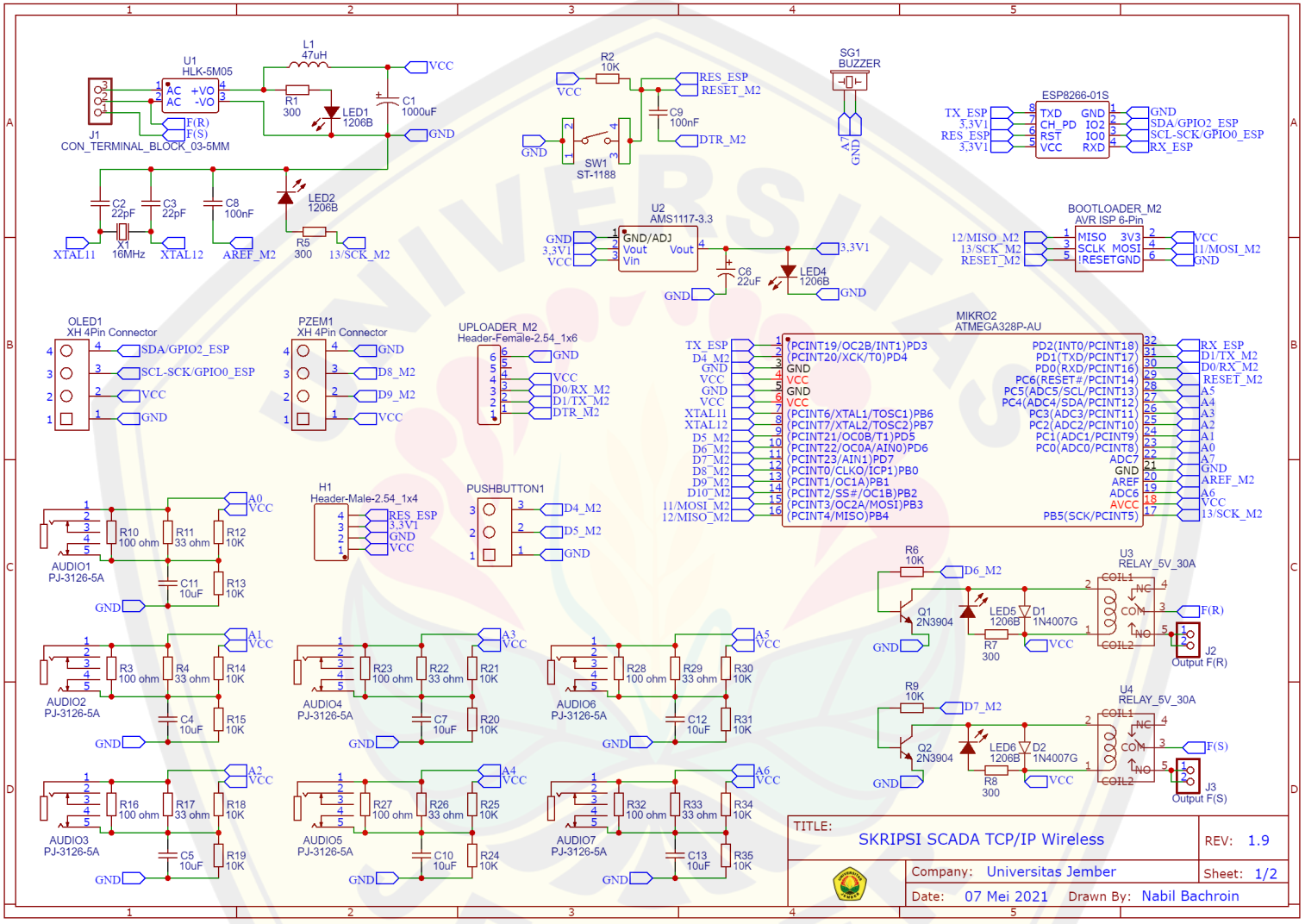
Komponen yang digunakan untuk *switching* yaitu transistor BC547. Komponen selain transistor menggunakan model ukuran SMD agar *board* PCB tidak banyak menghabiskan ruang. Pada terminal coil relai ditambahkan komponen dioda untuk pengaman arus bocor pada rangkaian *switching*. Kontak yang digunakan pada relai yaitu kontak NO (*Normally open*). Rangkaian skematik secara jelas yang digunakan seperti berikut:



Gambar 3.17 Rangkaian Skematik Relai

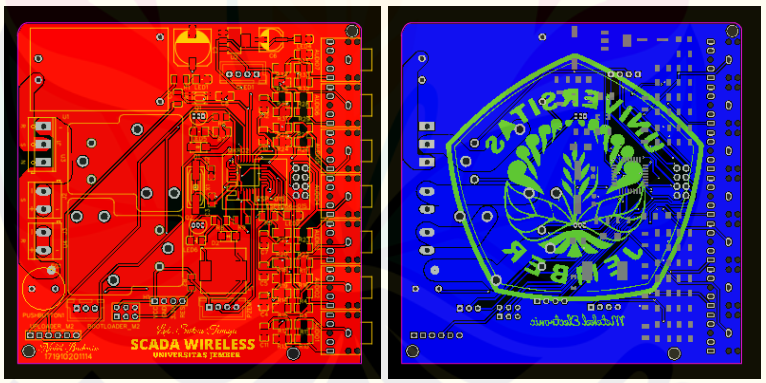
5) Rancangan Keseluruhan Sistem Minimum

Sistem minimum yang dirancang pada penelitian ini memiliki pin masukan dan keluaran yang sama seperti Arduino Nano. Pin Analog yang digunakan sebanyak 7 pin yang difungsikan untuk pin sensor arus. Didalam sistem minimum terdapat berbagai macam sub sistem meliputi perancangan rangkaian sumber tegangan, rangkaian *switching* relai, rangkaian untuk sensor arus, dan subsistem lainnya. Platform yang digunakan dalam perancangan sistem minimum pada penelitian ini yakni menggunakan EasyEDA, dimana proses perancangan sistem minimum dilakukan berbasis *website*. Terdapat dua tahapan dalam merancang sistem minimum yakni perancangan rangkaian skematik dan perancangan *layout* PCB. Berikut gambar rangkaian sistem minimum yang telah dirancang pada penelitian ini.



Gambar 3.18 Rangkaian Keseluruhan Sistem Minimum

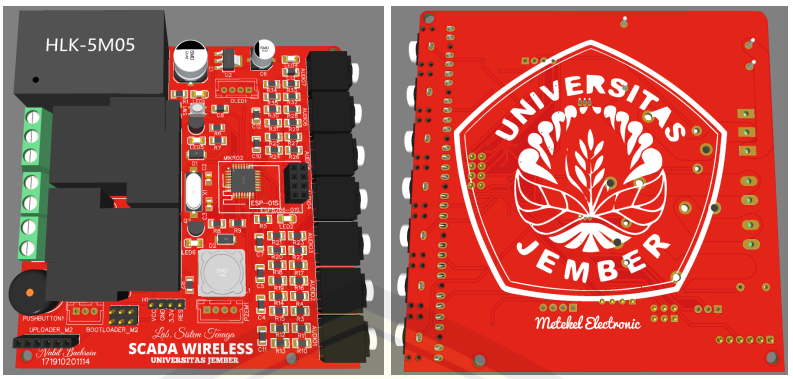
Seperti yang telah ditunjukkan pada gambar tersebut bahwa *chip* IC ATmega328P-AU dilengkapi dengan *clock* dari *crystal* dan dua buah kapasitor 22pF, komponen ini adalah syarat wajib untuk merangkai sebuah sistem minimum dengan mikrokontroler ATmega agar mikrokontroler dapat bekerja dengan baik. Pada perancangan sistem minimum tersebut dilengkapi dengan soket pin header untuk *burn bootloader*, dimana penjelasan *bootloader* telah dipaparkan pada sub bab sebelumnya. *Bootloader* yang digunakan pada sistem minimum disini yaitu *bootloader* dari arduino nano, pemilihan *bootloader* disini berdasarkan keseragaman *chip* IC yang digunakan. Pada sistem minimum tersebut juga telah dilengkapi dengan pin *uploader*, dimana pin tersebut digunakan untuk proses *upload* program menggunakan *software* Arduino IDE. Proses upload program pada penelitian ini menggunakan modul tambahan yaitu modul FT232RL-FTDI. Menginjak pada tahapan selanjutnya yaitu perancangan *layout* PCB, dalam perancangan *layout* PCB pada penelitian ini menggunakan dua layer, yakni *top* dan *bottom*. Berikut gambar hasil desain PCB pada sistem minimum yang digunakan sebagai RTU:



(a) (b)

(a) Tampak Atas; (b) Tampak Bawah

Gambar 3.19 Desain PCB



(a) (b)

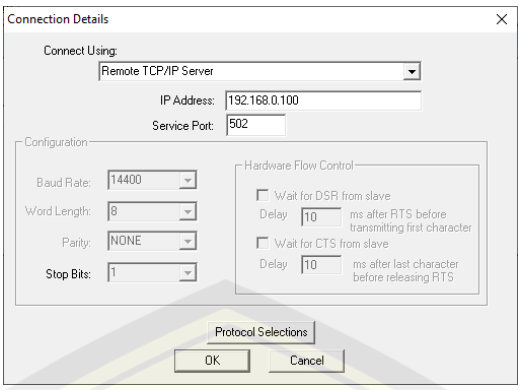
(c)

(a) Tampak Atas; (b) Tampak Bawah; (c) Tampak Detail

Gambar 3.20 Visualisasi 3D PCB

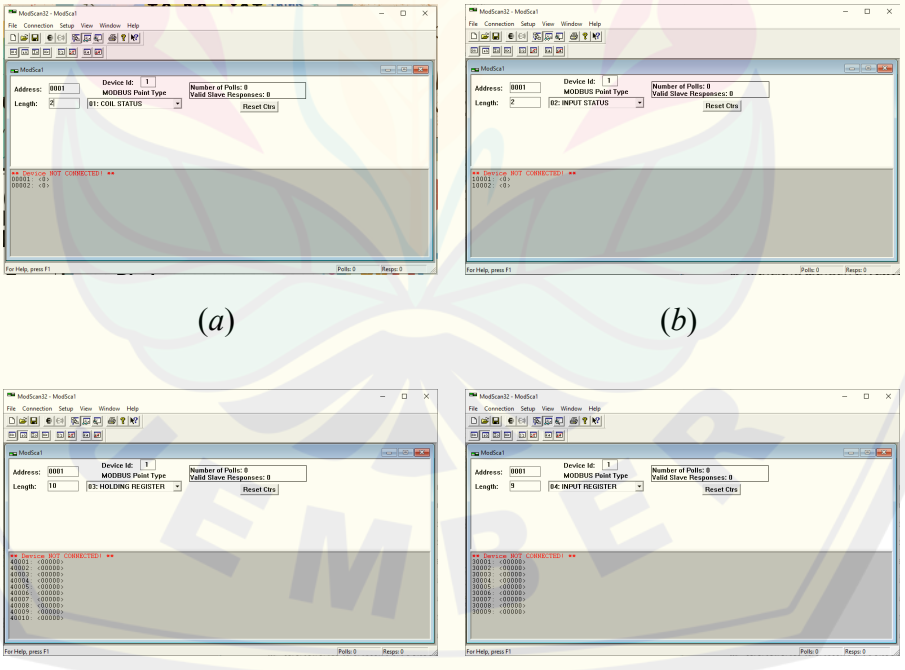
3.5.4 Rancangan MTU

Perangkat yang digunakan sebagai MTU dapat berupa *Personal Computer* (PC) maupun laptop. Dalam proses pengujian awal komunikasi menggunakan software Modscan untuk mengetahui bahwa sistem yang telah dibangun telah sesuai dengan protokol Modbus. Sistem operasi pada PC atau laptop yang akan digunakan dalam penelitian ini yaitu sistem operasi Windows hal ini dikarenakan software Modscan masih belum tersedia untuk sistem operasi Linux maupun Apple, sehingga jika menggunakan sistem operasi Linux dibutuhkan software tambahan ataupun proses tambahan untuk memasang software Modscan. Tampilan Modscan dapat dilihat seperti berikut:



Gambar 3.21 Menu pengaturan komunikasi pada ModScan

Pada Gambar 3.21 menunjukkan tampilan untuk mengatur komunikasi dengan RTU menggunakan protokol Modbus. Koneksi yang digunakan yaitu secara TCP/IP. Pengaturan yang dilakukan pada menu ini sama seperti mengatur komunikasi dengan protokol Modbus TCP/IP menggunakan kabel LAN yaitu dengan memasukkan alamat IP dari RTU dan menyamakan port yang digunakan pada RTU.



(*a*) (*b*)

(*c*) (*d*)

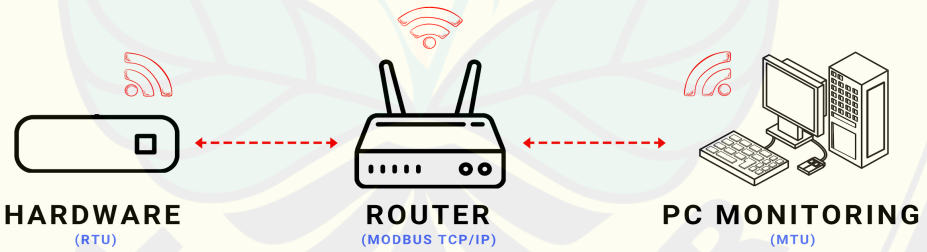
(a) Coil Status; (b) Input Status; (c) Holding Register; (d) Input Register

Gambar 3.22 Tampilan tiap kode fungsi pada ModScan

Pada Gambar 3.22 diketahui bahwa setiap kode fungsi pada ModScan akan menampilkan alamat register dan nilai dari register tersebut. Jumlah dari register tersebut disesuaikan dengan banyaknya register yang nilainya akan diisi oleh RTU. Penempatan data berupa nilai tersebut akan membentuk alamat register yang berurutan, alamat register yang digunakan juga dapat diatur oleh program pada RTU yang kemudian disesuaikan pada ModScan, pengaturan ini sesuai alamat register pertama yang diisi oleh RTU dan panjang register yang digunakan. Alamat register dan nilai pada register itulah yang akan digunakan sebagai pengujian selanjutnya, termasuk dalam perancangan pembuatan HMI yang dilakukan oleh saudari Nabila Dinda Rahmania. Jadi, hanya pada alamat register yang digunakan oleh RTU yang akan digunakan sebagai acuan utama pada perancangan HMI sehingga dengan demikian diharapkan komunikasi antara hardware dengan software dapat berjalan sesuai dengan harapan.

3.5.5 Rancangan Komunikasi

Komunikasi sistem SCADA yang akan digunakan pada penelitian ini menggunakan protokol Modbus TCP/IP berbasis *wireless* dengan media perantara yaitu perangkat router. Desain rancangan komunikasi tersebut dapat dilihat pada gambar berikut:



Gambar 3.23 Rancangan Komunikasi Sistem SCADA

Dari Gambar 3.23 tersebut dapat diketahui bahwa sinyal *Wireless Fidelity* (WiFi) yang dipancarkan oleh router akan digunakan sebagai media komunikasi. Mengingat pengertian WiFi sendiri adalah teknologi sebagai media penghantar komunikasi data yang tanpa membutuhkan kabel untuk perantaranya, dimana

teknologi ini menggunakan gelombang radio yang bekerja pada frekuensi tertentu sehingga mampu berkomunikasi maupun mentransfer data dengan cepat.

Atas dasar tersebut pula penelitian ini dilakukan dengan memanfaatkan kemampuan teknologi komunikasi jaringan nirkabel. Dengan demikian pada penelitian ini penulis memilih untuk membangun sebuah sistem komunikasi yang bersifat *local area network* (LAN) sehingga RTU dan MTU dapat berhubungan dalam area yang terbatas sesuai dengan jangkauan sinyal router yang digunakan sebagai media komunikasi data, dengan demikian sinyal WiFi yang dipancarkan oleh router pada Gambar 3.23 tidak harus memiliki koneksi internet.

Frekuensi kerja yang digunakan untuk komunikasi data pada penelitian ini menyesuaikan dari spesifikasi modul WiFi yang digunakan pada hardware yaitu mode 802,11b dengan frekuensi 2,4Ghz. Syarat agar komunikasi antara MTU dengan RTU dapat dilakukan yaitu dengan menghubungkan alamat IP dari perangkat yang akan digunakan. Adapun alamat penempatan data pada kode fungsi sesuai protokol Modbus yang harus disesuaikan antara MTU dengan RTU. Berikut kode fungsi yang digunakan pada penelitian ini:

Tabel 3.9 Kode Fungsi Protokol Modbus

Kode	Fungsi	Panjang Alamat	Keterangan
01	Coil Status	00001-00002	Sebagai penyimpan data relai
02	Input Status	10001-10002	Sebagai penyimpan data *push button*
03	Holding Register	40001-40010	Sebagai penyimpan nilai register, dapat juga difungsikan sebagai *input* maupun *output*
04	Input Register	30001-30009	Sebagai penyimpan hasil konversi ADC dari pembacaan sensor

Pada kode fungsi coil status terdapat 2 buah relai dimana 1 relai dapat dikontrol dan dimonitoring oleh MTU, sedangkan 1 relai lainnya hanya dapat dimonitoring oleh MTU. Pada kode fungsi input status terdapat 2 buah push *button* dimana kedua data ini hanya dapat dimonitoring oleh MTU karena *push*

button yang dimaksud disini terletak pada RTU. Pada kode fungsi holding register MTU dapat mengontrol atau merubah serta memonitoring nilainya. Sedangkan pada kode fungsi input register MTU hanya dapat memonitoring data hasil konversi ADC dari pembacaan sensor oleh RTU.

3.5.6 Rencana Pengujian

Adapun rencana pengujian dalam penelitian ini antara lain :

a. Pengujian Sensor PZEM-004T

Modul PZEM-004T menggunakan jalur komunikasi ganda yang digunakan untuk mengirim data kepada mikrokontroler dan menerima *request* dari mikrokontroler yang dihubungkan dengan pin komunikasi serial yaitu RX dan TX. Pengujian pada sensor PZEM-004T dibagi menjadi beberapa bagian, yaitu pengujian pembacaan tegangan, pengujian pembacaan arus, dan pengujian pembacaan nilai cos phi. Adapun proses kalibrasi sensor yang dilakukan yaitu dengan membandingkan data dari pembacaan sensor dengan nilai hasil pengukuran oleh multimeter, multimeter yang digunakan untuk membandingkan pembacaan tegangan yaitu multimeter merk DEKKO tipe CM-600AD, sedangkan multimeter yang digunakan untuk membandingkan pembacaan arus yaitu multimeter DEKKO tipe CM-600AD, dan alat yang digunakan untuk membandingkan pembacaan cos phi dari sensor yaitu Power Meter. Kemudian dari hasil perbandingan tersebut dihitung dengan bantuan Google Spreadsheets untuk mendapatkan persamaan garis lurus dengan harapan dapat memperkecil nilai persentase eror tersebut.

Setelah dikalibrasikan dengan cara dikonversikan berdasarkan perhitungan dari persamaan yang telah ditemukan maka hasil kalibrasi tersebut dituliskan pada kolom keempat tabel 3.6 yang kemudian dihitung kembali nilai persentase eror dari hasil pengukuran oleh multimeter dengan hasil kalibrasinya. Dari setiap 5 data yang telah didapatkan tersebut kemudian diketahui tingkat akurasi sensor beserta hasil kalibrasinya dengan perhitungan rata-rata eror dari setiap 5 data tersebut.

b. Pengujian Sensor YHDC SCT-013

Modul YHDC SCT-013 menggunakan jalur komunikasi tunggal yang digunakan untuk mengirim data kepada mikrokontroler yang dihubungkan dengan pin *analog to digital converter*. Untuk menghubungkan sensor arus ke mikrokontroler perlu adanya rangkaian tambahan supaya sensor arus tersebut dapat terhubung dengan aman dan error yang diakibatkan pembacaan nilai arus dapat diminimalisir, sehingga perlu dilakukan perhitungan nilai resistansi beban.

Berdasarkan penjelasan diatas maka rangkaian tambahan pada sensor arus YHDC SCT-013 tersebut diuji dengan tujuan untuk mengetahui akurasi pembacaan sensor terhadap hasil pengukuran arus pada beban dengan multimeter. Selama pengujian, pembacaan sensor akan dikalibrasi terhadap hasil pengukuran arus beban, dengan ini diharapkan pembacaan sensor arus YHDC SCT-013 mempunyai keakuratan yang tinggi. Pada salah satu bagian program sensor arus akan memproses pulsa masukan yang dihasilkan oleh sensor arus YHDC SCT-013 yaitu setelah melewati rangkaian tambahan yang telah dipaparkan sebelumnya berdasarkan pemilihan nilai resistor yang digunakan pada rangkaian tersebut. Hasil dari pengolahan input pulsa tersebut dikonversikan oleh mikrokontroler dalam satuan Ampere (A).

c. Pengujian Pengiriman Data MTU dengan RTU Secara *Wireless*

Pada pengujian ini tertuju pada pengujian komunikasi yang telah dirancang pada RTU. Pengujian ini diperlukan untuk mengetahui bahwa RTU mampu berkomunikasi dengan MTU menggunakan protokol Modbus TCP/IP secara *wireless*. Sehingga pada proses pengujian ini meliputi pengujian komunikasi dalam hal pengawasan dan pengontrolan. Sebelum melakukan pengujian pengawasan dan pengontrolan, terlebih dahulu dilakukan pengujian kemampuan berkomunikasi berdasarkan jarak perangkat RTU dengan MTU. Pengujian yang akan dilakukan dibagi menjadi dua yaitu pengujian tanpa halangan dan pengujian dengan halangan. Pengujian tanpa halangan dilakukan pada ruangan terbuka, sedangkan pengujian dengan halangan dilakukan dengan menempatkan perangkat MTU dan RTU tidak dalam satu ruangan yang sama.

Pada pengujian ini jarak yang tertulis merupakan penjumlahan jarak antara RTU dengan router dan router dengan MTU, dikarenakan komunikasi yang dilakukan oleh RTU maupun MTU selalu melewati router sebagai media komunikasi. Karena Sedangkan kekuatan sinyal yang ditulis pada kolom dua tersebut merupakan kekuatan sinyal dari router yang dibaca oleh Laptop/PC sebagai MTU. Sedangkan pembacaan kekuatan sinyal ini dilakukan dengan laptop/PC hal ini dikarenakan ESP8266-01S tidak dapat mengetahui berapa kekuatan sinyal yang diterima olehnya, sehingga pada pengujian ini router diletakkan dekat dengan RTU, sehingga dapat diasumsikan ESP8266-01S sudah mendapatkan kekuatan sinyal yang cukup untuk melakukan komunikasi. Dengan konfigurasi yang telah disebutkan diatas maka data status komunikasi MTU dengan RTU memiliki dua kemungkinan yaitu komunikasi sukses dan komunikasi eror. Setelah pengujian tersebut telah dilakukan, maka pengujian komunikasi selanjutnya yaitu pengujian pengawasan dan pengujian pengontrolan.

1) Pengujian Pengawasan

Pada pengujian pengawasan ini dilakukan dengan bantuan software modscan yang difungsikan sebagai MTU. Pengawasan yang dilakukan oleh MTU terhadap RTU berguna untuk mengetahui kesesuaian nilai yang ditampilkan pada modscan dengan nilai pada RTU. Pada pengujian ini dilakukan pengujian pengawasan secara keseluruhan pada komunikasi Modbus yang mempunyai empat kode fungsi. Pada setiap kode fungsi dilakukan pengujian sebanyak dua kali. Komunikasi dengan protokol Modbus TCP/IP secara *wireless* antara MTU dengan RTU dikatakan berhasil apabila nilai yang ditampilkan oleh ModScan telah sesuai dengan nilai yang ditampilkan oleh LCD Oled pada RTU.

2) Pengujian Pengontrolan

RTU dibangun dengan berbagai macam pengontrolan, antara lain kontrol manual melalui MTU serta kontrol otomatis berdasarkan waktu dan daya total maksimum. Pengujian pengontrolan manual ini dilakukan guna mengetahui respon kondisi pada RTU setelah dikontrol oleh MTU menggunakan Modscan. Pengujian ini juga difungsikan untuk mengetahui lama delay dari perubahan

kondisi yang terjadi antara waktu pengontrolan dari MTU dengan waktu respon RTU. Pada pengujian pengontrolan ini hanya dilakukan pada dua kode fungsi yaitu kode fungsi 01 untuk Coil Status dan kode fungsi 03 untuk Holding Register karena idealnya pada sistem SCADA yang dapat dikontrol oleh MTU hanya pada dua kode fungsi tersebut. Pada setiap kode fungsi dilakukan pengujian pengontrolan sebanyak dua kali dan pengujian ini dilakukan dengan penempatan RTU dan MTU di dalam ruangan yang sama. Pada pengujian kontrol otomatis dilakukan guna membatasi daya yang digunakan supaya sistem tidak bekerja melebihi kapasitas kerjanya, selain itu juga berupaya dalam penghematan energi. Kontrol otomatis ini dirancang pada fungsi coil status dengan alamat register 00001 dan dipengaruhi oleh 2 variabel yakni kontrol otomatis berdasarkan waktu serta kontrol otomatis berdasarkan daya total pada 8 beban. Sehingga ketika pada pukul 05.00 WIB maka beban yang terhubung dengan RTU akan otomatis menyala dan ketika pukul 22.00 WIB maka beban yang terhubung dengan RTU akan otomatis mati. Selain itu, ketika daya total yang terbaca melebihi daya maksimal yang telah ditentukan sebelumnya (6600 W) maka relai akan otomatis memutus arus sehingga beban yang terhubung akan mati.

BAB 4. HASIL DAN PEMBAHASAN

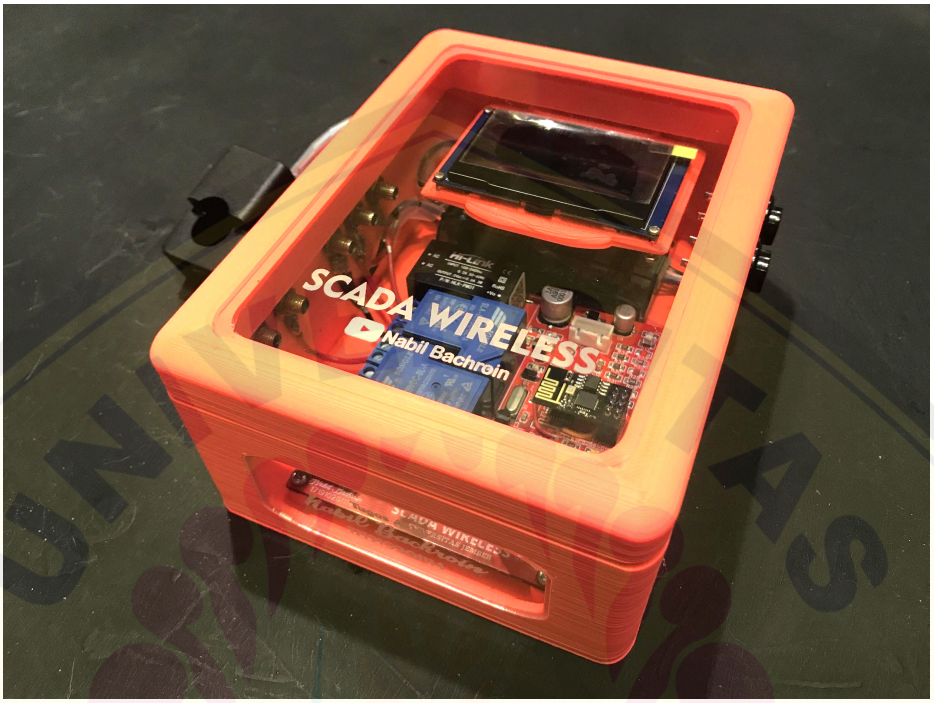
Pada bab ini akan dipaparkan perihal perancangan RTU dilengkapi dengan hasil pengujiannya berdasarkan cara kerja yang diharapkan beserta sistem komunikasi antara RTU dengan MTU. Pengujian yang dimaksud bertujuan untuk mengetahui kinerja dari sistem yang telah dibuat, dengan demikian dapat diimplementasikan pada sistem-sistem yang lainnya. Pada pembahasan bab sebelumnya telah menjelaskan metodologi mengenai perancangan sistem agar sistem yang dibangun dapat bekerja sesuai dengan yang diharapkan.

Pembahasan awal mengenai pengujian RTU meliputi pengujian sistem minimum yang telah dibuat untuk mengetahui apakah sistem minimum tersebut dapat berjalan dengan baik untuk jangka waktu yang diharapkan, pada pengujian sistem minimum ini terdiri dari pengujian keberhasilan upload pada mikrokontroler yang digunakan dan pengujian komunikasi antar mikrokontroler tersebut. Selain itu juga terdapat pengujian tiap blok subsistem RTU yang didalamnya terdapat pengujian kinerja LCD OLED dengan perubahan komunikasi yang digunakan yaitu dari SPI diubah ke komunikasi I2C, kemudian pengujian sensor tegangan dan sensor arus yang bertujuan untuk meningkatkan keakuratan dan ketelitian pembacaan sensor tersebut sehingga nilai yang ditampilkan oleh MTU kedepannya dapat menampilkan data yang lebih informatif dengan keakuratan yang sangat mendekati dengan hasil pembacaan alat ukur, pengujian tiap blok subsistem RTU selanjutnya yaitu pengujian kinerja relai berdasarkan rangkaian *switching* yang dibangun pada relai tersebut. Kemudian pada bab ini juga akan memaparkan hasil pengujian pengiriman data RTU dengan MTU secara *wireless*. Pengujian terakhir yaitu pengujian keseluruhan yang dilakukan langsung pada Gedung B Fakultas Teknik Universitas Jember.

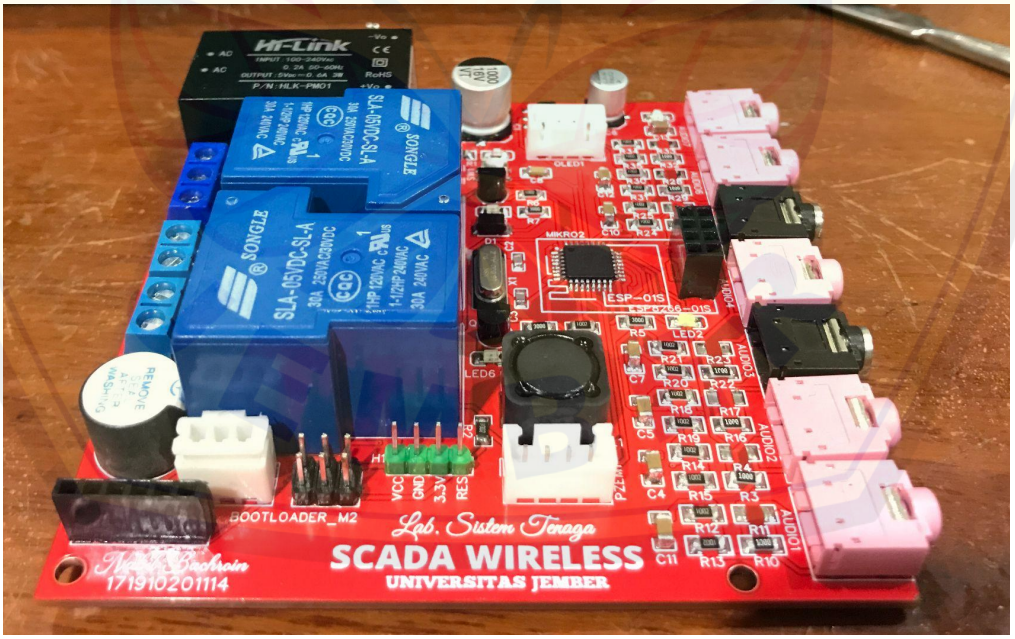
4.1 Hasil Perancangan RTU dan Pengujian RTU

RTU adalah salah satu komponen dalam sistem SCADA yang berfungsi sebagai terminal dalam menjalankan proses monitoring dan kontrol sistem

SCADA. RTU pada penelitian ini dibangun menggunakan dua buah mikrokontroler yaitu ATmega328P-AU dan ESP8266-01 yang terlihat pada gambar berikut:



Gambar 4.1 Realisasi Modul SCADA *Wireless*



Gambar 4.2 Hasil Perancangan RTU

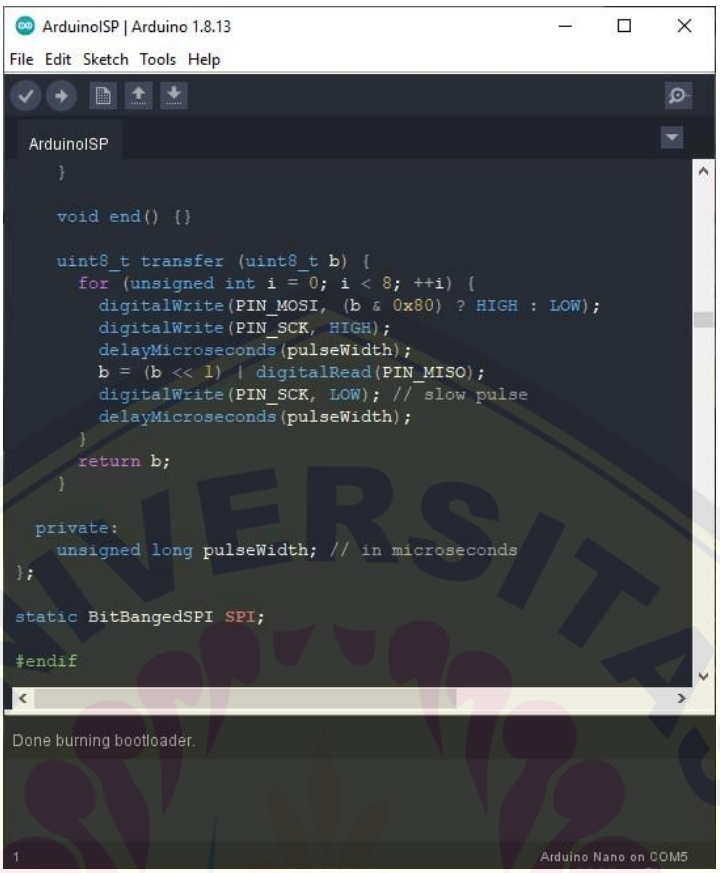
Pada RTU terdiri dari dua buah mikrokontroler yaitu ESP8266-01 dan ATmega328P-AU. Kedua mikrokontroler ini bertindak sebagai *slave* yang akan menerima dan merespon perintah dari MTU melalui komunikasi modbus TCP/IP dengan menggunakan media komunikasi untuk pengiriman data RTU dengan MTU yang berupa *wireless*.

4.1.1 Hasil Perancangan dan Pengujian Sistem Minimum

Sistem minimum yang telah dirancang perlu dilakukan pengujian pada setiap fungsi pada subsistem yang telah dibangun. Pengujian pada masing-masing fungsi dari subsistem ini dilakukan untuk mengetahui kemampuan maupun kelancaran dari kinerja yang telah dibangun pada penelitian ini.

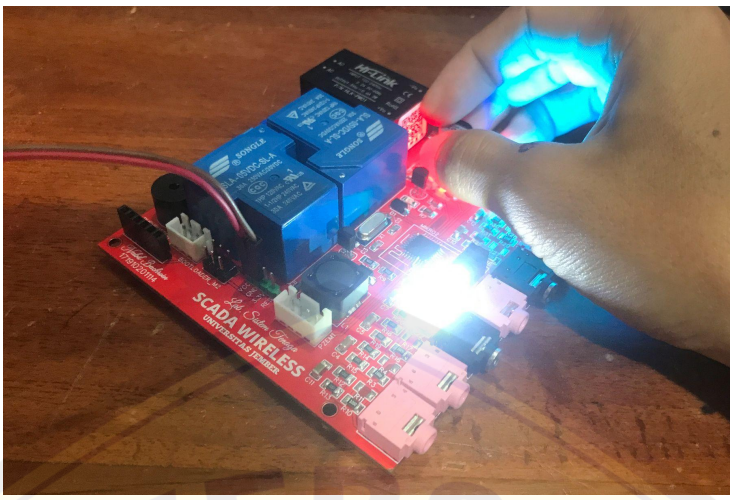
a. Pengujian *Burn Bootloader* dan *Upload*

Dalam penelitian ini *bootloader* adalah bagian paling awal untuk dapat memulai sebuah pemrograman pada sistem minimum yang dibuat mengingat pada penelitian ini tidak menggunakan sistem minimum yang sudah banyak dipasaran seperti Arduino, Teensy, OpenCM, STM32 dan sebagainya. Sehingga untuk melakukan pemrograman pada mikrokontroler harus dipastikan terlebih dahulu bahwa proses *burn bootloader* ini berhasil. Ketika proses *burn bootloader* berhasil dapat diketahui pada perangkat lunak Arduino IDE terdapat tulisan "*Done Burn Bootloader*" seperti pada gambar berikut:



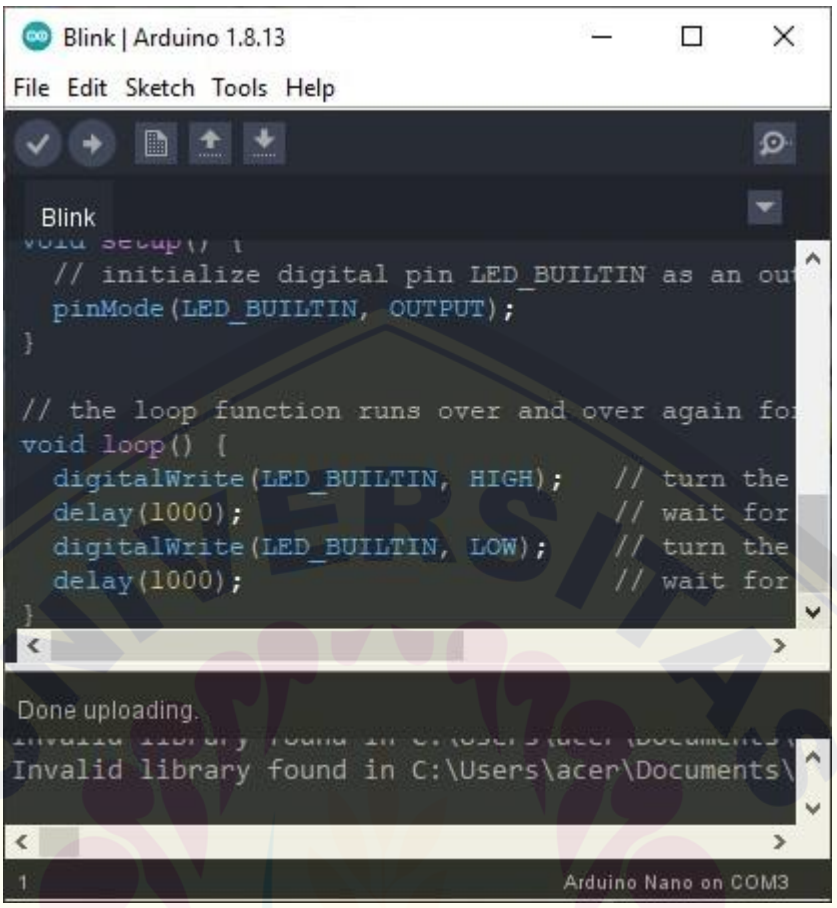
Gambar 4.3 Tampilan Ketika *Burn Bootloader* Sukses

Ketika proses *burning bootloader* telah berhasil, itu menandakan bahwa IC mikrokontroler telah terisi dengan tipe *bootloader* yang digunakan pada penelitian ini. Untuk mengetahui bahwa sistem minimum dapat bekerja dengan mikrokontroler yang telah terisi dengan *bootloader* maka pengujian yang perlu dilakukan yaitu menekan tombol reset yang telah dirancang pada sistem minimum. Keberhasilan kinerja dari sistem minimum ketika tombol reset ditekan pada pengujian ini ditandai dengan menyalanya lampu indikator berwarna putih seperti pada gambar dibawah ini:



Gambar 4.4 Kondisi LED Ketika Tombol Reset Ditekan

Setelah diketahui sistem minimum sudah dapat bekerja dengan baik menggunakan mikrokontroler yang telah terisi dengan *bootloader*, maka pengujian selanjutnya yaitu melakukan pemrograman pada perangkat lunak Arduino IDE kemudian dilakukan proses unggah program tersebut ke sistem minimum. Pemilihan *board* pada perangkat lunak disesuaikan dengan *bootloader* yang dipasang pada sistem minimum berdasarkan penyesuaian IC mikrokontroler yang digunakan. Ketika proses unggah program tersebut telah berhasil maka akan terdapat tulisan "*Done Uploading*" pada perangkat lunak Arduino IDE seperti pada gambar berikut:

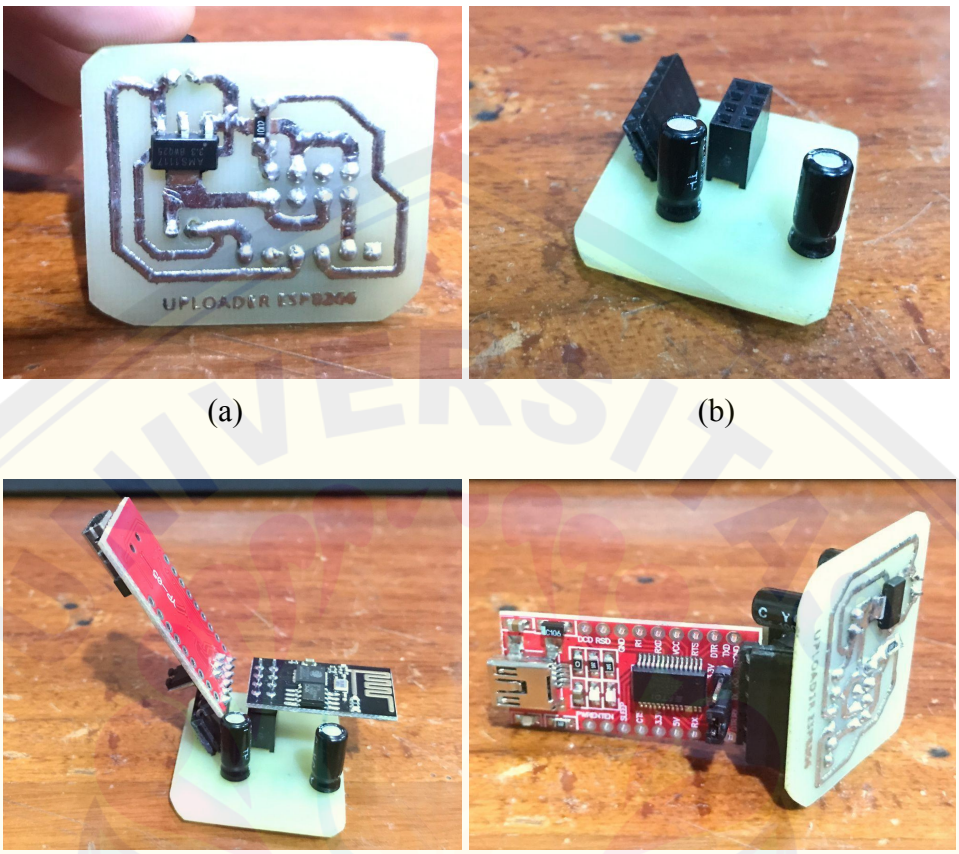


Gambar 4.5 Tampilan Ketika *Upload* ke ATmega Pada Sistem Minimum Berhasil

Setelah diketahui bahwa beberapa pengujian untuk konfigurasi penggunaan IC ATmega328P diatas sudah berhasil, maka dapat dipastikan bahwa sistem minimum yang dirancang sudah dapat digunakan sebagaimana sistem minimum yang ada dipasaran seperti yang dibuat oleh pabrik. Dengan demikian proses pengujian selanjutnya dan proses lainnya yang diperlukan dalam penelitian ini dapat dilanjutkan.

Penggunaan mikrokontroler ESP8266-01 adalah bagian paling penting agar komunikasi RTU dengan MTU dapat dilakukan secara wireless. Namun dalam pembelian ESP8266-01 diperlukan adanya rangkaian tambahan untuk mempermudah dalam melakukan pemrograman pada mikrokontroler tersebut. Dalam mengatasi permasalahan ini penulis merancang sebuah rangkaian tambahan yang kemudian telah direalisasikan dan diinformasikan pada bab ini.

Bentuk dari hasil realisasi rangkaian untuk keperluan *upload* program pada mikrokontroler ESP8266-01 dapat dilihat pada gambar berikut:



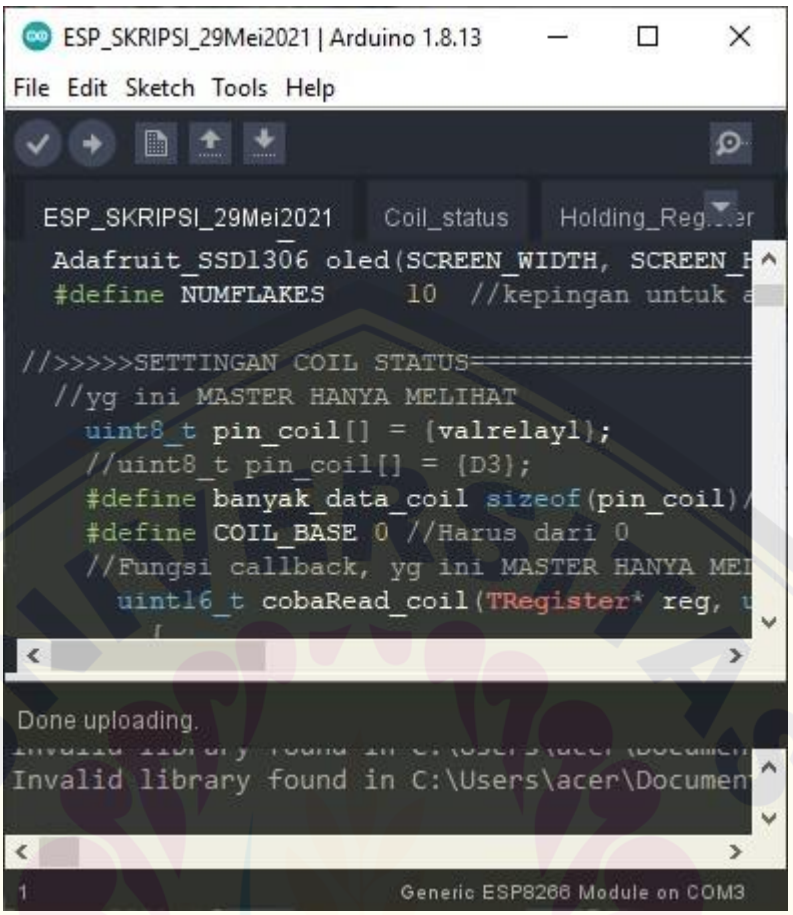
(a) (b)

(c) (d)

(a) Tampak Bawah, (b) Tampak Atas, (c) Konfigurasi Rangkaian Tampak Atas, (d) Konfigurasi Rangkaian Tampak Samping

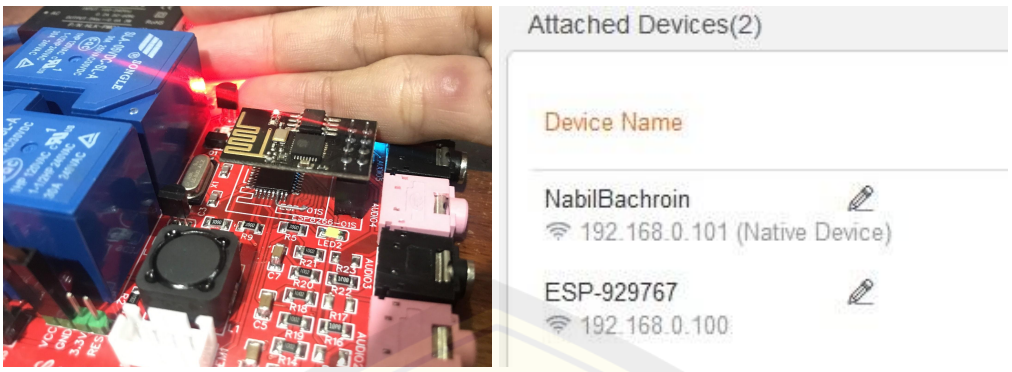
Gambar 4.6 Realisasi Rangkaian *Uploader* ESP8266-01

Pada gambar 4.6 tersebut menunjukkan untuk melakukan proses pemrograman dengan menggunakan rangkaian tersebut penggunaannya cukup memasangkan ESP8266-01 dan modul IC FT232-FTDI seperti pada gambar tersebut. Setelah mikrokontroler telah siap diisi dengan program, maka pengujian selanjutnya yaitu uji keberhasilan *upload* program pada ESP8266-01 menggunakan rangkaian tersebut. Hasil dari uji keberhasilan *upload* ke ESP8266-01 dapat dilihat pada gambar berikut:



Gambar 4.7 Hasil Uji *Upload* Pada ESP8266-01

Setelah dilakukan proses *upload* maka dapat dilihat pada perangkat lunak Arduino IDE terdapat tulisan "*Done Uploading*" yang menandakan proses *upload* menggunakan rangkaian tambahan tersebut berhasil dilakukan yang dibuktikan dengan gambar 4.7 diatas. Kemudian proses pengujian selanjutnya yaitu memastikan bahwa program yang telah dibuat pada Arduino IDE dapat diterjemahkan dengan baik dan dapat dieksekusi oleh mikrokontroler dengan bantuan rangkaian tambahan tersebut. Pada langkah ini dapat dilakukan pengujian secara singkat yaitu cukup melihat apakah led indikator merah menyala yang menandakan ESP8266-01 sudah dapat terhubung dengan jaringan yang diharapkan, sehingga pada pengujian ini cukup melakukan *upload* program yang memerintahkan ESP8266-01 agar terhubung dengan jaringan yang telah ditentukan. Hasil dari pengujian ini ditunjukkan pada gambar berikut:



(a) (b)

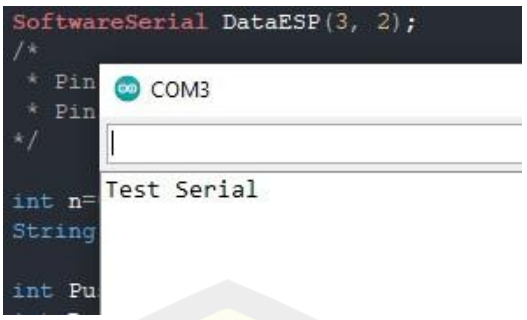
(a) Tampilan Indikator LED ESP8266-01; (b) Daftar Perangkat yang Terhubung

Gambar 4.8 Hasil Uji Koneksi ESP8266-01

Pada pengujian ini dapat diketahui bahwa lampu led indikator berwarna merah pada ESP8266-01 menyala dengan baik seperti yang terlihat pada gambar 4.8.(a) dan diperkuat dengan daftar perangkat yang telah terhubung dengan router ditunjukkan pada gambar 4.8.(b), yang menandakan perancangan rangkaian tersebut berhasil dan ESP8266-01 dapat digunakan untuk melakukan pemrograman selanjutnya dan siap untuk digunakan komunikasi antara RTU dengan MTU.

b. Pengujian Komunikasi Serial ESP8266-01 dengan ATmega328P-AU

Didalam sistem minimum terdapat dua buah mikrokontroler yang digunakan sebagai RTU. Dalam pengujian ini dimaksudkan untuk mengetahui apakah kedua mikrokontroler ini dapat berkomunikasi dengan baik untuk memproses data yang diinginkan oleh MTU. Pengujian ini dirasa perlu dilakukan, karena hal ini adalah bagian mendasar yang cukup penting untuk dipastikan keberhasilannya agar data yang dibaca oleh sensor dapat terkirim dengan tepat kepada MTU. Berikut gambar hasil uji komunikasi pada mikrokontroler yang digunakan:



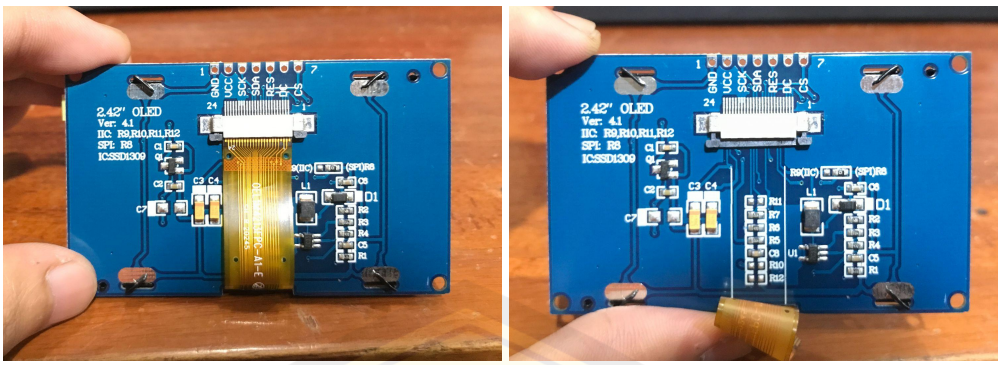
Gambar 4.9 Hasil Uji Komunikasi Mikrokontroler

Pada gambar 4.9 tersebut menunjukkan bahwa mikrokontroler telah berhasil melakukan komunikasi. Kalimat "Test Serial" merupakan perintah program yang dikirim oleh mikrokontroler lainnya. Pada gambar tersebut menunjukkan bahwa kalimat yang dikirim oleh mikrokontroler lainnya dapat ditampilkan pada *Serial Monitor* pada Arduino IDE. Sehingga kedua mikrokontroler pada RTU telah siap digunakan untuk komunikasi data dari hasil pembacaan sensor maupun data yang dikirimkan oleh MTU.

4.1.2 Pengujian Tiap Blok Subsistem RTU

a. Pengujian Perubahan Komunikasi LCD Oled

LCD adalah bagian utama dalam menginformasikan keadaan RTU ataupun data lainnya. Model komunikasi *default* dari LCD Oled yaitu menggunakan komunikasi SPI, namun pada penelitian ini dalam penggunaan LCD tersebut tidak dilakukan dengan komunikasi secara SPI dengan mikrokontroler ESP8266-01 melainkan harus menggunakan komunikasi I2C. Hal ini dikarenakan mikrokontroler ESP8266-01 tidak mendukung komunikasi SPI dikarenakan jumlah pin pada mikrokontroler tersebut sangat terbatas. Sehingga pada penelitian ini dilakukan perubahan konfigurasi komunikasi pada LCD Oled. Pengujian ini dilakukan untuk mengetahui apakah setelah dilakukan perubahan rangkaian pada LCD tersebut dapat melakukan komunikasi dengan baik untuk menampilkan tulisan maupun gambar yang diinginkan.

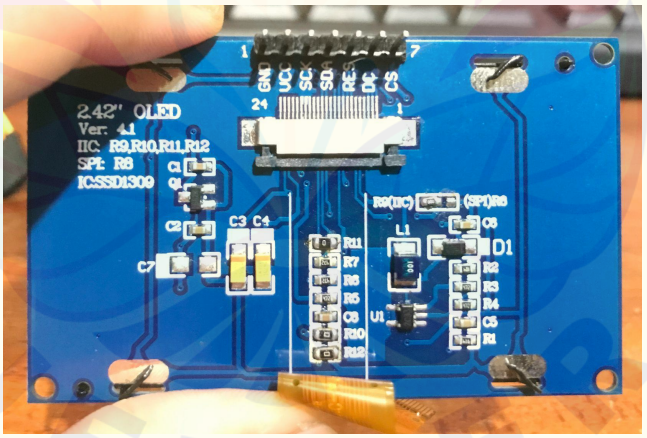


(a) (b)

(a) Tampak Luar; (b) Tampak Dalam

Gambar 4.10 Konfigurasi Komponen LCD Saat Komunikasi SPI

Gambar 4.10 diatas merupakan tampilan tampak belakang dari LCD Oled bawaan dari pabriknya yaitu ketika masih menggunakan komunikasi secara SPI. Untuk menggunakan LCD Oled tersebut dengan model komunikasi I2C maka perlu adanya perubahan penempatan resistor dan penambahan resistor pada jalur yang berada dibagian belakang dari LCD Oled tersebut. Hasil dari perubahan yang dilakukan dapat dilihat pada gambar berikut:

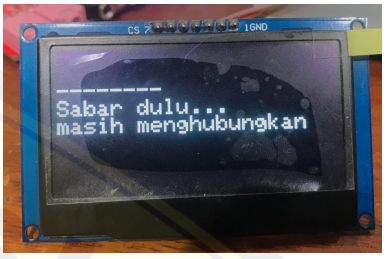


Gambar 4.11 Konfigurasi Komponen LCD Saat Komunikasi I2C

Setelah dilakukan perubahan konfigurasi pada komponen seperti yang terlihat pada gambar 4.11 maka selanjutnya yaitu melakukan pengujian kinerja dari LCD Oled. Pada pengujian ini dilakukan dengan memberikan program pada mikrokontroler ESP8266-01 untuk menampilkan tulisan dan menampilkan

gambar. Program dan hasil tampilan pada LCD Oled dapat dilihat pada tabel berikut ini:

Tabel 4.1 Hasil Pengujian Perubahan Komunikasi LCD Oled

No	Perintah	Tampilan LCD OLED
1	``` void oled_menunggu() { oled.clearDisplay(); oled.setTextColor(SSD1306_WHITE); oled.setTextSize(1); oled.setCursor(0,15); oled.println(F("--------")); oled.println(F("Sabar dulu...")); oled.println(F("masih menghubungkan")); oled.display(); delay(1500); } ``` Gambar 4.12 program untuk menampilkan tulisan	 Gambar 4.13 Tampilan Kalimat pada LCD Oled
2	``` #define tinggi_logo_unej 60 //+naik -turun #define lebar_logo_unej 61 const unsigned char logo_unej[] PROGMEM = { 0x00,0x00,0x00,0x0F,0x80,0x00,0x00,0x00, // 0x00,0x00,0x00,0x3F,0xE0,0x00,0x00,0x00, // 0x00,0x00,0x00,0xF2,0x78,0x00,0x00,0x00, // 0x00,0x00,0x03,0xDF,0x9E,0x00,0x00,0x00, // 0x00,0x00,0x0F,0x78,0xF7,0x80,0x00,0x00, // 0x00,0x00,0x1D,0xC0,0x1D,0xC0,0x00,0x00, // 0x00,0x00,0x73,0x00,0x06,0x70,0x00,0x00, // 0x00,0x00,0xEE,0x00,0x03,0xB8,0x00,0x00, // ``` Gambar 4.14 program untuk menampilkan gambar	 Gambar 4.15 Tampilan Gambar pada LCD Oled

Berdasarkan hasil pengujian tersebut dapat diketahui bahwa LCD OLED dapat berkomunikasi dengan baik dengan ESP8266-01 menggunakan model komunikasi secara I2C. Pada gambar 4.12 merupakan kode program untuk menampilkan kalimat pada LCD Oled yaitu cukup dengan memanggil dari nama inisilaisasi LCD Oled yakni "`oled.println(kalimat)`". Tanda "`F`" pada kode program merupakan perintah untuk menyimpan *string* yang ingin ditampilkan pada memori *Flash*, sehingga kalimat yang ingin ditampilkan tersebut tidak disimpan dalam memori mikrokontroler, hal ini bertujuan agar memori pada mikrokontroler tidak berkurang terlalu banyak. Pada gambar 4.14 merupakan cuplikan kode program untuk menampilkan bentuk gambar pada LCD Oled, kode program yang diberikan merupakan hasil konversi dari bentuk gambar

menjadi bitmap, kemudian kode biner maupun kode hex yang diperoleh tersebut yang digunakan pada kode program.

b. Pengujian Sensor PZEM-004T

Sensor PZEM-004T pada penelitian ini difungsikan untuk mengukur tegangan, arus pada beban yang pertama, dan cos phi pada kelistrikan Gedung B Fakultas Teknik Universitas Jember. Sehingga pada pengujian sensor PZEM-004T ini terdapat 3 macam pengujian yaitu pengujian pada masing-masing besaran tersebut.

Pengujian yang pertama yaitu pengujian pembacaan tegangan dilakukan dengan membandingkan nilai pembacaan multimeter DEKKO CM-600DR serta menghitung error persen dari perbandingan kedua nilai tersebut. Pengujian ini bertujuan untuk meningkatkan akurasi hasil pembacaan tegangan oleh sensor PZEM-004T. Pada perhitungan nilai error persen digunakan persamaan sebagai berikut:

$$E\% = \left|\frac{Ht - Hp}{Ht}\right| \times 100\% \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4.1)$$

Keterangan:

Ht = Hasil teori

Hp = Hasil pengukuran

Pada pengujian pembacaan tegangan dimulai dari tegangan 80V hingga 250V. Data hasil pengujian pembacaan tegangan pada sensor PZEM-004T dapat dilihat pada tabel berikut:

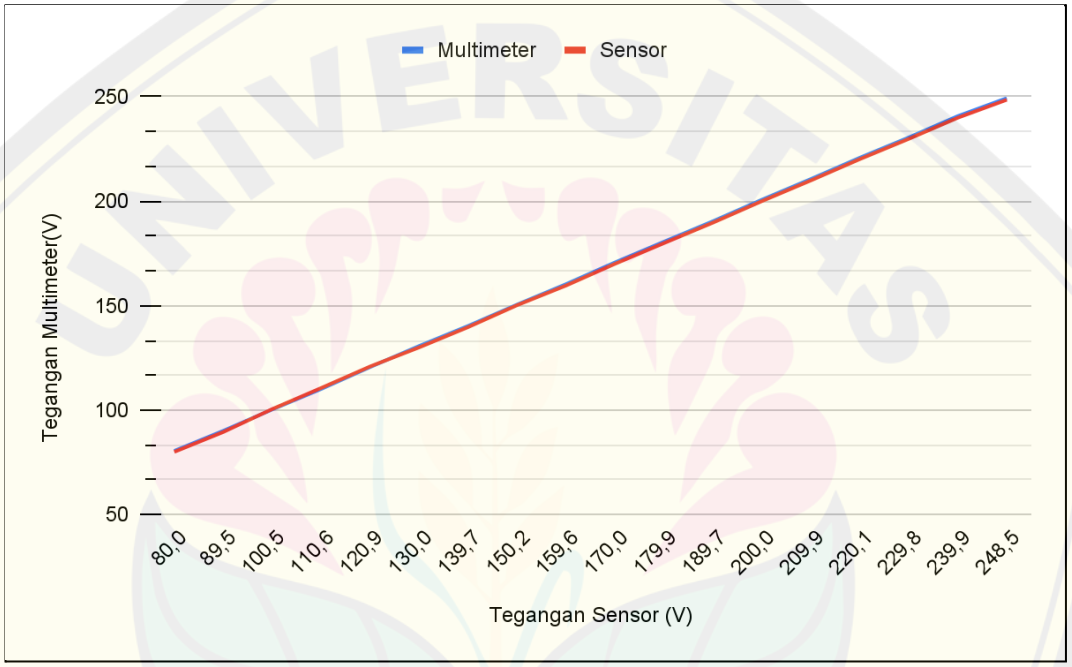
Tabel 4.2 Data Hasil Pengujian Pembacaan Tegangan Sensor PZEM-004T

Tegangan Multimeter (V)	Tegangan Sensor (V)	Error (%)
80,5	80,0	0,62
90,1	89,5	0,67
100,3	100,5	0,20
110,1	110,6	0,45
120,6	120,9	0,25
130,7	130,0	0,54
140,3	139,7	0,43
150,5	150,2	0,20
160,3	159,6	0,44
170,7	170,0	0,41
180,6	179,9	0,39
190,4	189,7	0,37
200,7	200,0	0,35
210,6	209,9	0,33
220,8	220,1	0,32
230,5	229,8	0,30
240,7	239,9	0,33
249,3	248,5	0,32
	Rata-rata error	0,38

Berdasarkan tabel 4.2 data hasil pengujian pembacaan tegangan sensor PZEM-004T didapatkan nilai perbandingan antara nilai hasil pembacaan oleh multimeter dengan nilai hasil pembacaan tegangan oleh sensor PZEM-004T yang digunakan pada RTU beserta hasil perhitungan error persen atas perbandingan

kedua nilai hasil pembacaan tersebut. Berdasarkan data hasil pengujian tersebut, diperoleh rata-rata error dalam pengujian pembacaan tegangan yaitu sebesar 0,38% dengan error tertinggi pada saat tegangan 90,1V yaitu sebesar 0,67% dan error terendah pada saat tegangan 100,3V dan 150,5V yaitu sebesar 0,20%.

Pembacaan tegangan antara sensor PZEM-004T dengan pembacaan tegangan oleh multimeter sudah sangat mendekati, hal ini dapat ditunjukkan pada grafik hasil pengujian yang telah diperoleh sebagai berikut:



Gambar 4.16 Grafik Perbandingan Pembacaan Tegangan Multimeter dan Sensor PZEM-004T

Selain pengujian keakuratan pembacaan tegangan oleh sensor PZEM-004T diatas, pada proses pengujian ini juga dilakukan pengujian kepresisian dari pembacaan sensor tersebut untuk mengetahui nilai presisi dari pembacaan sensor. Pengujian presisi ini dilakukan dengan cara menuliskan nilai sensor selama 5x dan kemudian hasilnya tersebut dihitung nilai rata-ratanya.

Tabel 4.3 Data Hasil Uji Presisi Pembacaan Tegangan

Tegangan (V)	Sensor Tegangan (V)					Rata - Rata	Standar Deviasi	Error (%)
	Uji 1	Uji 2	Uji 3	Uji 4	Uji 5			
80,5	80,3	80,0	80,0	80,0	80,0	80,1	0,134	0,55
90,1	89,5	89,5	89,5	89,5	89,6	89,5	0,045	0,64
100,3	100,6	100,5	100,5	100,5	100,6	100,5	0,055	0,24
110,1	110,6	110,6	110,6	110,6	110,7	110,6	0,045	0,47
120,6	120,9	120,9	120,9	120,8	120,9	120,9	0,045	0,23
130,7	130,0	130,0	130,3	130,3	130,0	130,1	0,164	0,44
140,3	139,5	139,7	139,7	139,7	139,8	139,7	0,110	0,44
150,5	149,8	150,2	149,7	149,8	149,8	149,9	0,195	0,43
160,3	159,6	159,7	159,7	159,6	159,6	159,6	0,055	0,41
170,7	170,0	170,0	170,2	170,0	170,0	170,0	0,089	0,39
180,6	179,9	179,9	179,9	179,9	179,9	179,9	0,000	0,39
190,4	190,1	190,0	189,6	189,7	189,7	189,8	0,217	0,30
200,7	200,0	200,0	199,9	200,0	200,1	200,0	0,071	0,35
210,6	209,8	209,8	209,8	209,9	209,8	209,8	0,045	0,37
220,8	219,9	219,9	220,0	220,1	220,1	220,0	0,100	0,36
230,5	229,8	229,8	229,7	229,7	229,7	229,7	0,055	0,33
240,7	239,9	239,9	239,9	239,9	240,0	239,9	0,045	0,32
249,3	248,6	248,5	248,6	248,5	248,5	248,5	0,055	0,30

Pengujian yang kedua pada sensor PZEM-004T yaitu pengujian pembacaan arus dilakukan dengan membandingkan nilai pembacaan multimeter DEKKO CM-600DR serta menghitung error persen dari perbandingan kedua nilai

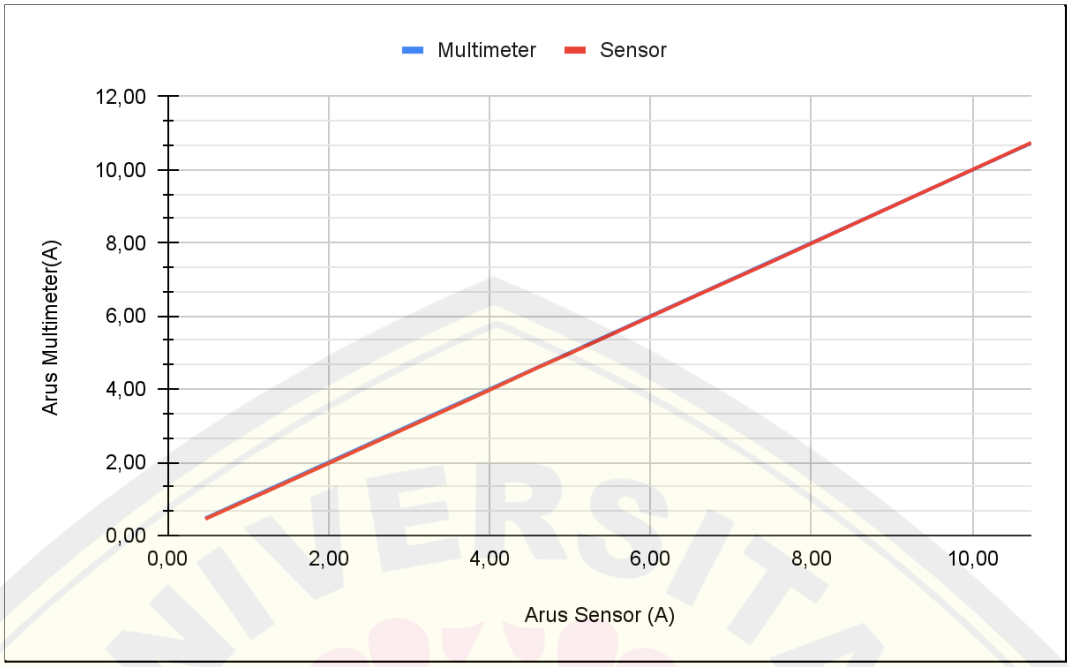
tersebut. Pengujian ini bertujuan untuk meningkatkan akurasi hasil pembacaan sensor yang digunakan.

Tabel 4.4 Data Hasil Pengujian Pembacaan Arus Sensor PZEM-004T

Arus Multimeter (A)	**Arus Sensor (A)**	**Error (%)**
0,45	0,47	4,44
1,18	1,21	2,54
2,42	2,45	1,24
3,75	3,78	0,80
4,59	4,60	0,22
5,00	5,02	0,40
6,58	6,59	0,15
7,40	7,42	0,27
8,23	8,24	0,12
10,74	10,73	0,09
	Rata-rata error	1,03

Berdasarkan tabel 4.4 data hasil pengujian pembacaan arus sensor PZEM-004T didapatkan nilai perbandingan antara nilai hasil pembacaan oleh multimeter dengan nilai hasil pembacaan sensor arus yang digunakan pada RTU beserta hasil perhitungan error persen atas perbandingan kedua nilai hasil pembacaan tersebut. Berdasarkan data hasil pengujian tersebut, diperoleh rata-rata error dalam pengujian sensor arus PZEM-004T yaitu sebesar 1,03% dengan error tertinggi pada saat arus 0,45A yaitu sebesar 4,44% dan error terendah pada saat arus 10,74A yaitu sebesar 0,09%.

Hasil pembacaan arus antara sensor PZEM-004T dengan pembacaan arus oleh multimeter sudah sangat mendekati, hal ini dapat ditunjukkan pada grafik hasil pengujian yang telah diperoleh sebagai berikut:



Gambar 4.17 Grafik Perbandingan Pembacaan Arus Multimeter dan Sensor PZEM-004T

Selain pengujian keakuratan pembacaan arus sensor PZEM-004T diatas, pada proses pengujian ini juga dilakukan pengujian kepresisian dari pembacaan sensor tersebut untuk mengetahui nilai presisi dari pembacaan sensor. Pengujian presisi ini dilakukan dengan cara menuliskan nilai sensor selama 5x dan kemudian hasilnya tersebut dihitung nilai rata-ratanya.

Tabel 4.5 Data Hasil Uji Presisi Pembacaan Arus Sensor PZEM-004T

Arus (A)										Alat ukur		Sensor		Error (%)
Uji 1		Uji 2		Uji 3		Uji 4		Uji 5						
Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Rata-Rata	Standar Deviasi	Rata-Rata	Standar Deviasi	
0,42	0,39	0,44	0,47	0,43	0,51	0,47	0,49	0,45	0,48	0,44	0,019	0,47	0,046	**5,88**
1,18	1,21	1,17	1,21	1,18	1,21	1,16	1,21	1,17	1,21	1,17	0,008	1,21	0,000	**3,24**
2,43	2,46	2,41	2,45	2,42	2,45	2,41	2,45	2,43	2,45	2,42	0,010	2,45	0,004	**1,32**
3,76	3,79	3,80	3,79	3,78	3,78	3,77	3,77	3,75	3,78	3,77	0,019	3,78	0,008	**0,27**
4,58	4,60	4,59	4,60	4,58	4,59	4,60	4,60	4,58	4,60	4,59	0,009	4,60	0,004	**0,26**
4,99	5,02	5,00	5,02	4,99	5,02	5,00	5,02	4,99	5,01	4,99	0,005	5,02	0,004	**0,48**
6,57	6,59	6,58	6,59	6,57	6,59	6,59	6,59	6,58	6,59	6,58	0,008	6,59	0,000	**0,18**
7,39	7,42	7,41	7,42	7,42	7,41	7,41	7,42	7,40	7,41	7,41	0,011	7,42	0,005	**0,14**
8,25	8,24	8,23	8,24	8,24	8,24	8,23	8,24	8,24	8,24	8,24	0,008	8,24	0,000	**0,02**
10,76	10,74	10,75	10,74	10,73	10,73	10,74	10,73	10,74	10,73	10,74	0,011	10,73	0,005	**0,09**

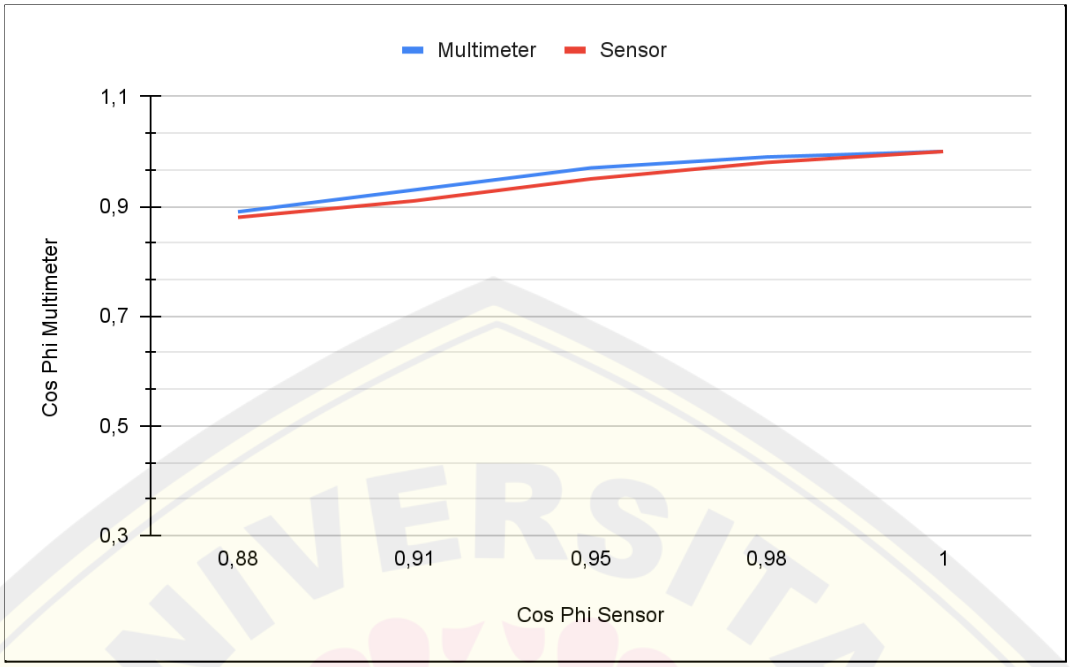
Pengujian yang ketiga pada sensor PZEM-0004T yaitu pengujian pembacaan cos phi dilakukan dengan membandingkan nilai pembacaan multimeter DEKKO CM-600DR serta menghitung error persen dari perbandingan kedua nilai tersebut. Pengujian ini bertujuan untuk meningkatkan akurasi hasil pembacaan cos phi oleh sensor PZEM-004T. Pengujian ini dilakukan menggunakan beban yang bervariatif, seperti beban lampu led, lampu pijar, kipas angin, dan beban-beban lainnya hingga nilai yang terukur oleh alat ukur ditunjukkan seperti tabel berikut ini.

Tabel 4.6 Data Hasil Pengujian Pembacaan Cos Phi Sensor PZEM-004T

Cos Phi Alat Ukur	Cos Phi Sensor	Error (%)
0,89	0,88	1,12
0,93	0,91	2,15
0,97	0,95	2,06
0,99	0,98	1,01
1	1	0,00
	Rata-rata error	1,27

Berdasarkan tabel 4.6 data hasil pengujian pembacaan cos phi sensor PZEM-004T didapatkan nilai perbandingan antara nilai hasil pembacaan oleh alat ukur dengan nilai hasil pembacaan cos phi oleh sensor PZEM-004T yang digunakan pada RTU beserta hasil perhitungan error persen atas perbandingan kedua nilai hasil pembacaan tersebut. Berdasarkan data hasil pengujian tersebut, diperoleh rata-rata error dalam pengujian pembacaan cos phi yaitu sebesar 1,27% dengan error tertinggi pada saat cos phi 0,93 yaitu sebesar 2,15% dan error terendah pada saat cos phi 1 yaitu sebesar 0%.

Hasil pembacaan cos phi antara sensor PZEM-004T dengan pembacaan cos phi oleh alat ukur sudah mendekati, hal ini dapat ditunjukkan pada grafik hasil pengujian yang telah diperoleh sebagai berikut:



Gambar 4.18 Grafik Perbandingan Pembacaan Cos Phi Power Meter dan Sensor PZEM-004T

Selain pengujian keakuratan pembacaan cos phi oleh sensor PZEM-004T diatas, pada proses pengujian ini juga dilakukan pengujian kepresisian dari pembacaan sensor tersebut untuk mengetahui nilai presisi dari pembacaan sensor. Pengujian presisi ini dilakukan dengan cara menuliskan nilai sensor selama 5x dan kemudian hasilnya tersebut dihitung nilai rata-ratanya.

Tabel 4.7 Data Hasil Uji Presisi Pembacaan Cos Phi

Cos Phi										Alat ukur		Sensor		Error (%)
Uji 1		Uji 2		Uji 3		Uji 4		Uji 5						
Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Rata-Rata	Standar Deviasi	Rata-Rata	Standar Deviasi	
0,88	0,88	0,89	0,88	0,89	0,88	0,87	0,89	0,86	0,89	0,88	0,013	0,88	0,005	**0,68**
0,94	0,9	0,93	0,91	0,93	0,91	0,93	0,91	0,92	0,92	0,93	0,007	0,91	0,007	**2,15**
0,96	0,94	0,97	0,95	0,97	0,95	0,98	0,95	0,97	0,95	0,97	0,007	0,95	0,004	**2,27**
0,99	0,98	0,99	0,98	0,99	0,98	0,99	0,98	0,99	0,98	0,99	0,000	0,98	0,000	**1,01**
1	0,99	1	1	1	1	1	0,99	1	1	1	0,000	1	0,005	**0,40**

c. Pengujian Sensor YHDC SCT-013

Sensor YHDC SCT-013 merupakan sensor arus yang digunakan pada hardware RTU, tipe yang digunakan pada RTU ini adalah sensor arus berbentuk clamp dengan model masukan arus hingga 30A dan model keluaran berupa tegangan dari 0V hingga 1V. Sensor ini digunakan untuk mengukur arus yang mengalir pada kelistrikan lantai 3 Gedung B Fakultas Teknik Universitas Jember. Pengujian ini dilakukan dengan membandingkan nilai pengukuran sensor dengan nilai hasil pengukuran oleh multimeter DEKKO CM-600DR serta menghitung error persen dari perbandingan kedua nilai tersebut. Pengujian ini bertujuan untuk meningkatkan akurasi hasil pembacaan sensor yang digunakan.

Tabel 4.8 Data Hasil Pengujian Sensor Arus 1

Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,51	**2,00**
1,33	1,33	**0,00**
2,54	2,55	**0,39**
3,73	3,72	**0,27**
4,07	4,11	**0,98**
6,56	6,56	**0,00**
7,38	7,36	**0,27**
8,2	8,22	**0,24**
9,43	9,42	**0,11**
10,99	10,99	**0,00**
Rata-rata error		**0,43**

Tabel 4.9 Data Hasil Pengujian Sensor Arus 2

Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,52	**4,00**
1,33	1,33	**0,00**
2,54	2,55	**0,39**
3,73	3,71	**0,54**
4,07	4,1	**0,74**
6,56	6,57	**0,15**
7,38	7,38	**0,00**
8,2	8,2	**0,00**
9,43	9,42	**0,11**
10,99	10,98	**0,09**
Rata-rata error		**0,60**

Tabel 4.10 Data Hasil Pengujian Sensor Arus 3

Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,51	**2,00**
1,33	1,35	**1,50**
2,54	2,53	**0,39**
3,73	3,71	**0,54**
4,07	4,11	**0,98**
6,56	6,55	**0,15**
7,38	7,37	**0,14**
8,2	8,19	**0,12**
9,43	9,43	**0,00**
10,99	11	**0,09**
Rata-rata error		**0,59**

Tabel 4.11 Data Hasil Pengujian Sensor Arus 4

Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,51	**2,00**
1,33	1,32	**0,75**
2,54	2,53	**0,39**
3,73	3,72	**0,27**
4,07	4,08	**0,25**
6,56	6,54	**0,30**
7,38	7,33	**0,68**
8,2	8,2	**0,00**
9,43	9,31	**1,27**
10,99	10,94	**0,45**
Rata-rata error		**0,64**

Tabel 4.12 Data Hasil Pengujian Sensor Arus 5

Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,52	**4,00**
1,33	1,33	**0,00**
2,54	2,53	**0,39**
3,73	3,71	**0,54**
4,07	4,08	**0,25**
6,56	6,56	**0,00**
7,38	7,39	**0,14**
8,2	8,21	**0,12**
9,43	9,42	**0,11**
10,99	11,02	**0,27**
Rata-rata error		**0,58**

Tabel 4.13 Data Hasil Pengujian Sensor Arus 6

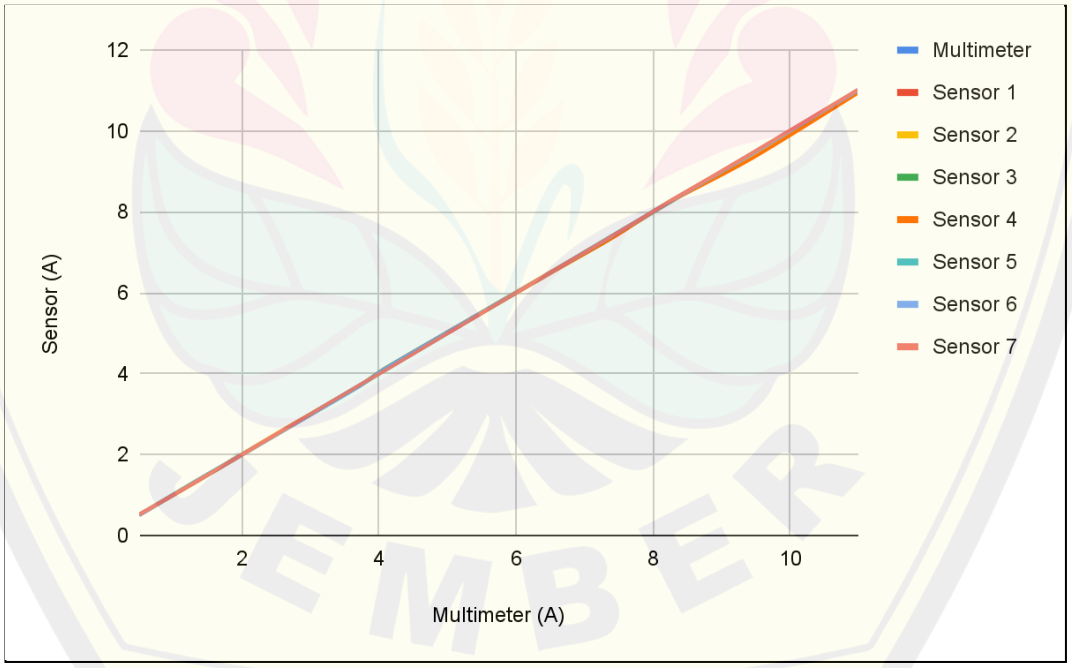
Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,51	**2,00**
1,33	1,34	**0,75**
2,54	2,53	**0,39**
3,73	3,71	**0,54**
4,07	4,1	**0,74**
6,56	6,56	**0,00**
7,38	7,38	**0,00**
8,2	8,2	**0,00**
9,43	9,43	**0,00**
10,99	11	**0,09**
Rata-rata error		**0,45**

Tabel 4.14 Data Hasil Pengujian Sensor Arus 7

Arus Multimeter (A)	Arus Sensor (A)	Error
0,5	0,52	**4,00**
1,33	1,33	**0,00**
2,54	2,54	**0,00**
3,73	3,73	**0,00**
4,07	4,06	**0,25**
6,56	6,56	**0,00**
7,38	7,38	**0,00**
8,2	8,21	**0,12**
9,43	9,44	**0,11**
10,99	11,02	**0,27**
Rata-rata error		**0,47**

Berdasarkan data hasil pengujian sensor arus YHDC SCT-013 pada tabel 4.8 hingga tabel 4.14 didapatkan nilai perbandingan antara nilai hasil pembacaan oleh multimeter dengan nilai hasil pembacaan sensor arus yang digunakan pada RTU beserta hasil perhitungan error persen atas perbandingan kedua nilai hasil pembacaan tersebut. Berdasarkan data hasil pengujian tersebut, diperoleh rata-rata eror terendah dalam pengujian sensor arus YHDC-SCT-013 pertama hingga ketujuh yaitu sebesar 0,43% yang terdapat pada hasil pengujian sensor pertama, dengan rata-rata error tertinggi dari setiap sensor yaitu sebesar 0,64% yang terdapat pada hasil pengujian sensor urutan ke-empat dan rata-rata eror dari seluruh pembacaan sensor yaitu 0,54%.

Pembacaan arus antara sensor YHDC SCT-013 yang pertama hingga sensor yang ketujuh dengan pembacaan arus oleh multimeter sudah sangat mendekati, hal ini dapat ditunjukkan pada grafik hasil pengujian yang telah diperoleh sebagai berikut:



Gambar 4.19 Grafik Perbandingan Pembacaan Arus Multimeter dan Sensor YHDC SCT-013

Selain pengujian keakuratan sensor YHDC SCT-013 diatas, pada proses pengujian ini juga dilakukan pengujian kepresisian dari pembacaan sensor tersebut untuk mengetahui nilai presisi dari pembacaan sensor. Pengujian presisi ini dilakukan dengan cara menuliskan nilai sensor selama 5x dan kemudian hasilnya tersebut dihitung nilai rata-ratanya.

Tabel 4.15 Data Hasil Uji Presisi Sensor YHDC SCT-013

Arus (A)										Alat ukur		Sensor		Error (%)
Uji 1		Uji 2		Uji 3		Uji 4		Uji 5						
Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Alat ukur	Sensor	Rata-Rata	Standar Deviasi	Rata-Rata	Standar Deviasi	
0,48	0,51	0,50	0,57	0,49	0,50	0,51	0,52	0,52	0,51	0,50	0,016	0,52	0,028	**4,40**
1,33	1,36	1,31	1,35	1,32	1,36	1,35	1,32	1,32	1,33	1,33	0,015	1,34	0,018	**1,36**
2,54	2,53	2,55	2,56	2,56	2,56	2,53	2,56	2,51	2,55	2,54	0,019	2,55	0,013	**0,55**
3,71	3,74	3,72	3,72	3,73	3,72	3,72	3,70	3,72	3,73	3,72	0,007	3,72	0,015	**0,05**
4,06	4,06	4,07	4,08	4,06	4,10	4,08	4,11	4,05	4,10	4,06	0,011	4,09	0,020	**0,64**
6,57	6,54	6,58	6,55	6,56	6,56	6,55	6,59	6,57	6,54	6,57	0,011	6,56	0,021	**0,15**
7,38	7,36	7,40	7,42	7,39	7,39	7,37	7,42	7,38	7,35	7,38	0,011	7,39	0,033	**0,05**
8,20	8,24	8,19	8,22	8,18	8,27	8,20	8,26	8,21	8,26	8,20	0,011	8,25	0,020	**0,66**
9,43	9,46	9,46	9,46	9,44	9,41	9,42	9,40	9,45	9,46	9,44	0,016	9,44	0,030	**0,02**
10,97	10,99	10,99	10,97	11,01	11,02	10,97	11,02	10,99	11,05	10,99	0,017	11,01	0,031	**0,22**

d. Pengujian Kinerja Relai

Pada perancangan pengontrolan, RTU dilengkapi dengan 2 buah relai dimana sebuah relai tidak dapat dikontrol oleh MTU dan sebuah relai lainnya dapat dikontrol oleh MTU. Pengontrolan yang disediakan pada RTU hanya diwakili dengan 2 relai, hal ini dikarenakan keterbatasan biaya. Sebab jika semua beban dirancang agar dapat dikontrol maka membutuhkan banyak relai sehingga biaya yang dikeluarkan maka akan sangat banyak. Tujuan pada pengujian ini yaitu untuk mengetahui kinerja dari rangkaian modul relai, mengingat bahwa pada penelitian ini digunakan relai 30A dengan tegangan kerja pada coil yaitu sebesar 5V. Realisasi rangkaian modul relai sudah sesuai dengan desain awal dan dapat berfungsi dengan baik. Modul Relai 5V sudah dapat menerima input trigger dengan baik yang dilakukan dengan switching pada transistor BC547 dengan memutus atau menghubungkan pada jalur bagian *ground*.

Tabel 4.16 Pengukuran Tegangan Pada Relai

No.	Keterangan	Tegangan Coil saat Relai Non-Aktif	Tegangan Coil saat Relai Aktif
1.	Relai 1	0V	4,51V
2.	Relai 2	0V	4,51V

Berdasarkan data pada tabel 4.16 Pengukuran tegangan pada relai dapat diketahui bahwa relai sudah dapat bekerja dengan baik. Tegangan yang terukur pada coil relai ketika relai diperintahkan untuk non-aktif yaitu 0V, hal ini dapat diketahui bahwa relai dapat dipastikan bahwa kondisinya benar-benar non-aktif. Ketika relai diperintahkan untuk aktif, tegangan yang terukur dengan multimeter yaitu 4,51V dan dengan demikian relai diharapkan dapat bekerja lebih tahan lama karena sudah bekerja dengan tegangan yang diharapkan.

4.2 Pengujian Keandalan Pengiriman Data Secara *Wireless*

Salah satu faktor lain yang mempengaruhi keberhasilan dari komunikasi secara *wireless* pada penelitian ini yaitu jarak antara RTU dan MTU dengan router. Sehingga pada penelitian ini dilakukan pengujian keandalan pengiriman

data secara *wireless*. Pengujian ini dilakukan di Laboratorium Fakultas Teknik Universitas Jember yang bertempat di Patrang. Pada pengujian ini dilakukan dua kali pengambilan data yakni ketika kondisi antara MTU dan RTU tidak disertai dengan halangan dan ketika disertai halangan berupa tembok. Saat pengujian tanpa halangan, RTU dengan MTU hanya terhalangi oleh adanya pohon-pohon yang berada di Laboratorium Patrang dan ketinggian dari penempatan router. Sedangkan ketika pengujian yang disertai dengan halangan, RTU dengan MTU terhalangi oleh tembok yakni RTU ditempatkan didalam ruangan sedangkan MTU berada diluar ruangan. Pada pengujian ini untuk penempatan router berada ditengah diantara RTU dengan MTU dan ketinggian router pada saat pengujian yaitu 65cm dari tanah.

Tabel 4.17 Pengujian Komunikasi Tanpa Halangan

Jarak (meter)	**Kekuatan Sinyal WiFi (dBm)**	**Status Komunikasi**
10	-48	Lancar
20	-60	Lancar
30	-65	Lancar
40	-68	Lancar
50	-70	Lancar
60	-73	Lancar
70	-78	Lancar
80	-86	Lancar
90	-87	Lancar
100	-88	Lancar

Tabel 4.18 Pengujian Komunikasi dengan Halangan

Jarak (meter)	Kekuatan Sinyal WiFi (dBm)	Status Komunikasi
10	-54	Lancar
20	-62	Lancar
30	-68	Lancar
40	-71	Lancar
50	-78	Lancar
60	-81	Lancar
70	-86	Lancar
80	-90	Lancar
90	-94	Lancar
100	-96	Kurang Lancar

Pada tabel 4.17 menunjukkan bahwa hasil pengujian komunikasi ketika tanpa halangan terbilang lancar. Dimulai jarak perangkat dari 10 meter hingga 100 meter, seluruh proses komunikasi data dapat dilakukan dengan baik. Pada tabel 4.18 menunjukkan bahwa hasil pengujian komunikasi ketika tanpa halangan terbilang lancar terkecuali ketika jarak perangkat 100 meter. Pada kondisi dengan halangan dan jarak 100 meter, komunikasi data yang dilakukan sebanyak 5 kali hanya berhasil 3 kali, sehingga pada tabel tersebut tertulis kurang lancar. Berdasarkan hasil pengujian diatas, komunikasi data antara RTU dengan MTU terbilang cukup handal dengan keberhasilannya saat melakukan pengujian komunikasi seperti yang tertulis pada kedua tabel tersebut.

4.2.1 Pengujian Pengawasan

Pada fungsi pengawasan ini merupakan proses *monitoring* status dari komponen yang terhubung dengan RTU. Pengujian fungsi ini dilakukan pada seluruh kode fungsi yang terdapat pada sistem SCADA. Pengujian ini juga dapat mengetahui bahwa komunikasi dengan protokol Modbus TCP/IP secara *wireless* antara MTU dengan RTU telah berhasil ketika nilai yang ditampilkan oleh ModScan telah sesuai dengan nilai yang telah ditampilkan oleh LCD Oled pada RTU. Pengujian ini dilakukan dengan mencatat selisih waktu pada saat MTU telah menampilkan nilai data yang diberikan oleh RTU.

Tabel 4.19 Hasil Pengujian Pengawasan

Kode Fungsi	**Komponen**	**Pengujian 1**			**Pengujian 2**		
		RTU	**MTU**	**Delay (detik)**	**RTU**	**MTU**	**Delay (detik)**
01 Coil Status	Relai	1	1	0,83	0	0	0,52
02 Input Status	Tombol Tekan	1	1	2,23	0	0	1,5
03 Holding Register	Nilai register	61680	61680	0,31	799	799	0,65
04 Input Register	Sensor Arus	0,12	0,12	4,45	1,19	1,19	4,06
	Sensor Tegangan	0	0	3,54	214,2	214,2	5,08
	Daya	1,30	1,30	3,45	105,4	105,4	4,06
Rata-rata delay (detik)		**2,47**			**2,65**		

Berdasarkan tabel 4.19 data hasil pengujian pengawasan rata-rata dari respon pada pengujian pertama yaitu 2,47 detik dan 2,65 detik untuk rata-rata pengujian kedua. Rata-rata delay dari 2 pengujian diatas adalah 2,56 detik.

Respon terlambat pada saat pengujian pertama yaitu pada nilai sensor arus yang berada pada kode fungsi *input register* sebesar 4,45 detik. Sedangkan respon terlambat pada saat pengujian kedua yaitu pada nilai sensor tegangan yang berada pada kode fungsi *input register* yaitu sebesar 5,08 detik.

Pada pengujian pengawasan juga terdapat pengujian untuk *monitoring* pada pembacaan daya. Pengujian selanjutnya yaitu pengujian *monitoring* nilai daya yang dilakukan dengan membandingkan nilai hasil perhitungan pembacaan sensor dengan nilai pembacaan alat ukur. Pada perhitungan untuk mencari nilai daya pada hasil pembacaan sensor diperoleh dengan rumus:

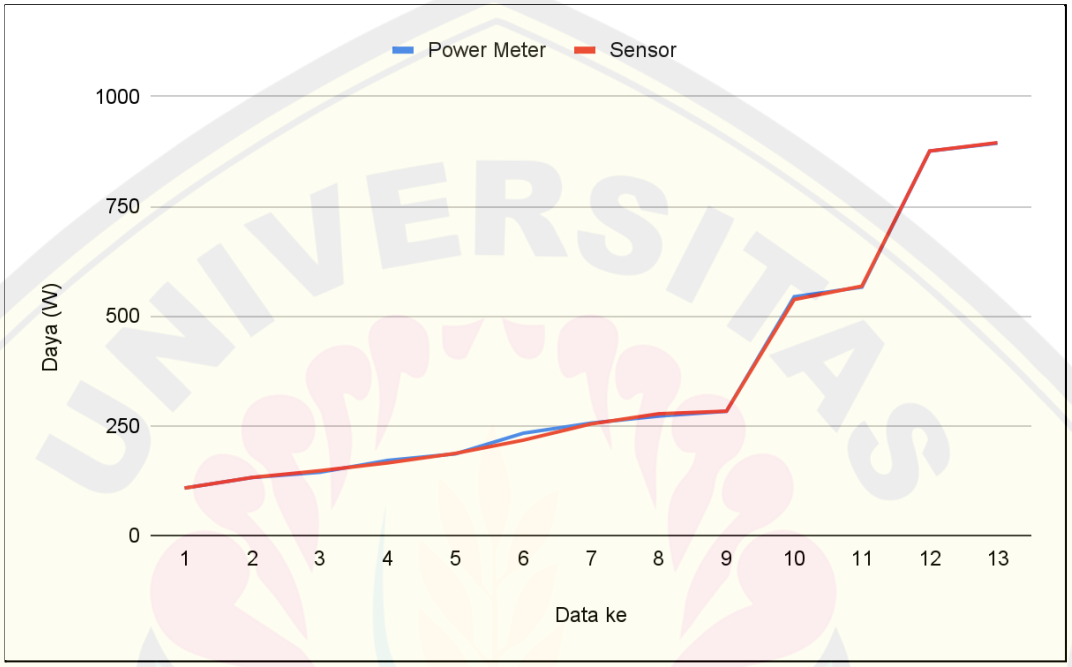
$$P = V \times I \times cos\,\varphi \quad \text{.....(4.2)}$$

Hasil pengujian *monitoring* nilai daya yang telah dilakukan pada penelitian ini dapat dilihat pada tabel berikut:

Tabel 4.20 Pengujian *Monitoring* Nilai Daya

Power Meter (W)	**Sensor (W)**	**Error (%)**
108	107,9	0,07
132	132,2	0,13
144	147,5	2,46
171	165,3	3,31
186	187,0	0,56
233	217,1	6,81
256	254,1	0,74
272	277,3	1,93
283	283,2	0,07
544	537,6	1,17
566	568,2	0,38
876	876,1	0,01
894	895,0	0,11
Rata-rata eror (%)		**1,37**

Berdasarkan dari tabel pengujian *monitoring* diatas diketahui nilai hasil perhitungan dari pembacaan sensor yang ditampilkan telah mendekati dengan pembacaan power meter. Rata-rata eror pada pengujian ini sebesar 1,37% dengan eror terbesar yaitu pada data ke-6 sebesar 6,81% dan eror terkecil yaitu pada data ke-12 sebesar 0,01%.



Gambar 4.20 Grafik Perbandingan Nilai Daya Sensor dan Daya Ukur

4.2.2 Pengujian Pengontrolan

Pada sistem SCADA salah satu fungsi yang terdapat dalam sistemnya yaitu fungsi pengontrolan. Pada penelitian ini fungsi pengontrolan berupa kontrol relai pada RTU dan kontrol berupa pemberian nilai pada kode fungsi Holding Register. Pengontrolan pada relai pada penelitian ini merupakan pengontrolan penyalaan maupun pemadaman pada beban yang dilakukan melalui MTU, beban yang dapat dikontrol antara lain lampu, kipas angin, AC atau beban lainnya yang tidak mengkonsumsi arus yang melebihi dari 30A. Sehingga pengujian ini lebih difokuskan pada pengujian secara manual yaitu dengan memberikan suatu perintah pada RTU melalui interface yang tersedia pada MTU secara *wireless*. Sedangkan untuk pengujian yang otomatis yaitu berdasarkan kondisi pada lapangan yang meliputi waktu penyalaan dan besar daya yang digunakan akan

dipaparkan pada pengujian keseluruhan. Pengujian ini dilakukan dengan mencatat selisih waktu pengiriman MTU dengan waktu eksekusi pada RTU. Pengujian ini bertujuan untuk mengetahui performa respon RTU terhadap perintah kontrol MTU.

Tabel 4.21 Hasil Pengujian Pengontrolan

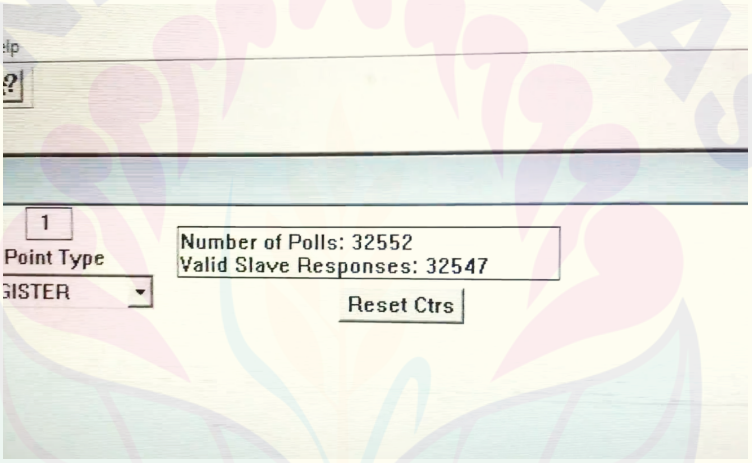
Kode Fungsi	Komponen	Kondisi	Pengujian 1	Pengujian 2
			Delay (detik)	Delay (detik)
01 Coil Status	Relai	Nyala	0,74	0,76
		Mati	1,02	1,12
03 Holding Register	Pemberian nilai register oleh MTU	790	0,40	0,33
		1234	0,24	0,29

Pada tabel 4.21 data hasil pengujian pengontrolan diperoleh dari perhitungan waktu dalam detik untuk RTU mengeksekusi permintaan dari MTU. Rata-rata dari hasil pengujian saat pengontrolan relai yaitu 0,91 detik. Sedangkan rata-rata dari hasil pengujian saat pengontrolan dengan pemberian nilai pada Holding Register sebesar 0,315 detik. Rata-rata delay dari keseluruhan pengujian pengontrolan yaitu sebesar 0,6125 detik.

Pada pengujian pengontrolan relai, delay yang tertulis adalah delay saat mikrokontroler ATmega telah mengeksekusi pada relai. Namun pada pengujian pengontrolan relai, delay yang dibutuhkan untuk RTU menerima data dari MTU yaitu berada pada nilai berkisar 0,24 detik yang diperoleh dari delay saat nilai sudah diterima oleh mikrokontroler ESP8266-01. Hanya saja komunikasi mikrokontroler ATmega mempunyai nilai delay yang sedikit lebih lama saat mengeksekusi untuk menyalakan relai, hal ini dipengaruhi oleh terlalu banyaknya proses yang harus dikerjakan oleh mikrokontroler ATmega saat mengolah data.

4.3 Pengujian Keseluruhan

Pengujian ini meliputi pengujian seluruh kinerja sistem yang dibuat untuk pengontrolan maupun pemantauan. Pengujian ini tidak dilakukan di Lantai 3 Gedung B Fakultas Teknik Universitas Jember dikarenakan terdapat peraturan untuk menyikapi pandemi yang sedang berlangsung sehingga pengujian ini dilakukan pada Laboratorium Fakultas Teknik Universitas Jember yang bertempat di Patrang. Terdapat 2 tahapan dalam pengujian ini yakni pengujian untuk pengontrolan beserta pemantauan dengan jumlah *polling* data yang banyak dan pengujian ketahanan kinerja dari RTU ketika dipasang pada sistem kelistrikan. Hasil dari pengujian pengontrolan dan pemantauan dapat dilihat pada gambar berikut ini:



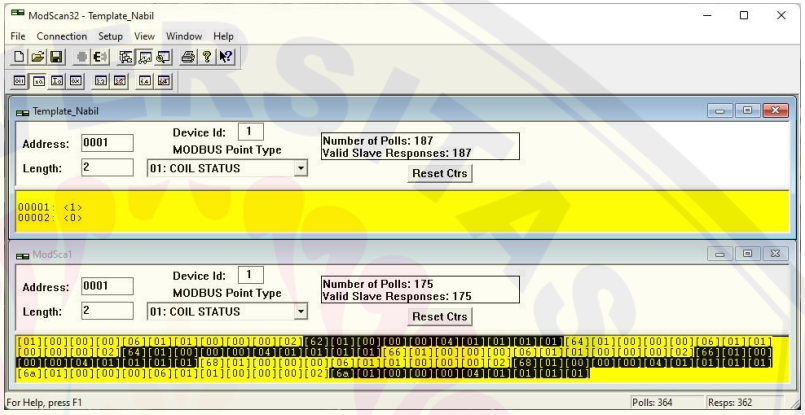
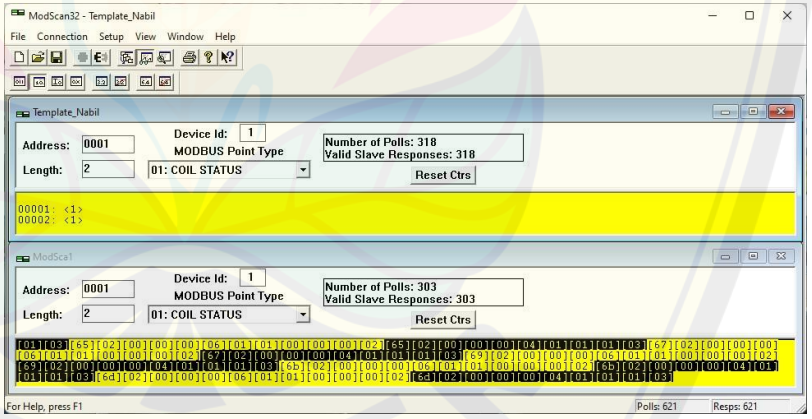
Gambar 4.21 Hasil Keberhasilan *Polling* Data

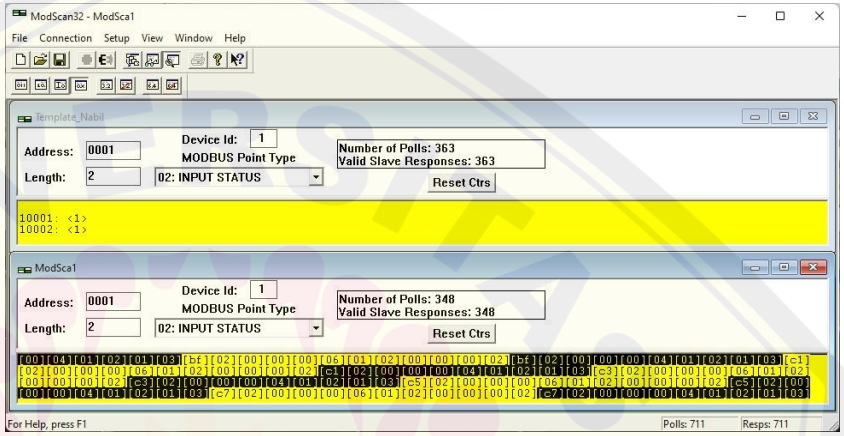
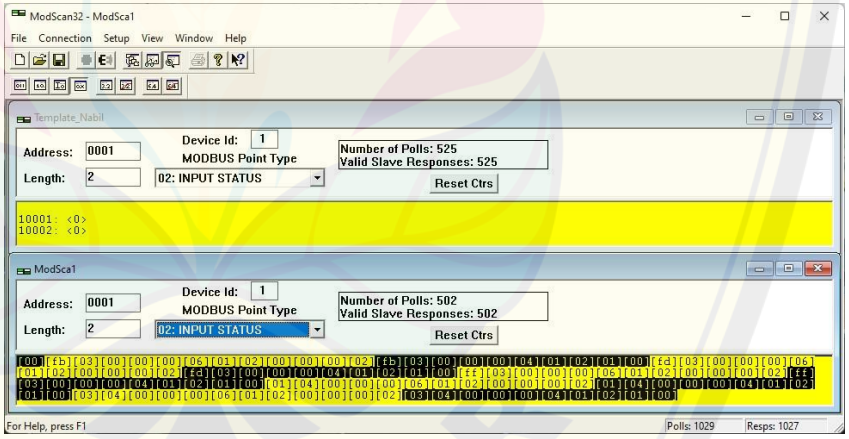
Pada gambar 4.21 menunjukkan bahwa jumlah *polling* data yang diminta oleh MTU dan jumlah yang berhasil direspon oleh RTU. Pada jumlah *polling* data yang ditampilkan pada *software* ModScan tersebut telah meliputi kegiatan pengontrolan dan pemantauan oleh MTU pada RTU. Pada gambar tersebut menunjukkan bahwa MTU telah melakukan *polling* data sebanyak 32.552 atau tiga puluh dua ribu lima ratus lima puluh dua data dan data yang berhasil di respon oleh RTU dan datanya yang telah dikatakan valid oleh *software* ModScan tersebut yaitu sebanyak 32.547 atau tiga puluh dua ribu lima ratus empat puluh tujuh data. Dimana waktu yang dibutuhkan oleh *software* ModScan setiap 1 kali

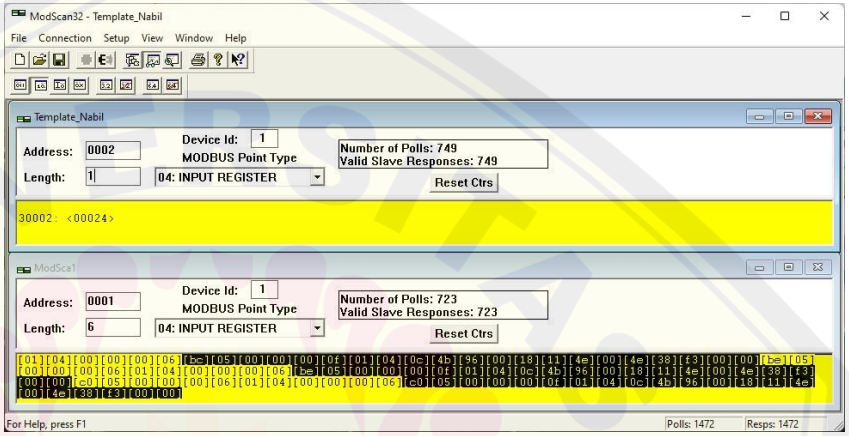
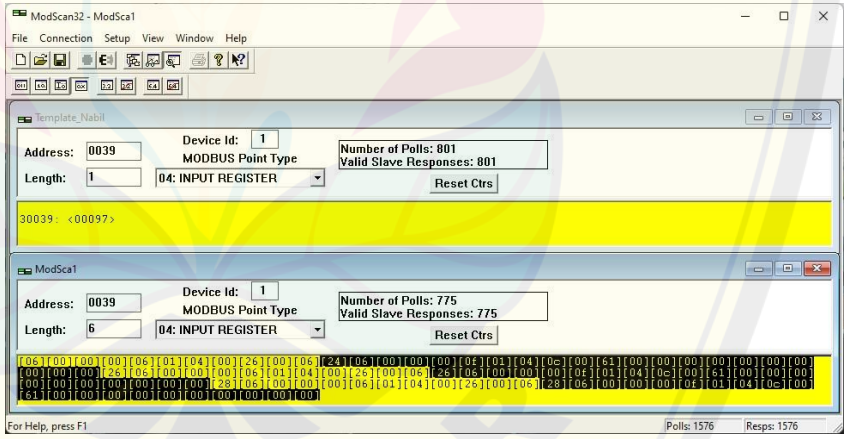
polling data yakni berkisar 1,5 detik. Dengan demikian dari jumlah *polling* data sebanyak 32.552, data yang gagal direspon oleh RTU dengan baik hanya 5 data. Sehingga pada kasus ini dapat dikatakan RTU yang telah dibangun pada penelitian ini telah mempunyai kinerja sistem yang sangat baik.

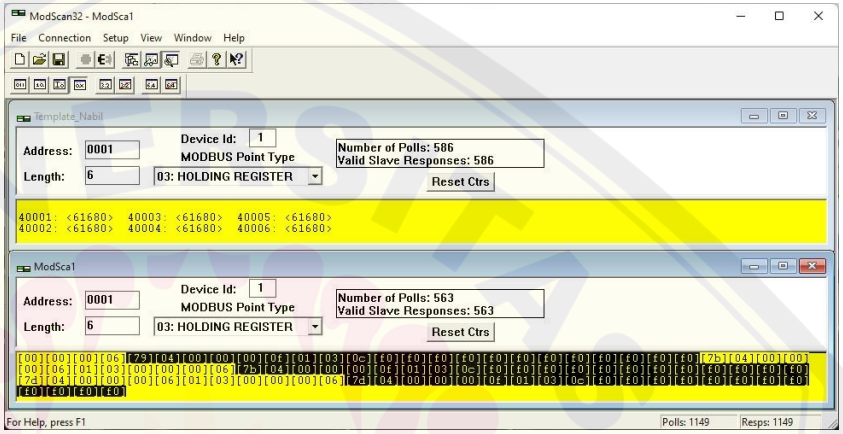
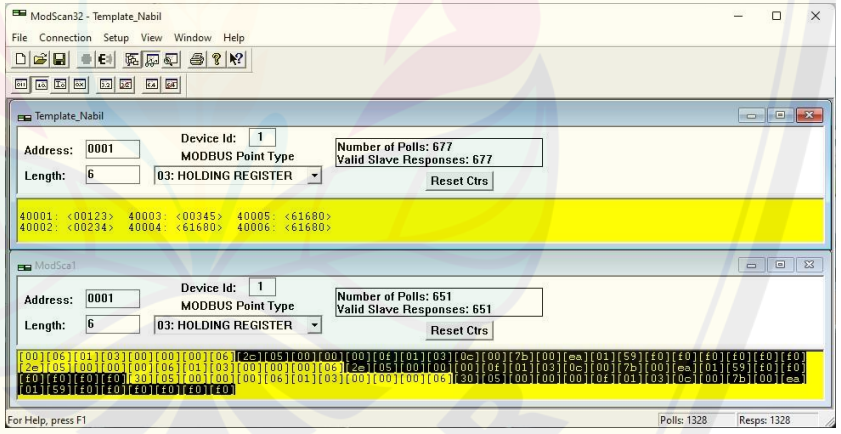
Pada keseluruhan pengujian tersebut yang termasuk dalam pengujian tahap kedua yakni hitungan waktu kinerja dari RTU. Pengujian ini dimulai pada pukul 20.45 WIB dan pengujian berakhir pada pukul 09.44 WIB. Dengan demikian dapat diketahui bahwa pada kegiatan pengujian ini RTU diuji kemampuannya dalam ketahanan melakukan *polling* data secara terus menerus secara *wireless* dan dengan waktu pengujian selama 13 jam telah menunjukkan hasil pengujian yang sangat baik. Hal ini juga diperkuat dengan keberhasilannya RTU dalam merespon permintaan data yang dilakukan oleh MTU. Dari ribuan data yang didapatkan selama pengujian 13 jam tersebut, berikut tampilan beberapa data trafik komunikasi *query* dan *response* antara MTU dengan RTU dapat dilihat pada tabel berikut:

Tabel 4.22 Trafik Data Komunikasi *Query* dan *Response*

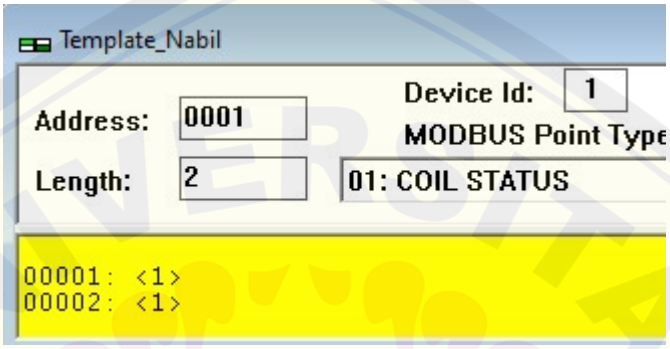
No	Kode Fungsi	Data dan Trafik Data
1	Coil Status	 Gambar 4.22 Data dan Trafik Data Coil Status Pengujian ke-1
		 Gambar 4.23 Data dan Trafik Data Coil Status Pengujian ke-2

No	Kode Fungsi	Data dan Trafik Data
2	Input Status	 Gambar 4.24 Data dan Trafik Data Input Status Pengujian ke-1
		 Gambar 4.25 Data dan Trafik Data Input Status Pengujian ke-2

No	Kode Fungsi	Data dan Trafik Data
3	Input Register	 Gambar 4.26 Data dan Trafik Data Input Register Pengujian ke-1  Gambar 4.27 Data dan Trafik Data Input Register Pengujian ke-2

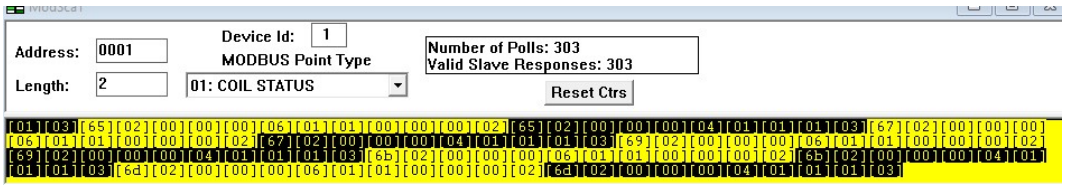
No	Kode Fungsi	Data dan Trafik Data
4	Holding Register	 Gambar 4.28 Data dan Trafik Data Holding Register Pengujian ke-1  Gambar 4.29 Data dan Trafik Data Holding Register Pengujian ke-2

Berdasarkan trafik data komunikasi pada tabel 4.22 Trafik Data Komunikasi *Query* dan *Response* dapat diketahui bahwa permintaan data oleh MTU dapat direspon dengan baik oleh RTU yang telah dibangun menggunakan protokol modbus. Sebagai keperluan analisis data trafik komunikasi, maka dapat diambil salah satu data dari tabel diatas misalnya pada data pengujian ke-2 coil status. Data pengujian ke-2 coil status dapat dilihat pada gambar berikut:



Gambar 4.30 Analisis Data Komunikasi Coil Status

Pada gambar 4.30 diatas merupakan data yang tertampil pada *interface* modscan. Pada gambar tersebut diketahui bahwa alamat dari RTU yang diinginkan datanya adalah RTU dengan ID 1 dapat dilihat pada kolom "Device Id". Selain itu data yang ditampilkan dimulai dari alamat register 00001 hingga 00002 yakni terdiri dari 2 data, dimana 2 data ini adalah sesuai dengan pemberian nilai pada kolom "Length" dan kode fungsinya yaitu 01 yang berarti coil status. Untuk mengetahui format data *query* yang dikirimkan oleh MTU pada RTU maupun format *respon* yang dikirimkan oleh RTU pada MTU maka dapat dilihat pada jendela trafik data, dimana jendela yang bawah tersebut digunakan sebagai MTU yang kedua dengan tujuan hanya menampilkan trafik data komunikasi sehingga *number of pools* antara jendela pertama dengan jendela yang kedua tersebut berbeda. Jendela MTU ke-2 dapat dilihat seperti gambar berikut ini:



Gambar 4.31 Analisis Trafik Data Komunikasi Coil Status

Gambar 4.32 Analisis Data *Query* Coil Status

Gambar 4.33 Analisis Data *Response* Coil Status

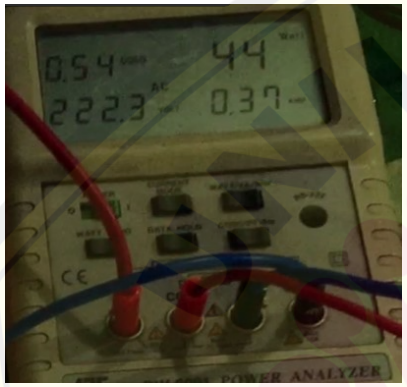
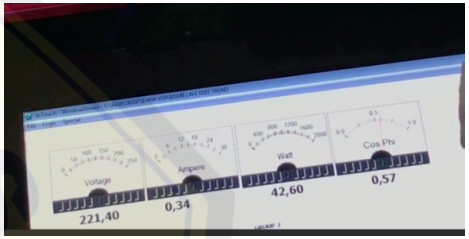
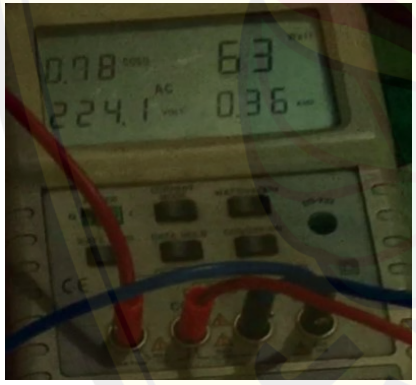
Trafik data tersebut ditampilkan dalam bentuk bilangan hexadesimal. Seperti yang telah dipaparkan pada Bab 2 pada gambar 2.2 *Frame* Data Modbus TCP/IP dimana untuk protokol modbus TCP/IP mempunyai frame data yang berbeda dengan modbus serial. Pada gambar 4.32 yaitu frame data permintaan oleh MTU dapat dilihat bahwa [6d][02] berlaku sebagai *Transaction Identifier* yang jika diubah dalam nilai desimal yaitu 27906 dimana nilai tersebut merupakan nilai pengenal nomor transaksi data yang akan dilakukan pada saat itu. Kemudian [00][00] adalah *Protocol Identifier* yang digunakan pada komunikasi ini yaitu protokol standar dari modbus TCP/IP. Setelah itu [00][06] disitu sebagai *Length Field* atau *Message Length* yang berarti jumlah data komunikasi yang dikirim setelah data [00][06] tersebut, [00][06] dalam desimal yaitu 6 yang berarti ada 6 data yang akan dikirimkan setelahnya dapat dilihat 6 data tersebut dimulai dari [01] hingga [02] yang berjumlah 6 data komunikasi. Kemudian [01] yang pertama bertindak sebagai *Unit ID* yaitu ID RTU pada penelitian ini adalah 1. Sedangkan [01] yang kedua adalah *Function Code* dimana 01 adalah fungsi kode untuk coil status dapat dilihat seperti pada gambar 4.31 pada bagian *point type* yaitu 01 coil status. Kemudian [00][00] adalah alamat data dari register pertama yang diminta berarti dikarenakan nilainya adalah 0 maka register yang diminta pertama yaitu 0001 ditambah dengan 0 berarti alamat register pertama yang diminta yaitu 0001. Kemudian [00][02] bertindak sebagai jumlah total register

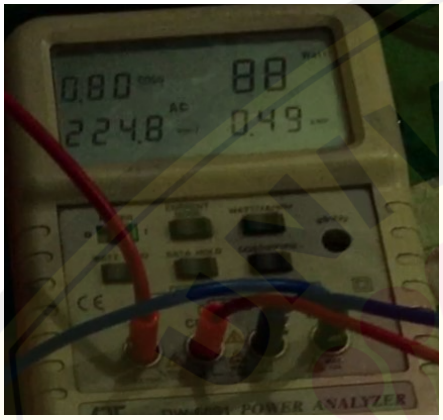
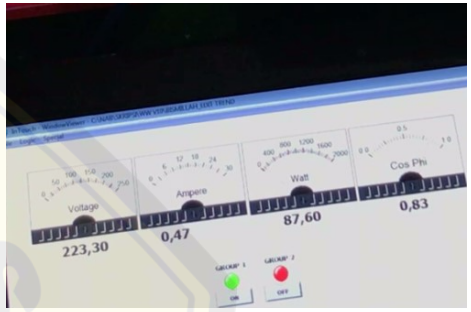
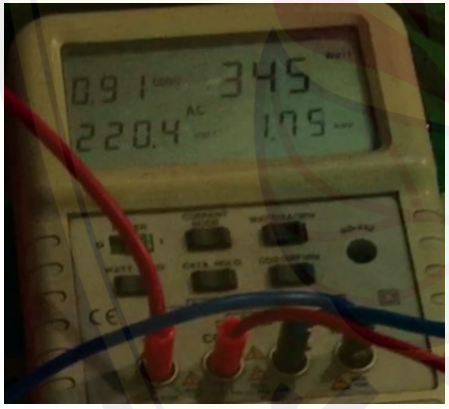
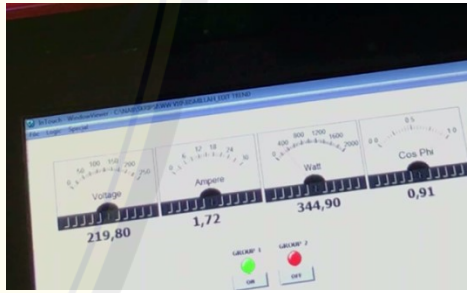
yang diminta yaitu 2 register berarti alamat yang diminta datanya dari kode fungsi status adalah 0001 hingga 0002 hal ini telah sesuai dengan gambar 4.30.

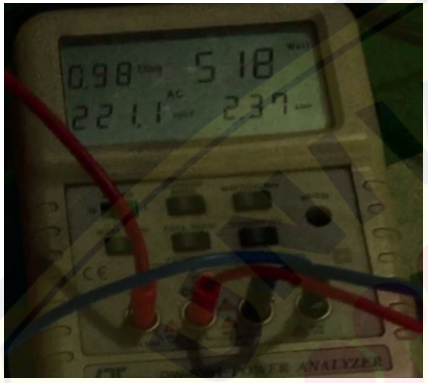
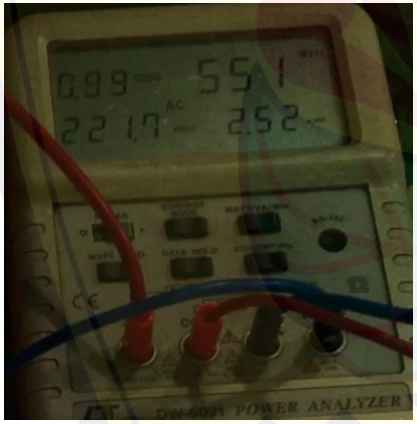
Kemudian untuk data *response* yang dikirim oleh RTU yang telah dibangun pada penelitian ini untuk menindaklanjuti permintaan dari MTU tersebut dapat dilihat pada gambar 4.33. Dapat dilihat bahwa [6d][02] disini sama seperti data *query* oleh MTU yaitu sebagai *Transaction Identifier* yang jika diubah dalam nilai desimal yaitu 27906 dimana nilai tersebut merupakan nilai pengenal nomor transaksi data yang akan dilakukan pada saat itu dan nilai pengenal transaksi ini sudah sama dengan permintaan MTU. Kemudian [00][00] adalah *Protocol Identifier* yang digunakan pada komunikasi ini yaitu protokol standar dari modbus TCP/IP sama seperti yang diminta oleh MTU. Setelah itu [00][04] disitu sebagai *Length Field* atau *Message Length* yang berarti jumlah data komunikasi yang dikirim setelah data [00][04] tersebut, [00][04] dalam desimal yaitu 4 yang berarti ada 4 data yang akan dikirimkan setelahnya dapat dilihat 4 data *response* tersebut dimulai dari [01] hingga [03] yang berjumlah 4 data komunikasi. Kemudian [01] yang pertama bertindak sebagai *Unit ID* yaitu sesuai dengan permintaan dari MTU. Sedangkan [01] yang kedua adalah *Function Code* dimana 01 adalah fungsi kode untuk coil status sesuai dengan permintaan dari MTU juga. Kemudian [01] yang ketiga menunjukkan jumlah data hexadesimal yang dikirim sebagai data dari *Protocol Data Unit* (PDU) seperti yang telah dipaparkan pada bab 2 sebelumnya. Dengan demikian diketahui bahwa nilai data dari 2 alamat register yang diminta oleh MTU telah dijawab oleh RTU hanya dengan 1 data hexadesimal yang terakhir yaitu [03]. Dikarenakan data pada kode fungsi coil status ini bertipe bilangan biner maka data [03] tersebut dikonversikan ke bilangan biner yaitu menjadi 11, dalam hal ini berarti nilai register pada alamat 0001 adalah 1 dan nilai dari register pada alamat 0002 adalah 1. Sehingga dengan adanya pengujian *frame data* komunikasi pada pengujian keseluruhan ini telah menunjukkan bahwa RTU yang telah dibangun telah mampu berkomunikasi dengan MTU menggunakan protokol modbus TCP/IP dengan sangat baik.

Pada pengujian keseluruhan ini beban yang tersambung telah bervariasi antara lain kipas angin 2 buah yaitu 20W dan 25W, lampu mercury 2 buah masing masing 250W, *vacum cleaner* dengan spesifikasi 600W, dan masih banyak beban lainnya yang sama seperti pada pengujian pembacaan sensor pada sub bab sebelumnya . Dari ribuan data yang didapatkan selama pengujian 13 jam tersebut, berikut tampilan beberapa dokumentasi nilai data yang telah didapatkan pada RTU maupun MTU dapat dilihat pada tabel berikut:

Tabel 4.23 Dokumentasi Pengujian Keseluruhan

Beban (W)	Alat Ukur	RTU	MTU
38	 Gambar 4.34 Dok. PA 38W	 Gambar 4.35 Dok. RTU 38W	 Gambar 4.36 Dok. MTU 38W
58	 Gambar 4.37 Dok. PA 58W	 Gambar 4.38 Dok. RTU 58W	 Gambar 4.39 Dok. MTU 58W

Beban (W)	Alat Ukur	RTU	MTU
83	 Gambar 4.40 Dok. PA 83W	 Gambar 4.41 Dok. RTU 83W	 Gambar 4.42 Dok. MTU 83W
333	 Gambar 4.43 Dok. PA 333W	 Gambar 4.44 Dok. RTU 333W	 Gambar 4.45 Dok. MTU 333W

Beban (W)	Alat Ukur	RTU	MTU
645	 Gambar 4.46 Dok. PA 645W	 Gambar 4.47 Dok. RTU 645W	 Gambar 4.48 Dok. MTU 645W
683	 Gambar 4.49 Dok. PA 683W	 Gambar 4.50 Dok. RTU 683W	 Gambar 4.51 Dok. MTU 683W

Dokumentasi gambar pada tabel 4.23 diatas didapatkan dari video dokumentasi yang telah dilakukan secara bersamaan. Berhubungan dengan keperluan membandingkan hasil pembacaannya diperlukan sebuah alat ukur yang mampu membaca berbagai besaran yang diperlukan tersebut dan sedangkan alat ukur yang tersedia pada laboratorium teknik elektro Universitas Jember hanya terdapat 1 buah sehingga pada tabel tersebut diketahui bahwa data yang ditampilkan pada RTU maupun MTU tersebut hanya menampilkan dari 1 pembacaan sensor saja. Berdasarkan pada tabel tersebut dapat diketahui selisih hasil pembacaan daya dari sensor dengan daya perhitungan dapat dilihat pada tabel berikut :

Tabel 4.24 Perbandingan Daya Perhitungan dengan Daya Terukur Sensor

No	Tegangan (V)	Arus (A)	Cos Phi	Daya Perhitungan (W)	Daya Terukur Sensor (W)	Error (%)
1	221,40	0,34	0,57	42,91	42,60	0,72
2	222,20	0,36	0,78	62,39	62,50	0,17
3	223,30	0,47	0,83	87,11	87,60	0,56
4	219,80	1,72	0,91	344,03	344,90	0,25
5	220,40	2,38	0,99	519,31	518,90	0,08
6	222,00	2,51	0,99	551,65	551,10	0,10

Perhitungan daya tersebut menggunakan rumus (4.2) sehingga berdasarkan tabel diatas diketahui selisih daya perhitungan dengan daya yang terukur oleh sensor mempunyai rata-rata error yaitu 0,31% dengan error terkecil yaitu 0,10% dan error terbesar berdasarkan pada tabel diatas adalah 0,72%.

Pada RTU yang telah dibuat sudah dilengkapi dengan adanya fitur kontrol otomatis. Kontrol otomatis dirancang pada kode fungsi coil status dengan alamat register 00001 dan dipengaruhi oleh 2 variabel yakni kontrol otomatis berdasarkan waktu serta kontrol otomatis berdasarkan setting daya maksimum.

Sehingga ketika pada pukul 05.00 WIB maka beban yang terhubung dengan RTU akan otomatis menyala dan ketika pukul 22.00 WIB maka beban yang terhubung dengan RTU akan otomatis mati. Selain itu, ketika daya total yang terbaca melebihi daya maksimal yang telah ditentukan sebelumnya (6600 W) maka relai akan otomatis memutus arus sehingga beban yang terhubung akan mati ataupun daya maksimum tersebut dapat diatur dengan cara memberikan nilai daya maksimum yang diinginkan pada kode fungsi holding register alamat 00001 seperti yang telah disebutkan diatas.

Tabel 4.25 Data Hasil Pengujian Kontrol Otomatis Berdasarkan Aturan Waktu

No.	Waktu (WIB)		Kondisi Beban		Status
	Setting	Ambil Data	Awal	Akhir	
1	05.00 - 22.00	05.00	Mati	Nyala	OK
2	06.00 - 22.00	05.00	Mati	Mati	OK
3	06.00 - 22.00	06.00	Mati	Nyala	OK
4	05.00 - 22.00	20.13	Nyala	Nyala	OK
5	05.00 - 22.00	21.03	Nyala	Nyala	OK
6	05.00 - 22.00	21.59	Nyala	Mati	OK
7	05.00 - 22.00	21.59	Nyala	Mati	OK
8	05.00 - 22.00	23.05	Mati	Mati	OK
9	05.00 - 22.00	23.09	Mati	Mati	OK

Tabel 4.26 Data Hasil Pengujian Kontrol Otomatis Berdasarkan Aturan Daya Maksimum

No.	Daya Total (Watt)		Waktu (WIB)	Kondisi Beban		Status
	Maks	Sensor		Awal	Akhir	
1	20	12 x 2	5.32	Nyala	Mati	OK
2	40	12 x 2	5.33	Mati	Nyala	OK
3	30	12 x 2	5.27	Mati	Nyala	OK
4	100	12 x 2	5.21	Mati	Nyala	OK
5	0	12 x 2	5.20	Nyala	Mati	OK

Berdasarkan tabel 4.25 data pertama diketahui dengan setting waktu menyala yaitu 05.00-22.00 WIB dan ketika pukul 05.00 WIB dengan beban dalam kondisi awal mati maka beban otomatis menyala sedangkan saat pukul 22.00 WIB maka beban akan otomatis mati meskipun daya total yang terbaca oleh sensor belum melebihi batas daya maksimum yang telah ditentukan. Salah satu dokumentasi pada percobaan ini dapat dilihat pada link video berikut ini: https://unej.id/kontrolwaktu.

Kemudian pada percobaan selanjutnya berdasarkan tabel 4.26 baris pertama saat daya total yang terbaca oleh sensor sebesar 24W dan setting daya maksimumnya adalah 20W pada pukul 5.32 WIB maka beban dengan kondisi awal menyala akan mati, karena daya total yang terbaca oleh sensor telah melebihi daya maksimum yang telah ditentukan sebelumnya.Salah satu dokumentasi pada percobaan ini dapat dilihat pada link video berikut ini: https://unej.id/kontroldaya.

4.4 Analisis Sistem SCADA Secara Keseluruhan

Dalam membangun sebuah RTU yang menggunakan protokol komunikasi Modbus TCP/IP berbasis *wireless* perlu memperhatikan tahapan-tahapan dalam perancangan sistem elektronika dan perancangan kode program. Tahapan-tahapan

dan kemampuan untuk menguasai sistem elektronika sangat diperlukan dalam penelitian ini, perancangan sistem elektronika yang sangat dibutuhkan pada penelitian ini yaitu pada perancangan pembacaan sensor arus YHDC SCT-013 supaya nilai yang didapatkan dapat mendekati nilai pembacaan dari multimeter. Selain itu dalam perancangan sistem elektronika, dibutuhkan perancangan rangkaian sumber tegangan yang memiliki kestabilan yang terjamin, hal ini dikarenakan pada RTU terdapat dua buah mikrokontroler yang mempunyai spesifikasi tegangan kerja yang berbeda dan pada RTU terdapat banyak komponen yang terhubung. Sehingga sumber tegangan yang stabil sangat dibutuhkan untuk mendapatkan kinerja RTU yang maksimal dan untuk mendapatkan umur kerja RTU yang lebih lama. Sehingga pada perancangan sumber tegangan untuk RTU pada penelitian ini dilengkapi dengan kapasitor 1000µF pada jalur tegangan 5V dan kapasitor 100nF pada jalur tegangan 3,3V yang menggunakan IC regulator AMS1117 seperti yang terlihat pada gambar 3.18 rangkaian keseluruhan sistem minimum. Dengan adanya kapasitor yang telah disebutkan diatas maka RTU akan mempunyai kemampuan untuk menyimpan sedikit tegangan yang mana hal ini berperan ketika tegangan sumber terdapat kendala yang dapat mengakibatkan sistem pada RTU mati, hal ini juga telah terjadi saat penelitian ini dilakukan, meskipun sumber daya mengalami adanya pemadaman sejenak yang mengakibatkan beban lampu padam namun sistem pada RTU masih tetap berjalan dengan baik karena adanya komponen kapasitor dan induktor pada rangkaian skematik RTU hanya saja kejadian ini tidak terdokumentasikan oleh penulis.

Alur struktur dari perancangan program telah diatur dalam perancangan flowchart. Sehingga dalam membangun program, flowchart sangat berperan dalam menterjemahkan proses berjalannya program. Setiap subsistem perancangan program telah dipaparkan pada bab sebelumnya dan berdasarkan seluruh pengujian dapat dikatakan bahwa program yang terbangun meliputi program pembacaan sensor, pengambilan data waktu, pengontrolan, pengawasan, dan subprogram lainnya telah mencapai seperti yang diharapkan. Sistem SCADA merupakan sistem kendali beserta *monitoring* sistem jarak jauh secara *real time*.

Pada sistem SCADA mempunyai 3 fungsi utama yaitu fungsi *telecontrol, telemetering,* dan *telesignal*. Pada fungsi *telecontrol* yang berarti pada fungsi ini melakukan kegiatan pengontrolan pada sistem. Pada penelitian ini fungsi kontrol digunakan untuk melakukan pengontrolan pada relai dan pemberian nilai register pada kode fungsi Holding Register. Berdasarkan hasil pengujian yang telah dilakukan fungsi kontrol telah dapat bekerja dengan cukup baik. Perintah yang dikirim oleh MTU ke RTU dapat direspon dengan mengeksekusi relai dengan *delay* yang terbilang masih cukup pendek dan dapat dilakukan secara *real-time*. Kemudian pada fungsi *telemetering* pada penelitian ini digunakan untuk melakukan pembacaan nilai tegangan, arus, dan daya. Berdasarkan hasil pengujian yang telah dilakukan, RTU dapat merespon dengan baik perintah dari MTU untuk melakukan pengiriman data hasil pembacaan sensor yang terhubung dengan mikrokontroler pada RTU. Keberhasilan pada fungsi pengujian *telemetering* ini telah ditunjukkan pula dengan ditampilkannya nilai hasil pembacaan sensor pada RTU pada *software* ModScan dan nilai yang diberikan dilakukan secara *real-time*. Kemudian pada fungsi *telesignal* yang merupakan proses pembacaan status pada MTU. Berdasarkan pengujian yang telah dilakukan menunjukkan sistem telah berjalan dengan biak dengan *delay* yang cukup pendek. Pada pengujian ini *delay* sangat dipengaruhi oleh program yang telah dirancang pada mikrokontroler yang bertempat di RTU, mengingat bahwa sistem yang dibangun pada penelitian ini cukup kompleks dan dibangun dengan sistem komunikasi secara *wireless*, sehingga *delay* yang dibutuhkan cukup lama namun masih dapat dilakukan secara *real time*.

Pada hasil pengujian keandalan sistem dalam berkomunikasi secara *wireless* terhadap jarak dan halangan yang diberikan menunjukkan kinerja dari sistem yang telah dibangun telah mempunyai kemampuan yang baik. Dimulai dari jarak antar perangkat 10 meter hingga 100 meter, seluruh proses komunikasi data dapat diselesaikan dengan baik. Kemudian berdasarkan data yang tertulis pada tabel 4.17 dan tabel 4.18 semuanya terbilang cukup lancar. Namun merujuk pada tabel 4.18 menunjukkan bahwa hasil pengujian komunikasi pada kondisi dengan

halangan dan pada jarak 100 meter, dari *polling* data yang dilakukan sebanyak 5 kali pada jarak tersebut hanya berhasil 3 kali, sehingga pada tabel tersebut tertulis kurang lancar. Sedangkan komunikasi ketika tanpa halangan dan dengan halangan yang lainnya menunjukkan komunikasi antara RTU dan MTU secara *wireless* dapat terselesaikan dengan lancar.

Berdasarkan hasil dari sistem SCADA yang telah dibangun, sistem berjalan dengan baik. Berdasarkan dari pengujian yang telah dilakukan pada fungsi *monitoring* nilai besaran listrik mempunyai rata-rata eror yang tergolong sangat kecil yakni sebesar 1,37% dengan rata-rata delay pada pengujian pertama sebesar 2,47 detik dan pada pengujian kedua sebesar 2,65 detik sehingga informasi yang ditampilkan pada *interface* yang terdapat pada MTU tidak beda jauh dengan nilai hasil pembacaan alat ukur. Sedangkan untuk hasil pengujian pada fungsi pengontrolan memiliki delay rata-rata yang tergolong masih cukup singkat, pada pengujian saat pengontrolan dengan pemberian nilai pada Holding Register sebesar 0,315 detik, sedangkan rata-rata dari hasil pengujian saat pengontrolan relai yaitu 0,91 detik. Pada pengujian pengontrolan relai, delay yang tertulis adalah delay saat mikrokontroler ATmega telah mengeksekusi pada relai. Namun pada pengujian pengontrolan relai, delay yang dibutuhkan untuk RTU menerima data dari MTU yaitu berada pada nilai 0,24 detik yang diperoleh dari delay saat nilai sudah diterima oleh mikrokontroler ESP8266-01. Hanya saja komunikasi mikrokontroler ATmega mempunyai nilai delay yang sedikit lebih lama saat mengeksekusi untuk menyalakan relai, hal ini dipengaruhi oleh terlalu banyaknya proses yang harus dikerjakan oleh mikrokontroler ATmega saat mengolah data.

Analisis berdasarkan hasil pengujian *frame data* komunikasi pada pengujian keseluruhan pada sub bab sebelumnya menunjukkan bahwa data komunikasi yang dikirimkan oleh MTU pada RTU telah dapat dibaca dengan baik oleh RTU. Hal ini ditunjukkan dengan keberhasilan RTU ketika mengirim data respon pada MTU, dimana data respon yang dikirim oleh RTU yang telah dibangun dapat diterjemahkan dengan baik oleh MTU dan datanya sudah bisa

ditampilkan dengan baik pada *interface* yang terdapat pada MTU yaitu Modscan. Selain itu data yang dikirimkan oleh RTU juga telah sesuai dengan data yang terdapat pada lapangan, sehingga hal ini menunjukkan bahwa perancangan komunikasi pada RTU telah sesuai dengan aturan protokol *frame data* dari Modbus TCP/IP.

Penambahan fitur pengontrolan yang dapat bekerja secara otomatis, dapat membantu dalam manajemen energi yang digunakan. Pada perancangan pengontrolan berdasarkan daya, dirancang nilai daya maksimum dapat diubah pada sebuah register. Perancangan ini ditempatkan pada holding register karena kode fungsi tersebut merupakan kode fungsi yang memungkinkan untuk menyimpan data maupun menyimpan nilai yang bervariasi. Sehingga dengan dapat diaturnya nilai daya maksimum yang diinginkan dapat digunakan jika sewaktu-waktu ingin membatasi daya penggunaan pada beban dengan demikian fitur ini dapat mempermudah kedepannya dalam manajemen energi. Begitu pula dengan adanya pengontrolan nyala atau matinya sebuah beban berdasarkan waktu, hal ini sangat bermanfaat untuk diterapkan karena pengguna dapat mengatur waktu penyalaan dan membatasi waktu nyala tersebut berdasarkan waktu yang diinginkan.

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dari seluruh pengujian yang telah dilakukan pada penelitian ini, didapatkan beberapa kesimpulan yaitu:

1. Perancangan *Remote Terminal Unit* (RTU) SCADA berhasil dilakukan dengan pembangunan berupa sistem minimum yang cukup kompleks telah menunjukkan kinerja yang baik. Hal ini dapat dibuktikan dengan melihat tabel 4.24 yang menunjukkan selisih daya perhitungan dengan daya yang terukur oleh sensor mempunyai rata-rata error yaitu 0,31 persen.
2. Realisasi RTU dengan 8 grup beban berhasil dilakukan dengan rata-rata error pembacaan daya dari sensor sebesar 1,37 persen seperti pada tabel 4.20, sedangkan rata-rata delay dari keseluruhan pengujian pengontrolan yaitu sebesar 0,6125 detik seperti pada tabel 4.21 Hasil Pengujian Pengontrolan, sedangkan delay rata-rata yang dibutuhkan oleh RTU pada proses pengiriman data yakni sebesar 2,56 detik dengan delay terlambat adalah 5,08 detik seperti pada tabel 4.19 Hasil Pengujian Pengawasan.

5.2 Saran

Pada penelitian desain dan implementasi *Remote Terminal Unit* (RTU) SCADA berbasis *wireless* terdapat beberapa saran dari penulis dengan harapan pada penelitian selanjutnya dapat lebih membuahkan hasil yang sangat maksimal, yaitu:

1. Pemilihan mikrokontroler yang digunakan sebagai RTU mempunyai memori yang cukup besar dan mendukung komunikasi data yang banyak sehingga proses pengolahan data dapat lebih cepat dan proses komunikasi pada RTU dapat dilakukan lebih cepat lagi.
2. Sistem dapat dikembangkan dan diterapkan pada pemrosesan data yang lebih kompleks sehingga diharapkan penelitian yang dilakukan mempunyai manfaat yang terasa oleh orang lain.

3. Pemilihan komponen pada perancangan PCB dapat menggunakan komponen model SMD yang lebih kecil lagi sehingga ukuran RTU terlihat mempunyai desain yang lebih premium.
4. Parameter yang digunakan sebagai pengontrolan otomatis dapat ditambah dengan parameter selain berdasarkan waktu dan berdasarkan daya yang dikonsumsi supaya sistem yang dibangun mempunyai fitur yang lebih banyak dari penelitian ini.

DAFTAR PUSTAKA

Pahlevi, F.S.R. 2020 : Rancang Bangun *Human Machine Interface* (HMI) Pada Sistem Monitoring dan Kontrol Kelistrikan Gedung Perkantoran. *Skripsi.* Jember: Program Sarjana Fakultas Teknik Universitas Jember.

Permana, A.F. 2017 : Pintu Pemberitahu Kegiatan Ruangan Menggunakan HMI SCADA Berbasis Modul Mikrokontroler (Hardware Sistem Alarm dan Kunci Otomatis). Tugas Akhir. Bandung: Program Diploma Teknik Elektronika Politeknik Negeri Bandung.

Matti, W.E., Aziz J.S. 2012 : Design and Implementation of General Purpose Remote Terminal Unit (R.T.U). *Global Journals Inc.* 12 (7) : 27 - 34.

Jusoh, W.N.S.E.W., Ghani, M.R.Ab., Hanafiah, M.A.M., Raman. S.H. 2014 : Remote Terminal Unit (RTU) Hardware Design And Development For Distribution Automation System. *IEEE Innovative Smart Grid Technologies*. 572 - 576.

Boyer, S.A. 2004. SCADA Supervisory Control and Data Acquisition. 3rd ed. United States of America : ISA - The Instrumentation, System, and Automation Society.

LAMPIRAN

Lampiran 3.1 Listing Program Mikrokontroler ATmega328P-AU

```
#include <Nabil_PZEM.h>
#include "Nabil_SCT.h"
#include <SoftwareSerial.h>

SoftwareSerial DataESP(3, 2);

Nabil_PZEM pzem(&Serial);

ArusLibNabil ArusSCT1;
ArusLibNabil ArusSCT2;
ArusLibNabil ArusSCT3;
ArusLibNabil ArusSCT4;
ArusLibNabil ArusSCT5;
ArusLibNabil ArusSCT6;
ArusLibNabil ArusSCT7;

  int pin_sct1 = A0;
  int pin_sct2 = A1;
  int pin_sct3 = A2;
  int pin_sct4 = A3;
  int pin_sct5 = A4;
  int pin_sct6 = A5;
  int pin_sct7 = A6;

int n=0, b=0;
int PushButton1=4, PushButton2=5, valPB1=1, valPB2=1;
int Relay1=6, Relay2=7;
int BUZZER=10;
```

```
int statusWifi=0;
int keep_dulu;
int gantian=0;

double
voltage, currentpzem, power, pf, ArusRms1=0,
     DayaAktifClamp1=0,
     ArusRms2=0,
     DayaAktifClamp2=0,
     ArusRms3=0,
     DayaAktifClamp3=0,
     ArusRms4=0,
     DayaAktifClamp4=0,
     ArusRms5=0,
     DayaAktifClamp5=0,
     ArusRms6=0,
     DayaAktifClamp6=0,
     ArusRms7=0,
     DayaAktifClamp7=0;

void setup() {
  Serial.begin(9600);
  DataESP.begin(19200);
  //DataESP.begin(9600);
  //DataESP.setBaudRate(9600);  //NeoSW

  pinMode(pin_sct1, INPUT);
  pinMode(pin_sct2, INPUT);
  pinMode(pin_sct3, INPUT);
  pinMode(pin_sct4, INPUT);
  pinMode(pin_sct5, INPUT);
```

```
  pinMode(pin_sct6, INPUT);
  pinMode(pin_sct7, INPUT);

  ArusSCT1.current(pin_sct1, 45);
  ArusSCT2.current(pin_sct2, 45);
  ArusSCT3.current(pin_sct3, 45);
  ArusSCT4.current(pin_sct4, 45);
  ArusSCT5.current(pin_sct5, 45);
  ArusSCT6.current(pin_sct6, 45);
  ArusSCT7.current(pin_sct7, 45);

  pinMode(PushButton1, INPUT_PULLUP);
  pinMode(PushButton2, INPUT_PULLUP);
  pinMode(Relay1, OUTPUT);
  pinMode(Relay2, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(13, OUTPUT);

     digitalWrite(BUZZER, LOW); delay(25);
     digitalWrite(BUZZER, HIGH); delay(125);
     digitalWrite(BUZZER, LOW); delay(50);
     digitalWrite(BUZZER, HIGH); delay(125);
     digitalWrite(BUZZER, LOW); delay(50);
     digitalWrite(BUZZER, HIGH); delay(125);
     digitalWrite(BUZZER, LOW);
  delay(250);
  digitalWrite(Relay1, LOW);
  digitalWrite(Relay2, LOW);
  b=0;
}
```

```
void loop() {

  switch (statusWifi)
     {
     case 5: //wifi putus
     digitalWrite(BUZZER, HIGH); delay(100);
     digitalWrite(BUZZER, LOW); delay(50);
     digitalWrite(BUZZER, HIGH); delay(400);
     digitalWrite(BUZZER, LOW);
     statusWifi=0;
     break;
     case 4: //wifi terhubung
     digitalWrite(BUZZER, HIGH); delay(400);
     digitalWrite(BUZZER, LOW); delay(50);
     digitalWrite(BUZZER, HIGH); delay(100);
     digitalWrite(BUZZER, LOW);
     statusWifi=0;
     break;
     case 3: //perubahan data oleh master
     digitalWrite(BUZZER, HIGH); delay(100);
     digitalWrite(BUZZER, LOW); delay(20);
     digitalWrite(BUZZER, HIGH); delay(100);
     digitalWrite(BUZZER, LOW);
     statusWifi=0;
     break;
     default:
     break;
     }

  voltage = pzem.voltage();
  pf = pzem.pf();
```

```
      currentpzem = pzem.current();
      power = pzem.power();

//Clamp 1
  double simpanArusRms1 = ArusSCT1.calcIrms(1480);
  ArusRms1=kalibrasi(simpanArusRms1);
  DayaAktifClamp1 = ArusRms1*voltage*pf;
//Clamp 2
      double simpanArusRms2 = ArusSCT2.calcIrms(1480);
      ArusRms2=kalibrasi(simpanArusRms2);
      DayaAktifClamp2 = ArusRms2*voltage*pf;
//Clamp 3
  double simpanArusRms3 = ArusSCT3.calcIrms(1480);
  ArusRms3=kalibrasi(simpanArusRms3);
  DayaAktifClamp3 = ArusRms3*voltage*pf;
//Clamp 4
      double simpanArusRms4 = ArusSCT4.calcIrms(1480);
      ArusRms4=kalibrasi(simpanArusRms4);
      DayaAktifClamp4 = ArusRms4*voltage*pf;
//Clamp 5
  double simpanArusRms5 = ArusSCT5.calcIrms(1480);
  ArusRms5=kalibrasi(simpanArusRms5);
  DayaAktifClamp5 = ArusRms5*voltage*pf;
//Clamp 6
      double simpanArusRms6 = ArusSCT6.calcIrms(1480);
      ArusRms6=kalibrasi(simpanArusRms6);
      DayaAktifClamp6 = ArusRms6*voltage*pf;
//Clamp 7
  double simpanArusRms7 = ArusSCT7.calcIrms(1480);
  ArusRms7=kalibrasi(simpanArusRms7);
  DayaAktifClamp7 = ArusRms7*voltage*pf;
```

```
valPB1=digitalRead(PushButton1);
valPB2=digitalRead(PushButton2);
if(valPB1==0)
   {
   if(b==0) b=1;
   else b=0;
   }

if(gantian==0)
   {
   kirim_data(); delay(25);
   ambil_data();
   gantian=1;
   }
else if(gantian==1)
   {
   ambil_data();
   ambil_data();
   kirim_data(); delay(25);
   gantian=0;
   }

if(valPB1==0)
   {
   if(b>=1) digitalWrite(Relay1, HIGH);
   else if(b==0) digitalWrite(Relay1, LOW);
   digitalWrite(13, HIGH);
   delay(225);
   digitalWrite(13, LOW);
   }
```

```
  if(n>=1) digitalWrite(Relay2, HIGH);
  else if(n==0) digitalWrite(Relay2, LOW);
}

double kalibrasi(double nilai)
  {
     double hasil = ((nilai*1.084)-1.036)+1;
     return hasil;
  }


void kirim_data()
  {
     DataESP.println
     (
     String('^')
     + String(voltage)
     + String(" ")
     + String(currentpzem)
     + String(" ")
     + String(power)
     + String(" ")
     + String(ArusRms1)
     + String(" ")
     + String(DayaAktifClamp1)
     + String(" ")
     + String(ArusRms2)
     + String(" ")
     + String(DayaAktifClamp2)
     + String(" ")
     + String(ArusRms3)
```

```
      + String(" ")
      + String(DayaAktifClamp3)
      + String(" ")
      + String(ArusRms4)
      + String(" ")
      + String(DayaAktifClamp4)
      + String(" ")
      + String(ArusRms5)
      + String(" ")
      + String(DayaAktifClamp5)
      + String(" ")
      + String(ArusRms6)
      + String(" ")
      + String(DayaAktifClamp6)
      + String(" ")
      + String(ArusRms7)
      + String(" ")
      + String(DayaAktifClamp7)
      + String(" ")
      + String(valPB1)
      + String(" ")
      + String(valPB2)
      //UNTUK RELAY YG HANYA BISA DIBACA OLEH MASTER
      + String(" ")
      + String(b)
      );
  }


void ambil_data()
  {
```

```
    if(DataESP.available())
    {
    keep_dulu = DataESP.read();
    }
    //UNTUK Relay
    if (keep_dulu==1) n=keep_dulu;
    else if(keep_dulu<2)
    {
         n=keep_dulu;
    }
    //UNTUK BUZZER
    else
    {
         statusWifi=keep_dulu;
    }
    //DataESP.checkRxTime();
}
```

Lampiran 3.2 Listing Program Mikrokontroler ESP8266-01

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <ModbusIP_ESP8266.h>

int valrelay1, valrelay2;
int simpanrelay;
int kontrol=3, terhubung=4, terputus=5;
float Pf;
boolean prosesterhubung=false;
unsigned long cekkoneksi;

//coba bikin tipe data buat IP, semoga bisa
  IPAddress ip_masternya;

//OLED SCL>D1, SDA>D2
  #define OLED_RESET     -1
  #define SCREEN_WIDTH 128
  #define SCREEN_HEIGHT 64
  Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);
  #define NUMFLAKES      10  //kepingan untuk animasi

//>>>>>SETTINGAN COIL
STATUS=============================================
==============
  //yg ini MASTER HANYA MELIHAT
```

```
      uint8_t pin_coil[] = {valrelay1};
      #define banyak_data_coil
sizeof(pin_coil)/sizeof(uint8_t) //Master tidak bisa
mengontrol
      #define COIL_BASE 0
      //Fungsi callback, yg ini MASTER HANYA MELIHAT
      uint16_t cobaRead_coil(TRegister* reg, uint16_t
val)
      {
            //Lihat Modbus.h untuk definisi TRegister dan
TAddress
//             if(reg->address.address==0)
//             {
//             oled.setTextColor(SSD1306_BLACK,
SSD1306_WHITE);
//             oled.println(F("   COIL   "));
//             oled.println(F("  STATUS  "));
//             oled.setTextColor(SSD1306_WHITE);
//             oled.setTextSize(1);
//             oled.print(F("     Reg"));
//             oled.println(F("  Val"));
//             }
//             oled.print(F("     "));
//             oled.print(reg->address.address);
//             oled.println(simpanrelay);
//             oled.display();
            if(reg->address.address < COIL_BASE)
                  return 0;
            uint8_t offset = reg->address.address -
COIL_BASE;
            if(offset >= banyak_data_coil)
```

```
                return 0;
            return COIL_VAL(simpanrelay);
        }
        //Fungsi Callback, yg ini MASTER HANYA MELIHAT
        uint16_t cobaWrite_coil(TRegister* reg, uint16_t
val)
        {
            return reg->value;
        }
    //yg ini MASTER BISA MENULIS
        int save=0;
        uint8_t pin_coil2[] = {valrelay2};
        //uint8_t pin_coil2[] = {D5};
        #define banyak_data_coil2
sizeof(pin_coil2)/sizeof(uint8_t) ////Master bisa
mengontrol
        //Fungsi callback, yg ini MASTER BISA MENULIS
        uint16_t cobaRead_coil2(TRegister* reg, uint16_t
val)
        {
            if(val>0) save=1;
            else if(val==0) save=0;
            Serial.write(save); delay(5);
            Serial.write(save);
            //return
COIL_VAL(pin_coil2[(reg->address.address -
banyak_data_coil)]);
            //return val;
            return COIL_VAL(val);
        }
        //Fungsi Callback, yg ini MASTER BISA MENULIS
```

```
      uint16_t cobaWrite_coil2(TRegister* reg, uint16_t
val)
      {
            return COIL_VAL(val);
            //return val;
            //return reg->value;
      }

//>>>>>SETTINGAN UNTUK HOLDING
REGISTER=============================================
======
  #define banyak_data_HoldingRegister 6
  //Fungsi Callback untuk membaca data yang sesuai
      uint16_t cbRead_HoldingRegister(TRegister* regHR,
uint16_t valHR)
      {
      return valHR;
      }
  //Fungsi Callback untuk menulis data dengan proteksi
      uint16_t cbWrite_HoldingRegister(TRegister* regHR,
uint16_t valHR)
      {
      delay(500);
      return valHR;
      }

//>>>>>SETTINGAN UNTUK INPUT
STATUS===============================================
========
  //Alamat Register
      const int SWITCH_ISTS   = 0;
```

```
      const int SWITCH_ISTS2  = 1;
  //>>>>>COBA BUAT PROGRAM CALLBACK UNTUK INPUT
STATUS=====================================
      #define banyaknya_Ists 2
      #define nilai_offset 0
      uint16_t proses_Ists(TRegister* regIS, uint16_t
valIS)
      {

            return valIS;
      }

//>>>>>UNTUK PARSING KOMUNIKASI DATA SENSOR
  float tegangan, arusPZEM, dayaPZEM, arusClamp1,
dayaClamp1, arusClamp2, dayaClamp2,
      arusClamp3, dayaClamp3, arusClamp4, dayaClamp4,
arusClamp5, dayaClamp5,
      arusClamp6, dayaClamp6, arusClamp7, dayaClamp7;
  float k,l;

//>>>>>SETTINGAN UNTUK INPUT
REGISTER=================================================
========
  //Penempatan alamat register untuk masing2 data dari
sensor
      const int SENSOR_IREG   = 0;      //tegangan
      const int SENSOR_IREG2  = 1;      //arus PZEM
      const int SENSOR_IREG3  = 2;      //daya PZEM
      const int SENSOR_IREG4  = 3;      //arus Clamp1
      const int SENSOR_IREG5  = 4;      //daya Clamp1
      const int SENSOR_IREG6  = 12;     //arus Clamp2
```

```
      const int SENSOR_IREG7  = 13;      //daya Clamp2
      const int SENSOR_IREG8  = 14;   //arus Clamp3
      const int SENSOR_IREG9  = 15;   //daya Clamp3
      const int SENSOR_IREG10 = 16;      //arus Clamp4
      const int SENSOR_IREG11  = 24;     //daya Clamp4
      const int SENSOR_IREG12  = 25;     //arus Clamp5
      const int SENSOR_IREG13  = 26;     //daya Clamp5
      const int SENSOR_IREG14  = 27;     //arus Clamp6
      const int SENSOR_IREG15  = 28;     //daya Clamp6
      const int SENSOR_IREG16  = 36;  //arus Clamp7
      const int SENSOR_IREG17  = 37;  //daya Clamp7
  long TimerSampling;
  //>>>>>COBA BUAT PROGRAM CALLBACK UNTUK INPUT
REGISTER====================================
      #define banyaknya_Ireg 100
      #define nilai_offset_Ireg 0
      uint16_t proses_Ireg(TRegister* regIreg, uint16_t
valIreg)
      {
            return valIreg;
      }

//>>>>>NAMA MODBUS
  ModbusIP mbTCPWifi;

//>>>>>Fungsi Callback untuk menampilkan ip dari master
(ModScan atau lainnya)
  bool cbConn(IPAddress ip_master)
      {
      //prosesterhubung=true;
      //ip_masternya=ip_master;
```

```
      return true;
      }

//>>>>>UNTUK RESET KETIKA TERPUTUS
  void(*saya_reset) (void) = 0;
  bool cbDISConn(IPAddress ip_master)
      {
      oled.clearDisplay();
      oled.setTextColor(SSD1306_WHITE);
      oled.setTextSize(1);
      oled.setCursor(10,15);
      oled.println(F("Wah Master Terputus"));
      oled.setCursor(35,25);
      oled.println(F("IP Master ="));
      oled.setCursor(20,35);
      oled.println(ip_master);
      oled.display();
      delay(100);
      oled.clearDisplay();
      saya_reset();
      return true;
      }


//.....................................................
.....................................//
//.....................................>>>>>VOID SETUP
<<<<<.....................................//
//.....................................................
.....................................//
```

```
void setup()
  {
     Serial.begin(19200);

     Wire.begin(2, 0);
     if(!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C))
     {
     //Serial.println(F("LCD OLED tidak terpasang"));
     for(;;);
     }
     oled_menunggu();

     pinMode(LED_BUILTIN, OUTPUT);
     delay(1000);
     //WiFi.begin("Virus", "tanyapakrt");
     //WiFi.begin("KUCUR RIKO", "naizacell00");
     //WiFi.begin("NABIL BACHROIN", "gratis123");
     WiFi.begin("MODUL SCADA", "tenagasukses");

     while (WiFi.status() != WL_CONNECTED)
     {
     //Serial.println("...sabar dulu masih
menghubungkan...");
     delay(50);
     }
     gambar_loading();
     oled.clearDisplay();
     tampilkan_logo_lab();
     oled.clearDisplay();
     oled.setTextColor(SSD1306_WHITE);
     oled.setTextSize(1);
```

```
      oled.setCursor(13,15);
      oled.println(F("Berhasil Terhubung"));
      oled.setCursor(35,25);
      oled.println(F("IP Slave ="));
      oled.setCursor(20,35);
      oled.println(WiFi.localIP());
      oled.display();
      delay(500);
      oled.clearDisplay();
      oled.setCursor(0,0);
      oled.setTextColor(SSD1306_BLACK, SSD1306_WHITE);
      oled.print(F("   ")); oled.print(WiFi.localIP());
oled.println(F("   "));
      oled.display();
      tampilkan_logo_unej();

      //Cek koneksi dari master
      mbTCPWifi.onConnect(cbConn);
      mbTCPWifi.onDisconnect(cbDISConn);

      //Start koneksi Modbus
      mbTCPWifi.server();

      //>>>>SETTINGAN UNTUK COIL STATUS
      for (uint8_t i = 0; i <
banyak_data_coil+banyak_data_coil2; i++)
      //Setting Coil yg digunakan
      mbTCPWifi.addCoil(COIL_BASE, COIL_VAL(false),
banyak_data_coil+banyak_data_coil2);
      //Tambahkan 1 callback untuk beberapa Coil tsb.
Ini dipanggil untuk mendapatkan nilai Coilnya
```

```
    mbTCPWifi.onGetCoil(COIL_BASE, cobaRead_coil,
banyak_data_coil);
    mbTCPWifi.onSetCoil(COIL_BASE, cobaWrite_coil,
banyak_data_coil);
    //Coba ini untuk nerima data biar master bisa
nulis nilai 0/1 atau false/true
    mbTCPWifi.onGetCoil(banyak_data_coil,
cobaRead_coil2, banyak_data_coil+banyak_data_coil2);
    mbTCPWifi.onSetCoil(banyak_data_coil,
cobaWrite_coil2, banyak_data_coil+banyak_data_coil2);

    //>>>>>SETTINGAN UNTUK INPUT STATUS
//   pinMode(pin_InputStatus, INPUT_PULLUP);
//   pinMode(pin_InputStatus2, INPUT_PULLUP);
    //Tambahkan Alamat Registernya - pakai addIsts()
untuk digital input
    mbTCPWifi.addIsts(SWITCH_ISTS);
    mbTCPWifi.addIsts(SWITCH_ISTS2);
    //>>>>>COBA MBUAT SETTINGAN CALLBACK UNTUK INPUT
STATUS
    mbTCPWifi.onGetIsts(nilai_offset, proses_Ists,
banyaknya_Ists);

    //>>>>>SETTINGAN UNTUK HOLDING REGISTER
            if (!mbTCPWifi.addHreg(0, 0xF0F0,
banyak_data_HoldingRegister))
//Serial.println("Error");
            mbTCPWifi.onGetHreg(0,
cbRead_HoldingRegister, banyak_data_HoldingRegister);
```

```
                mbTCPWifi.onSetHreg(0,
cbWrite_HoldingRegister, banyak_data_HoldingRegister);

    //>>>>>SETTINGAN UNTUK INPUT REGISTER
    mbTCPWifi.addIreg(SENSOR_IREG);
    mbTCPWifi.addIreg(SENSOR_IREG2);
    mbTCPWifi.addIreg(SENSOR_IREG3);
    mbTCPWifi.addIreg(SENSOR_IREG4);
    mbTCPWifi.addIreg(SENSOR_IREG5);
        mbTCPWifi.addIreg(SENSOR_IREG6);
        mbTCPWifi.addIreg(SENSOR_IREG7);
        mbTCPWifi.addIreg(SENSOR_IREG8);
        mbTCPWifi.addIreg(SENSOR_IREG9);
        mbTCPWifi.addIreg(SENSOR_IREG10);
    mbTCPWifi.addIreg(SENSOR_IREG11);
    mbTCPWifi.addIreg(SENSOR_IREG12);
    mbTCPWifi.addIreg(SENSOR_IREG13);
    mbTCPWifi.addIreg(SENSOR_IREG14);
    mbTCPWifi.addIreg(SENSOR_IREG15);
        mbTCPWifi.addIreg(SENSOR_IREG16);
        mbTCPWifi.addIreg(SENSOR_IREG17);
    //>>>>>COBA MBUAT SETTINGAN CALLBACK UNTUK INPUT
REGISTER
        mbTCPWifi.onGetIreg(nilai_offset_Ireg,
proses_Ireg, banyaknya_Ireg);
    TimerSampling = millis(); //Ditempatkan tepat
sebelum void loop biar timernya akurat
  }

//.................................................
.....................................//
```

```
//.....................................>>>>> VOID LOOP
<<<<<.....................................//
//..................................................
.....................................//

void loop()
  {
     //>>>>>UNTUK MENJALANKAN Modbus
     mbTCPWifi.task();
     jalankan_INPUT_STATUS();
     parsing_data();
     jalankan_INPUT_REGISTER();

     //DIBAWAH INI UNTUK AMBIL DATA
     //printOledAmbilData();
  }


void printOledAmbilData()
  {
     oled.clearDisplay();
     oled.setCursor(0,0);
     oled.setTextColor(SSD1306_BLACK, SSD1306_WHITE);
     oled.println(F("   AMBIL DATA YUK   "));
     oled.setTextColor(SSD1306_WHITE);
     oled.print(F("V=")); oled.print(tegangan);
oled.print(F(" I1=")); oled.print(arusPZEM);
     oled.print(F(" PF=")); oled.println(Pf);
     oled.display();
  }
```

```
//
void jalankan_INPUT_REGISTER()
  {
     if (millis() > TimerSampling + 250)
     {
     TimerSampling = millis();
          mbTCPWifi.Ireg(SENSOR_IREG,  tegangan*100);
          mbTCPWifi.Ireg(SENSOR_IREG2,  arusPZEM*100);
          mbTCPWifi.Ireg(SENSOR_IREG3,  dayaPZEM*100);
          mbTCPWifi.Ireg(SENSOR_IREG4,
arusClamp1*100);
          mbTCPWifi.Ireg(SENSOR_IREG5,
dayaClamp1*100);
          mbTCPWifi.Ireg(SENSOR_IREG6,
arusClamp2*100);
          mbTCPWifi.Ireg(SENSOR_IREG7,
dayaClamp2*100);
          mbTCPWifi.Ireg(SENSOR_IREG8,
arusClamp3*100);
          mbTCPWifi.Ireg(SENSOR_IREG9,
dayaClamp3*100);
          mbTCPWifi.Ireg(SENSOR_IREG10,
arusClamp4*100);
          mbTCPWifi.Ireg(SENSOR_IREG11,
dayaClamp4*100);
          mbTCPWifi.Ireg(SENSOR_IREG12,
arusClamp5*100);
          mbTCPWifi.Ireg(SENSOR_IREG13,
dayaClamp5*100);
          mbTCPWifi.Ireg(SENSOR_IREG14,
arusClamp6*100);
```

```
          mbTCPWifi.Ireg(SENSOR_IREG15,
dayaClamp6*100);
          mbTCPWifi.Ireg(SENSOR_IREG16,
arusClamp7*100);
          mbTCPWifi.Ireg(SENSOR_IREG17,
dayaClamp7*100);

      }
  }

void jalankan_INPUT_STATUS()
  {
     mbTCPWifi.Ists(SWITCH_ISTS ,k);
     mbTCPWifi.Ists(SWITCH_ISTS2,l);
  }



void oled_menunggu()
  {
     oled.clearDisplay();
     oled.setTextColor(SSD1306_WHITE);
     oled.setTextSize(1);
     oled.setCursor(0,15);
     oled.println(F("--------"));
     oled.println(F("Sabar dulu..."));
     oled.println(F("masih menghubungkan"));
     oled.display();
     delay(200);
  }

#define tinggi_logo_unej 60   //+naik -turun
```

```
#define lebar_logo_unej  61
const unsigned char logo_unej[] PROGMEM = {
  0x00,0x00,0x00,0x0F,0x80,0x00,0x00,0x00,
  0x00,0x00,0x00,0x3F,0xE0,0x00,0x00,0x00,
  0x00,0x00,0x00,0xF2,0x78,0x00,0x00,0x00,
  0x00,0x00,0x03,0xDF,0x9E,0x00,0x00,0x00,
  0x00,0x00,0x0F,0x78,0xF7,0x80,0x00,0x00,
  0x00,0x00,0x1D,0xC0,0x1D,0xC0,0x00,0x00,
  0x00,0x00,0x73,0x00,0x06,0x70,0x00,0x00,
  0x00,0x00,0xEE,0x00,0x03,0xB8,0x00,0x00,
  0x00,0x03,0x98,0x33,0x80,0xEE,0x00,0x00,
  0x00,0x07,0x60,0xF3,0xCC,0x37,0x00,0x00,
  0x00,0x0C,0xC6,0xF3,0xDE,0x19,0x80,0x00,
  0x00,0x3B,0x36,0xF7,0x8E,0x66,0xE0,0x00,
  0x00,0x76,0x36,0xFA,0xDE,0xC3,0x70,0x00,
  0x00,0xEC,0x9E,0x60,0x1E,0xDD,0xB8,0x00,
  0x01,0xD8,0xCE,0x0F,0x81,0x8E,0xDC,0x00,
  0x03,0x26,0xE0,0x6F,0xF0,0x9E,0x26,0x00,
  0x06,0x47,0x62,0xFF,0xFA,0x18,0x1B,0x00,
  0x0D,0x9F,0x8F,0xF0,0x7F,0x11,0xCD,0x80,
  0x1B,0x1F,0x8F,0xF0,0x5F,0x07,0xC6,0xC0,
  0x3A,0x0E,0x3E,0x82,0x1A,0xE7,0xC3,0x60,
  0x74,0x66,0x7F,0x82,0x0F,0xE1,0x81,0x70,
  0x6C,0x78,0x3F,0x87,0x0E,0xE1,0xB0,0xB0,
  0xD8,0x18,0xD7,0x07,0x07,0xD0,0xF8,0xD8,
  0xF0,0xC9,0xFE,0x17,0x43,0xFC,0xF8,0x78,
  0xF0,0xF9,0xFE,0x1F,0xC3,0xFC,0xF0,0x78,
  0xF0,0x30,0xCC,0x0F,0xC1,0xB0,0x60,0x78,
  0xF0,0x01,0xEC,0xC7,0x41,0xBC,0x00,0x78,
  0xF0,0x03,0xFC,0xDA,0xC1,0xFC,0x00,0x78,
  0xD0,0x03,0xD8,0xDF,0xC0,0xFE,0x00,0x58,
```

```
0xD0,0x03,0x88,0xCF,0xC1,0x8C,0x00,0x58,
0xD0,0x0F,0x8C,0xDB,0xC1,0x1D,0x00,0x58,
0x78,0x0D,0xF0,0x5F,0xC0,0x7B,0x00,0xF0,
0x78,0x0E,0xF8,0x6F,0xF1,0xF3,0x00,0xF0,
0x78,0x0F,0x3C,0x66,0x7B,0xE7,0x00,0xF0,
0x68,0x0E,0xC6,0x33,0xC2,0x1B,0x80,0xB0,
0x68,0x0B,0x7A,0x3B,0x80,0xF6,0x80,0xB0,
0x3C,0x0B,0x2F,0x8F,0x8F,0xED,0x81,0xE0,
0x3C,0x0D,0xA1,0xCF,0x3C,0x6D,0x01,0xE0,
0x34,0x04,0xF0,0xE7,0x68,0x79,0x01,0x60,
0x36,0x06,0x3E,0x97,0x5B,0xE3,0x01,0x60,
0x1E,0x07,0xF7,0xCE,0x9F,0x7F,0x03,0xC0,
0x1A,0x02,0x07,0xFF,0xFE,0x06,0x03,0xC0,
0x1A,0x03,0x1C,0x1F,0xC1,0xCC,0x02,0xC0,
0x0F,0x01,0xF0,0x6F,0xF0,0x7C,0x07,0x80,
0x0D,0x00,0xC1,0xFF,0xFC,0x38,0x05,0x80,
0x0D,0x80,0x7F,0xFF,0xFF,0xE0,0x0D,0x80,
0x07,0x80,0x0F,0xFF,0xFF,0x80,0x0F,0x00,
0x06,0x80,0x0F,0xFF,0xEF,0x98,0x0B,0x00,
0x03,0xC0,0x03,0x7F,0xF7,0x3C,0x1E,0x00,
0x03,0x41,0xB8,0xFF,0xF8,0x7E,0x16,0x00,
0x03,0x61,0xF6,0x6F,0xB1,0xBE,0x36,0x00,
0x01,0xA0,0xE7,0x83,0x07,0x98,0x2C,0x00,
0x01,0xB0,0x0F,0xB0,0x77,0x88,0x6C,0x00,
0x00,0xD0,0x0F,0x7E,0xF3,0xC0,0x58,0x00,
0x00,0xF8,0x0E,0x7E,0xF3,0xC0,0xD8,0x00,
0x00,0x6C,0x02,0x7E,0xF9,0x01,0xB0,0x00,
0x00,0x37,0x00,0x5C,0xF0,0x07,0x60,0x00,
0x00,0x33,0xF8,0x00,0x00,0xFE,0x60,0x00,
0x00,0x1C,0x3F,0xFF,0xFF,0xE1,0xC0,0x00,
0x00,0x0F,0xE0,0x3F,0xF0,0x3F,0x80,0x00,
```

```
  0x00,0x01,0xFF,0xFF,0xFF,0xFC,0x00,0x00,
  0x00,0x00,0x0F,0xFF,0xFF,0x80,0x00,0x00
};

#define tinggi_logo_lab 65
#define lebar_logo_lab  70
const unsigned char logo_lab[] PROGMEM = {
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
.........................................................
.................
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
.........................................................
.................
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
.........................................................
.................
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
.........................................................
.................
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
.........................................................
.................
  0x00,0x00,0x00,0x00,0x38,0x00,0x00,0x00,0x00, //
.....................................###.................
.................
  0x00,0x00,0x00,0x00,0x38,0x00,0x00,0x00,0x00, //
.....................................###.................
.................
  0x00,0x00,0x00,0x00,0x7C,0x00,0x00,0x00,0x00, //
.....................................#####...............
.................
```

```
  0x00,0x00,0x00,0x00,0xFC,0x00,0x00,0x00,0x00, //
.............................######.................
.................
  0x00,0x00,0x00,0x00,0xFE,0x00,0x00,0x00,0x00, //
.............................#######................
.................
  0x00,0x00,0x00,0x01,0xFE,0x00,0x00,0x00,0x00, //
............................########................
.................
  0x00,0x00,0x00,0x01,0xCF,0x00,0x00,0x00,0x00, //
...............................###..####..............
.................
  0x00,0x00,0x00,0x03,0xC7,0x80,0x00,0x00,0x00, //
.............................####...####.............
.................
  0x00,0x00,0x00,0x03,0xC7,0x80,0x00,0x00,0x00, //
.............................####...####.............
.................
  0x00,0x00,0x00,0x07,0x83,0xC0,0x00,0x00,0x00, //
............................####.....####.............
.................
  0x00,0x00,0x00,0x0F,0x03,0xC0,0x00,0x00,0x00, //
...........................####......####.............
.................
  0x00,0x00,0x00,0x0F,0x01,0xE0,0x00,0x00,0x00, //
...........................####......####.............
.................
  0x00,0x00,0x00,0x1E,0x01,0xE0,0x00,0x00,0x00, //
..........................####.......####.............
.................
```

```
  0x00,0x00,0x00,0x1E,0x00,0xF0,0x00,0x00,0x00, //
.........................####.........####.........
.................
  0x00,0x00,0x00,0x3C,0x6C,0x78,0x00,0x00,0x00, //
.........................####...##.##...####.........
.................
  0x00,0x00,0x00,0x38,0x7C,0x78,0x00,0x00,0x00, //
.........................###....#####...####.........
.................
  0x00,0x00,0x00,0x78,0xFC,0x3C,0x00,0x00,0x00, //
.........................####...######....####.........
.................
  0x00,0x00,0x00,0xF0,0xF8,0x3C,0x00,0x00,0x00, //
.........................####....#####.....####.........
.................
  0x00,0x00,0x00,0xF0,0xF8,0x1E,0x00,0x00,0x00, //
.........................####....#####......####........
.................
  0x00,0x00,0x01,0xE1,0xF0,0x0E,0x00,0x00,0x00, //
.........................####....#####........###........
.................
  0x00,0x00,0x01,0xC1,0xF0,0x0F,0x00,0x00,0x00, //
.........................###.....#####........###........
.................
  0x00,0x00,0x03,0xC3,0xE0,0x07,0x80,0x00,0x00, //
.........................####....#####.........####......
.................
  0x00,0x00,0x03,0x83,0xE0,0x07,0x80,0x00,0x00, //
.........................###.....#####.........####......
.................
```

```
  0x00,0x00,0x07,0x83,0xE1,0x83,0xC0,0x00,0x00, //
.....................####.....#####....##.....####.....
.................
  0x00,0x00,0x0F,0x03,0xC3,0x81,0xC0,0x00,0x00, //
.....................####......####....###......###.....
.................
  0x00,0x00,0x0F,0x07,0xCF,0x01,0xE0,0x00,0x00, //
.....................####.....#####..####.......####....
.................
  0x00,0x00,0x1E,0x07,0xFF,0x00,0xE0,0x00,0x00, //
.....................####......##########........###....
.................
  0x00,0x00,0x1C,0x07,0xFE,0x00,0xF0,0x00,0x00, //
.....................###.......#########.........####...
.................
  0x00,0x00,0x3C,0x0F,0xFE,0x00,0x78,0x00,0x00, //
.....................####......##########.........####..
.................
  0x00,0x00,0x78,0x0F,0x1E,0x00,0x78,0x00,0x00, //
.................####.......####...####..........####..
.................
  0x00,0x00,0x78,0x1C,0x1E,0x00,0x3C,0x00,0x00, //
.................####......###.....####..........####.
.................
  0x00,0x00,0xF0,0x00,0x1C,0x00,0x1C,0x00,0x00, //
.................####............###............###.
.................
  0x00,0x00,0xE0,0x00,0x1C,0x00,0x1E,0x00,0x00, //
.................###.............###............####
.................
```

```
  0x00,0x01,0xE0,0x00,0x1C,0x00,0x0F,0x00,0x00, //
..............####..............###.............###
#..............
  0x00,0x03,0xC0,0x00,0x38,0x00,0x0F,0x00,0x00, //
..............####..............###.............###
#..............
  0x00,0x03,0xC0,0x00,0x38,0x00,0x07,0x80,0x00, //
..............####..............###.............##
##.............
  0x00,0x07,0x80,0x00,0x30,0x00,0x03,0x80,0x00, //
..............####..............##..............#
##.............
  0x00,0x07,0x80,0x00,0x78,0x00,0x03,0xC0,0x00, //
..............####............####.............#
###............
  0x00,0x0F,0x00,0x00,0xF8,0x00,0x01,0xE0,0x00, //
..............####............#####.............
####............
  0x00,0x1E,0x00,0x00,0x78,0x00,0x01,0xE0,0x00, //
...........####.................####.............
####............
  0x00,0x1E,0x00,0x00,0x70,0x00,0x00,0xF0,0x00, //
...........####.................###..............
.####...........
  0x00,0x3C,0x00,0x00,0x60,0x00,0x00,0xF0,0x00, //
..........####.................##...............
.####...........
  0x00,0x3C,0x00,0x00,0x60,0x00,0x00,0x78,0x00, //
..........####.................##...............
..####..........
```

```
  0x00,0x78,0x00,0x00,0x00,0x00,0x00,0x78,0x00, //
.........####.....................................
..####..........
  0x00,0x70,0x00,0x00,0x00,0x00,0x00,0x3C,0x00, //
.........###......................................
...####.........
  0x00,0xF0,0x00,0x00,0x00,0x00,0x00,0x1E,0x00, //
........####.....................................
....####........
  0x00,0xE6,0x66,0xEE,0x4C,0xCD,0xF3,0x9E,0x00, //
........###..##..##..##.###.###..#..##..##..##.#####..#
##..####........
  0x01,0xC4,0x64,0x4C,0x44,0xCC,0x23,0x8F,0x00, //
.......###...#...##..#...#..##...#...#..##..##....#...#
##...####.......
  0x03,0xC4,0x24,0x04,0x04,0x08,0x01,0x8F,0x80, //
......####...#....#..#.......#.......#......#.........
##...#####......
  0x03,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x80, //
......#################################################
##########.......
  0x07,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x80, //
.....#################################################
##########.......
  0x06,0x34,0x66,0x6C,0xF1,0xB7,0xCC,0x63,0xC0, //
.....##...##.#...##..##..##.##..####...##.##.#####..##.
..##...####......
  0x07,0x30,0x62,0x40,0xF9,0x10,0x89,0x63,0x80, //
.....###..##.....##...#..#......#####..#...#....#...#..
#.##...###.......
```

```
  0x07,0x00,0x26,0x00,0xF9,0x00,0x80,0x01,0x80, //
.....###.........#..##.........#####..#........#.....
........##.......
  0x07,0x32,0x66,0x4C,0xF9,0x94,0x8C,0x63,0x80, //
.....###..##..#..##..##..#..##..#####..##..#.#..#...##.
..##...###.......
  0x07,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x80, //
.....#########################################################
##########.......
  0x03,0xFF,0xFF,0xFF,0x8E,0xFF,0xFF,0xFE,0x00, //
......############################...###.###############
########.........
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
.........................................................
.................
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00  //
.........................................................
.................
};

void gambar_loading()
  {
     int16_t i;
     oled.clearDisplay();
     for(i=0; i<oled.width(); i+=4)
     {
     oled.drawLine(0, 0, i, oled.height()-1,
SSD1306_WHITE);
     oled.display(); // Update screen with each
newly-drawn line
     delay(1);
```

```
      }
      for(i=0; i<oled.height(); i+=4)
      {
      oled.drawLine(0, 0, oled.width()-1, i,
SSD1306_WHITE);
      oled.display();
      delay(1);
      }
      delay(200);
  }

void tampilkan_logo_unej(void)
  {
      oled.drawBitmap(
      (oled.width()  - lebar_logo_unej) / 2,
      (oled.height() - tinggi_logo_unej) / 2,
      logo_unej, lebar_logo_unej, tinggi_logo_unej, 1);
      oled.display();
      delay(300);
  }

void tampilkan_logo_lab(void)
  {
      oled.drawBitmap(
      (oled.width()  - lebar_logo_lab) / 2,
      (oled.height() - tinggi_logo_lab) / 2,
      logo_lab, lebar_logo_lab, tinggi_logo_lab, 1);
      oled.display();
      delay(400);
      //prosesterhubung=true;
  }
```

```
void parsing_data()
  {
     if (Serial.available())
     {
     if(Serial.find('^'))
          {
          delay(15);
          tegangan=Serial.parseFloat();
          arusPZEM=Serial.parseFloat();
          dayaPZEM=Serial.parseFloat();
               arusClamp1=Serial.parseFloat();
               dayaClamp1=Serial.parseFloat();
          arusClamp2=Serial.parseFloat();
          dayaClamp2=Serial.parseFloat();
               arusClamp3=Serial.parseFloat();
               dayaClamp3=Serial.parseFloat();
          arusClamp4=Serial.parseFloat();
          dayaClamp4=Serial.parseFloat();
               arusClamp5=Serial.parseFloat();
               dayaClamp5=Serial.parseFloat();
          arusClamp6=Serial.parseFloat();
          dayaClamp6=Serial.parseFloat();
               arusClamp7=Serial.parseFloat();
               dayaClamp7=Serial.parseFloat();
          k=Serial.parseFloat();
          l=Serial.parseFloat();
          simpanrelay=Serial.parseInt();
          //Pf=Serial.parseFloat();
```

```
            delay(45);
            }
    }
    //PROGRAM statusWifi '&'
}
```

Lampiran 3.3 Perancangan File Header Sensor ZMPT-004T

```
#ifndef NABIL_PZEM_H
#define NABIL_PZEM_H

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

// #define PZEM004_NO_SWSERIAL
#if (not defined(PZEM004_NO_SWSERIAL)) &&
(defined(__AVR__) || defined(ESP8266))
#define PZEM004_SOFTSERIAL
#endif

//#if defined(PZEM004_SOFTSERIAL)
//#include <SoftwareSerial.h>
//#include <NeoSWSerial.h>
//#include <AltSoftSerial.h>
//#endif

#define PZEM_DEFAULT_ADDR     0xF8


class Nabil_PZEM
{
public:
     //Nabil_PZEM(uint8_t receivePin, uint8_t
transmitPin, uint8_t addr=PZEM_DEFAULT_ADDR);
```

```
      Nabil_PZEM(HardwareSerial* port, uint8_t
addr=PZEM_DEFAULT_ADDR);
      ~Nabil_PZEM();

      float voltage();
      float current();
      float power();
      float pf();

      //bool setAddress(uint8_t addr);
      //uint8_t getAddress();

private:

      Stream* _serial; // Serial interface
      bool _isSoft;  // Is serial interface software
      uint8_t _addr;   // Device address

      struct {
      float voltage;
      float current;
      float power;
      float pf;
      }  _currentValues;

      uint64_t _lastRead;

      void init(uint8_t addr);

      bool updateValues();
```

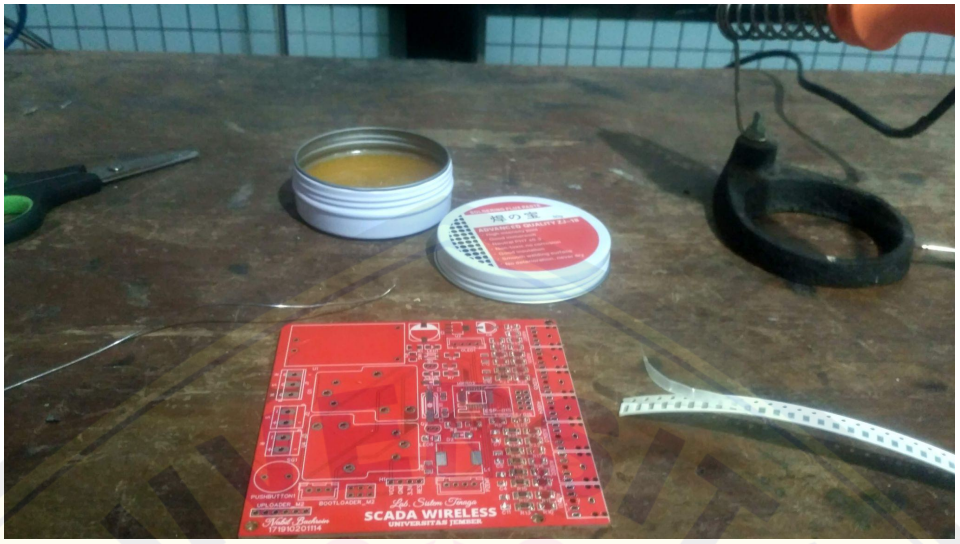
```
    uint16_t recieve(uint8_t *resp, uint16_t len); //
Receive len bytes into a buffer

    bool sendCmd8(uint8_t cmd, uint16_t rAddr,
uint16_t val, bool check=false); // Send 8 byte command

    void setCRC(uint8_t *buf, uint16_t len);
// Set the CRC for a buffer
    bool checkCRC(const uint8_t *buf, uint16_t len);
    uint16_t CRC16(const uint8_t *data, uint16_t len);
};

#endif
```

Lampiran 3.4 Perancangan File Header Sensor YHDC SCT-013

```
#ifndef Nabil_SCT_h
#define Nabil_SCT_h

#if defined(ARDUINO) && ARDUINO >= 100

#include "Arduino.h"
#else
#include "WProgram.h"

#endif
#ifndef READVCC_CALIBRATION_CONST
#define READVCC_CALIBRATION_CONST 1126400L
#endif

#if defined(__arm__)
#define ADC_BITS    12
```

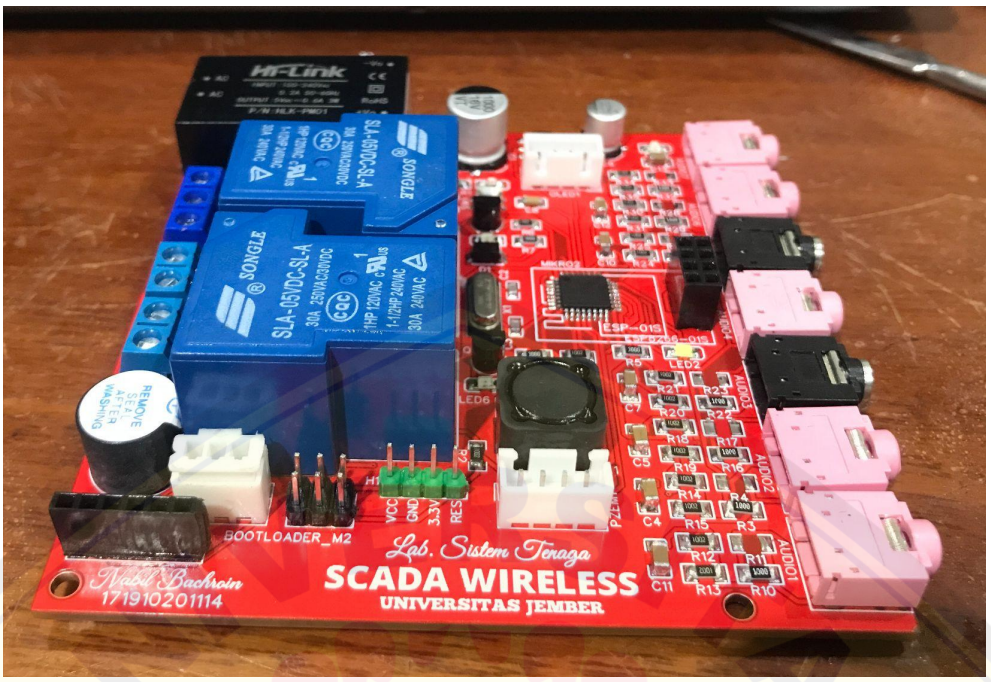
```
#else
#define ADC_BITS    10
#endif

#define ADC_COUNTS  (1<<ADC_BITS)
class ArusLibNabil
{
  public:
     double calcIrms(unsigned int NUMBER_OF_SAMPLES);
     long readVcc();
     double Irms;

  private:
     unsigned int inPinI;
     double ICAL;
     int sampleI;
     double filteredI;
     double offsetI;                          //Low-pass
filter output

     double phaseShiftedV;
     double sqI,sumI;
     boolean lastVCross, checkVCross;
};

#endif
```

Lampiran Dokumentasi

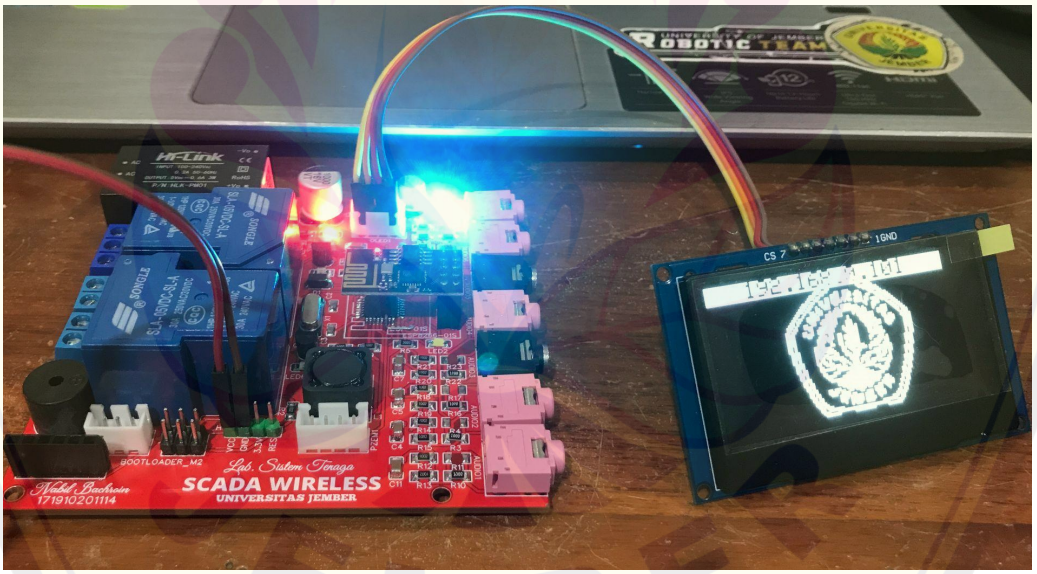


Gambar 1. RTU Sebelum Terpasang Komponen

Gambar 2. Proses Pemasangan Komponen



Gambar 3. Tampilan Hasil Pemasangan Komponen Pada RTU



Gambar 4. Proses Pengujian RTU

Gambar 5. Tampilan Akhir RTU