BAB III

METODE PENELITIAN

Penelitian ini menggunakan pendekatan kuantitatif dengan metode penelitian eksperimental atau uji coba. Tujuan dari penelitian ini yaitu membuat sistem kontrol utilitas gedung dari sisi perangkat keras dan perangkat lunak yang dapat diterapkan pada aplikasi gedung bertingkat dimana di setiap lantai gedung dioperasikan oleh beberapa unit kontroler yang terhubung dengan HMI pada komputer yang dioperasikan oleh pengelola gedung melalui jaringan *ethernet*.

Penelitian berupa eksperimen pada perancangan sistem, perancangan dan pembuatan unit kontroler, penulisan program pada MCU dan aplikasi pada HMI. Unit kontroler pada sistem dapat berjumlah lebih dari satu, tetapi penulis hanya akan membangun satu buah unit kontroler.

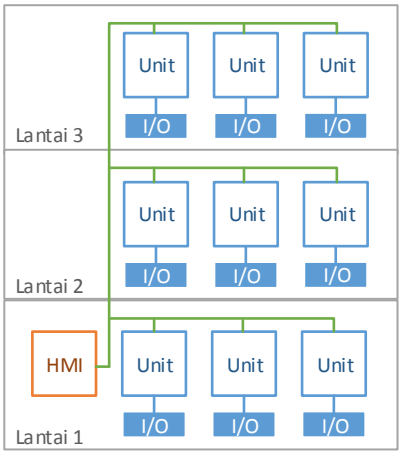
3.1 Spesifikasi Sistem

Perancangan sistem kontrol dan automasi gedung dimulai dari menentukan spesifikasi sistem. Spesifikasi ini akan menjadi acuan dalam perancangan, pembuatan dan pengembangan sistem hingga akhir penelitian. Sistem kontrol terdiri atas satu atau lebih unit kontroler dan satu komputer sebagai kontrol pusat. Hal pertama yang penulis tentukan adalah karakteristik unit kontroler.

a. Unit kontroler mengoperasikan relay yang terhubung dengan penerangan atau peralatan listrik berdaya rendah pada suatu area yang terdri atas satu atau beberapa ruang, sehingga dalam satu lantai dapat dipasang beberapa unit.
b. Unit kontroler berkomunikasi dengan HMI melalui jaringan *ethernet* dengan protokol modbusTCP.
c. Unit kontroler hanya dapat dikontrol dengan aplikasi HMI yang dirancang pada penelitian ini dengan lebih dari satu HMI dapat terhubung dalam sistem di saat bersamaan.

d. Unit kontrol masih dapat bekerja walaupun tanpa terhubung dengan HMI karena seluruh konfigurasi unit kontroler disimpan dalam EEPROM.

e. Unit kontroler menggunakan alamat IP statis yang ditentukan dari data di EEPROM.



Gambar 3.1 Letak unit kontroler dalam gedung.

Setelah itu adalah penentuan jenis dan jumlah I/O pada unit kontroler:

Tabel 3.1 Jenis I/O pada unit kontroler.

No.	Tipe I/O	Jumlah
1	Keluaran Diskrit	8
2	Masukan Diskrit	6
3	Masukan Analog	6

Setelah itu adalah penentuan algoritma program mikrokontroler dan HMI.

a. Program MCU berbasis *interrupt*, sehingga MCU tidak menggunakan *polling* untuk mengecek setiap I/O secara berkala.

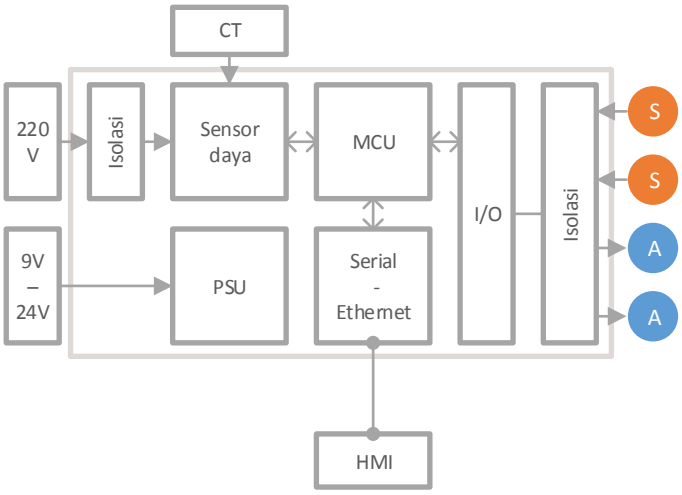
b. MCU memiliki RTC (*real time clock*) yang dapat diselaraskan dengan waktu di komputer pengguna.

c. MCU memiliki 16 jadwal yang mengatur keluaran diskrit dan dapat dinyalakan atau dimatikan secara manual melalui HMI.

d. Setiap masukan (diskrit dan analog) memiliki dua *link* terpisah yang masing-masing dapat diatur agar tehubung dengan satu keluaran atau satu grup keluaran untuk mengontrol keluaran tersebut secara otomatis.

e. *Link* pada masukan dapat menghormati jadwal yang dikenakan pada keluaran diskrit tujuan, sehingga masukan hanya mempengaruhi keluaran saat berada dalam jadwal yang aktif.
f. Setiap masukan analog memiliki nilai pengali (*multiplier*), *offset*, batas atas dan batas bawah yang dapat diatur agar dapat menampilkan nilai akhir yang diinginkan.
g. Masukan analog akan mengaktifkan *link* yang pertama saat melebihi batas atas dan mengaktifkan *link* yang kedua saat kurang dari batas bawah.
h. Keluaran diskrit dapat dimasukkan ke jadwal manapun yang pengguna kehendaki dengan nomor jadwal lebih besar menjadi hasil keluaran akhir jika terjadi hasil yang berbeda diantara jadwal tersebut.
i. Kontrol manual untuk keluaran diskrit hanya dapat dilakukan melalui HMI.
j. HMI melakukan *polling* data pada seluruh I/O setiap 500ms.
k. HMI dapat mengubah konfigurasi setiap I/O dan jadwal pada kontroler.
l. HMI dirancang agar dapat tersambung dengan seluruh unit kontoler dalam gedung secara bersamaan.

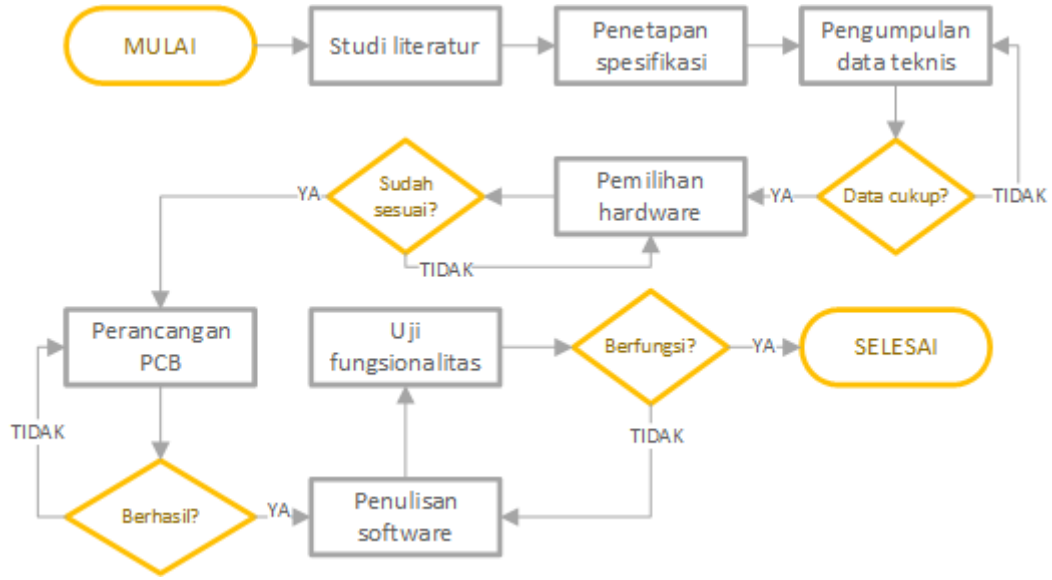
Perangkat keras unit kontroler yang dirancang pada penelitian ini terdiri dari atas regulator tegangan, mikrokontroler, *ethernet controller* dan sensor daya. Relay dan trafo arus terpisah dengan unit kontroler agar dapat disesuaikan dengan beban yang terhubung. Adapun diagram blok unit kontroler ditunjukkan oleh gambar 3.2.



Gambar 3.2 Diagram blok unit kontroler.

3.2 Prosedur Penelitian

Prosedur yang dilakukan dalam penelitian ini terdiri atas beberapa langkah. Pertama, studi literatur terkait perancangan automasi bangunan secara umum, yaitu perangkat keras dan aplikasi berbasis mikrokontroler yang dibutuhkan. Kemudian dilanjutkan dengan penetapan spesifikasi sistem sebagai acuan. Setelah itu mengumpulkan data teknis dan pemilihan perangkat keras dan aplikasi yang akan digunakan, baik dari sistem yang sudah ada, *datasheet* perangkat terkait dan rujukan lainnya. Setelah itu dilakukan perancangan PCB dan pemasangan komponen. Setelah itu dilakukan penulisan program untuk mikrokontroler dan aplikasi HMI. Selanjutnya adalah pengujian fitur yang terpasang pada kontroler, yaitu mendeteksi input dari sensor, menyalakan relay, penjadwalan, pengaturan *setpoint* dan komunikasi dengan HMI. Adapun diagram alir penelitian ditunjukkan pada gambar 3.3.



Gambar 3.3 Diagram alir penelitian.

3.3 Perancangan Unit Kontroler

3.3.1 *Microcontroller Unit* (MCU)

MCU merupakan inti dari alat yang akan mengontrol berbagai sistem utilitas seperti pencahayaan, AC, *plumbing* dan lain sebagainya. Unit MCU tersedia dari

berbagai *manufacturer* yang memiliki berbagai fitur dari MCU yang bersifat umum, performa tinggi hingga yang terfokus pada penghematan daya. Pemilihan MCU sangat krusial karena akan menentukan kinerja alat, ekosistem *programming* yang tersedia dan *peripheral* yang dapat dipasang. kriteria umum dalam pemilihan MCU diantaranya adalah keseimbangan antara fitur dan harga dan mudah untuk diprogram.

Terdapat berbagai tipe MCU yang dapat digunakan, seperti ATMEGA, ATTiny, PIC, ESPLORA, RaspberryPi dan STM32. Penulis memilih STM32F103C8T6, yang berbasis ARM Cortex-M3 karena memiliki jumlah GPIO (*general purpose input* output) yang cukup dengan harga yang lebih murah. MCU ini memiliki berbagai fitur, namun fitur yang penulis perhatikan adalah sebagai berikut:

a. Frekuensi *clock* CPU hingga 72 MHz,
b. PLL (*phase locked loop*) untuk *clock* CPU,
c. NVIC (*Nested vectored interrupt controller*),
d. 128 KBytes flash memory,
e. 20 KBytes SRAM (*static random access memory*),
f. 37 GPIO (*general purpose input output*),
g. RTC (*real-time clock*) (STMicroelectronics, 2015).

Sistem *clock* dalam STM32 memiliki berbagai fitur dimana masing-masing fungsi *peripheral* dapat memili *clock* yang berbeda. Selain itu dengan adanya PLL dan *prescaler* memungkinkan MCU untuk bekerja dengan frekuensi yang lebih tinggi dibandingkan frekuensi osilator (STMicroelectronics, 2015).

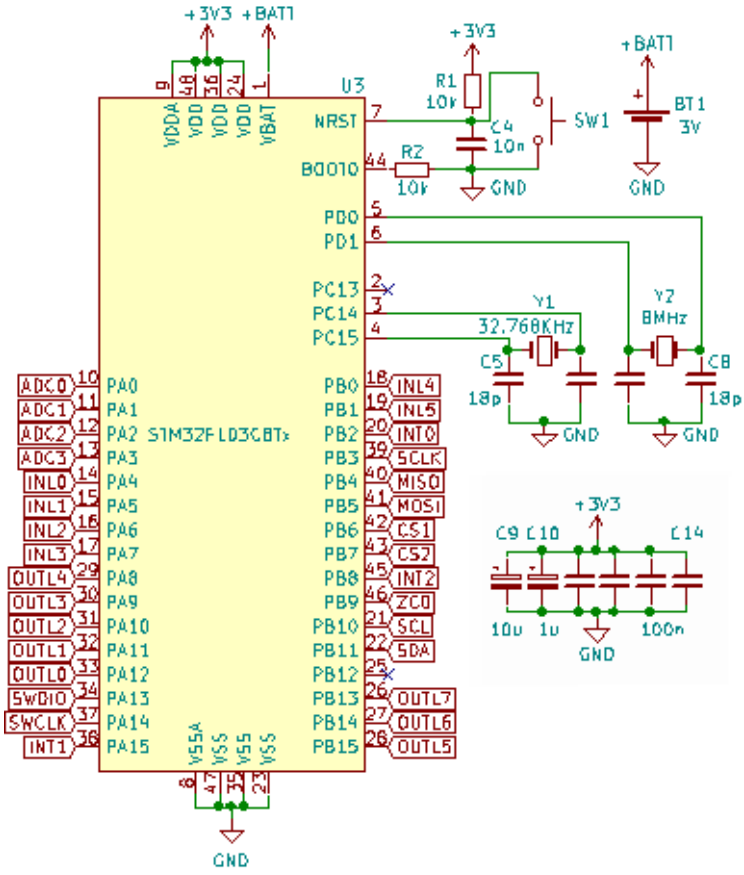
NVIC memiliki 43 jalur *interrupt* yang dapat diaktifkan sesuai kebutuhan dengan 16 tingkat prioritas yang dapat diatur. Jalur *interrupt* ini dapat bersumber dari GPIO, RTC dan jalur *peripheral* seperti ADC, SPI dan I2C. Tingkat prioritas dapat diatur sedemikian rupa sehingga *interrupt* yang memiliki prioritas lebih tinggi dapat diproses terlebih dahulu dan menunda *interrupt* yang prioritasnya lebih rendah, sifat ini dinamakan *preemption*. Sedangkan di saat terdapat dua *interrupt* dengan tingkat prioritas yang sama, *interrupt* yang diproses adalah yang pertama

muncul dengan dilanjutkan *interrupt* yang tertunda, sifat ini dinamakan *tail-chaining*. Dengan memanfaatkan NVIC, penulis dapat mengembangkan aplikasi berbasis *interrupt*, dimana MCU tidak perlu melakukan *polling* dimana MCU harus mengecek setiap *input* apakah ada perubahan atau tidak dan hanya bekerja saat *interrupt* terjadi.

STM32F103C8T6 adalah MCU yang memiliki 37 pin GPIO (*general purpose input output*), tetapi tidak semuanya akan digunakan. Berikut skema *pinout* yang akan digunakan:

a. 2 pin untuk kristal osilator utama dengan frekuensi 8 MHz,
b. 2 pin untuk kirstal osilator RTC dengan frekuensi 32,768 KHz,
c. 6 pin sebagai masukan diskrit dan terhubung sebagai *interrupt* melalui EXTI (*external interrupt / event controller*)
d. 4 pin sebagai masukan analog yang terhubung melalui *attenuator*,
e. 8 pin sebagai keluaran diskrit,
f. 8 pin untuk komunikasi SPI yang terhubung dengan sensor daya dan *ethernet controller*; dimana 3 pin digunakan untuk *clock* dan data, 3 pin sebagai *interrupt* dan 2 pin sebagai pin untuk memilih IC tujuan komunikasi,
g. 3 pin untuk komunikasi I2C dimana 2 pin sebagai *clock* dan data dan 1 pin sebagai *interrupt*,
h. 2 pin sebagai SWD (*serial wire debug*), sebagai *interface* untuk memprogram dan *debugging* CPU.

Dari 37 GPIO yang tersedia pada STM32F103C8T6, 35 terpakai dan 2 tersisa. Skema *pinout* pada MCU ditunjukkan pada gambar berikut.



Gambar 3.4 Skema *pinout* pada MCU.

3.3.2 Keluaran Diskrit

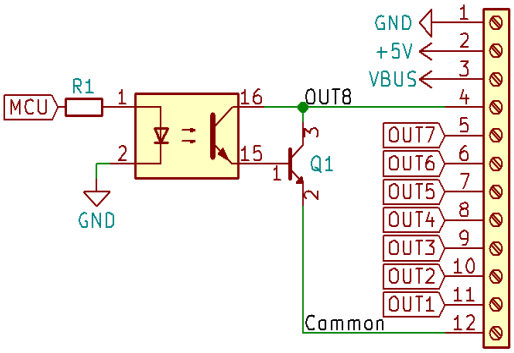
Tabel 3.2 Daftar keluaran diskrit.

No.	I/O	Tipe	Beban Terhubung
1	I/O0	Keluaran relay	Lampu
2	I/O1	Keluaran relay	Lampu
3	I/O2	Keluaran relay	Lampu
4	I/O3	Keluaran relay	Lampu
5	I/O4	Keluaran relay	Lampu
6	I/O5	Keluaran relay	Lampu
7	I/O6	Keluaran relay	Lampu
8	I/O7	Keluaran relay	Lampu

Terdapat delapan pin keluaran diskrit yang dipisahkan dengan menggunakan *optocoupler* Everlight ELQ3H7 dan transistor untuk menghasilkan keluaran *oper-collector* yang ekuivalen dengan *normally-open switch*. Tipe keluaran ini dipilih

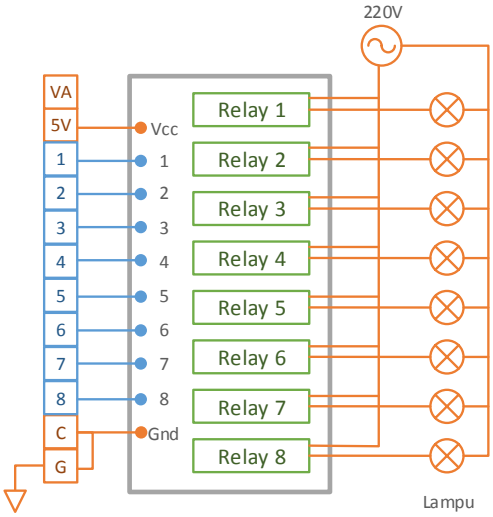
daripada tipe *push-pull* yang menghasilkan tegangan *high* atau *low* karena tegangan pada beban yang terhubung dengan terminal belum tentu sama.

Berdasarkan *datasheet* ELQ3H7, arus rekomendasi pada sisi dioda adalah 5mA. Untuk membatasi arus menjadi 5mA, besar nilai R_{40} adalah V_{R40} / I_F. Dengan V_{R1} = 3,3V–V_F = 3,3V–1,2V = 2,1V dan I_F = 5mA, maka nilai R_1 = 420Ω. Namun nilai resistor sebesar 420Ω tidak tersedia, sehingga nilai yang digunakan adalah 430Ω. Sisi transistor dihubungkan dengan transistor 3904 yang memiliki kapasitas arus lebih besar yaitu 200mA dibandingkan ELQ3H7 yang hanya 50mA.



Gambar 3.5 Rangkaian *optocoupler* pada pin keluaran MCU.

Terminal pada keluaran diskrit akan dihubungkan dengan modul relay 8 kanal dengan tegangan 5V dan kapasitas arus 10A. Modul relay mendapat sumber tegangan dari keluaran PSU. Beban terpasang melalui terminal *normally-open* pada modul relay. Skema *wiring* keluaran diskrit dapat dilihat pada gambar 3.6.



Gambar 3.6 Skema *wiring* keluaran diskrit.

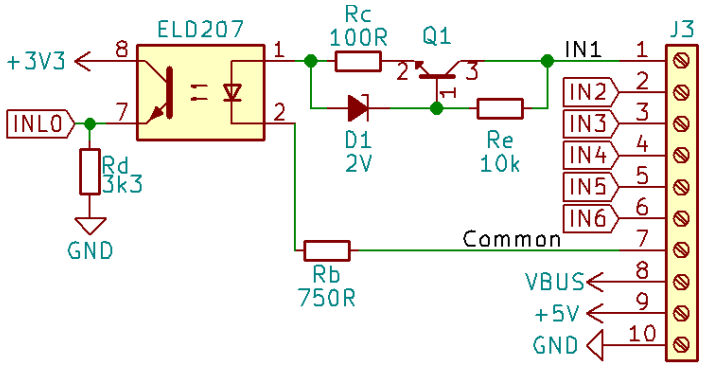
3.3.3 Masukan Diskrit

Tabel 3.3 Daftar masukan diskrit.

No.	I/O	Tipe	Sensor Terhubung
1	I/O8	Masukan diskrit	Sensor PIR
2	I/O9	Masukan diskrit	Saklar
3	I/O10	Masukan diskrit	-
4	I/O11	Masukan diskrit	-
5	I/O12	Masukan diskrit	-
6	I/O13	Masukan diskrit	-

STM32 adalah MCU yang beroperasi pada tegangan 3,3V dengan sebagian besar GPIO-nya dapat menerima tegangan input hingga 5V. Namun demikian proteksi perlu dipasang pada pin pin tersebut, untuk menghindari kerusakan MCU karena tegangan maupun arus berlebih. Proteksi yang penulis gunakan adalah dengan *optocoupler*.

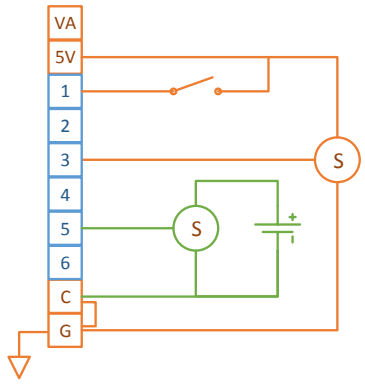
Terdapt enam pin masukan diskrit yang dipisahakan oleh *optocoupler* Everlight ELD207 dengan terminal masukan dengan rangkaian sesuai pada gambar 3.7. Pada sisi dioda, dipasang rangkaian tambahan untuk meregulasi arus. Prinsip kerjanya adalah transistor Q_1 akan mempertahankan tegangan pada D_1 akan sama dengan tegangan $R_C - V_{BE\ (Sat.)}$. Dengan $D_1 = 2V$ dan $V_{BE\ (Sat.)} = 1V$, maka tegangan pada $R_C = 1V$ dengan arus $I_F = 1V / 100\Omega = 10mA$. Dengan skema rangkaian ini, arus dioda I_F akan terjaga pada nilai 10mA selama tegangan pada terminal input lebih dari tegangan drop pada *optocoupler* dan transistor.



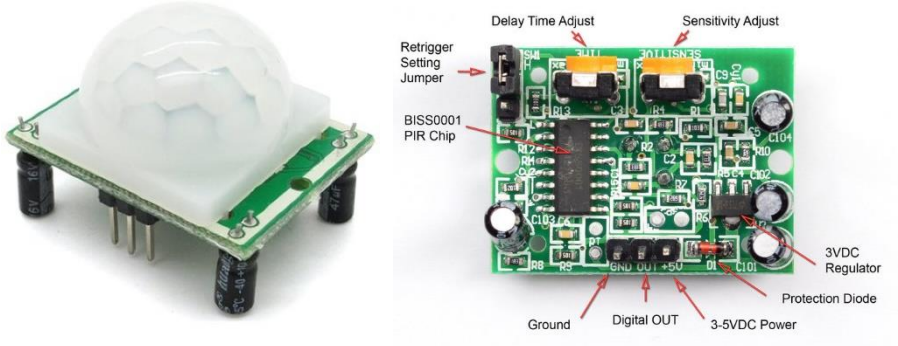
Gambar 3.7 Rangkaian *optocoupler* pada terminal masukan.

Berdasarkan *datasheet*, CTR saat I_F = 10mA berkisar 100% hingga 200%. Dengan demikian I_C akan berkisar 10mA hingga 20mA. Dengan arus 10mA, nilai R_D agar dapat menghasilkan tegangan 3,3V adalah 330Ω. Tetapi penulis memasang nilai R_D sebesar 3,3kΩ untuk mengurangi arus kolektor menjadi 1mA.

Terminal pada masukan diskrit dapat terhubung dengan saklar / tombol atau sensor dengan keluaran tegangan hingga 24V. Jika sensor beroperasi dengan sumber tegangan eksternal, maka titik 0V pada sensor perlu dihubungkan dengan *common* pada terminal masukan. Jika sensor beroperasi dengan sumber tegangan dari PSU, maka *common* harus dihubunkan dengan titik 0V PSU. Pada penelitian ini, terminal masukan dihubungkan dengan sensor PIR yang memiliki keluran tegangan sensor sebesar 3,3V sebagai *trigger* otomatis dan saklar *normally-open* sebagai *manual override*. Sensor PIR mendapat sumber tegangan 5V dari keluaran PSU, sedangkan saklar akan terhubung langsung dengan terminal 5V dan terminal masukan diskit seperti gambar berikut.



Gambar 3.8 Skema *wiring* pada terminal masukan diskrit.



Gambar 3.9 Sensor PIR.
(Sumber: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview)

3.3.4 Sensor Daya

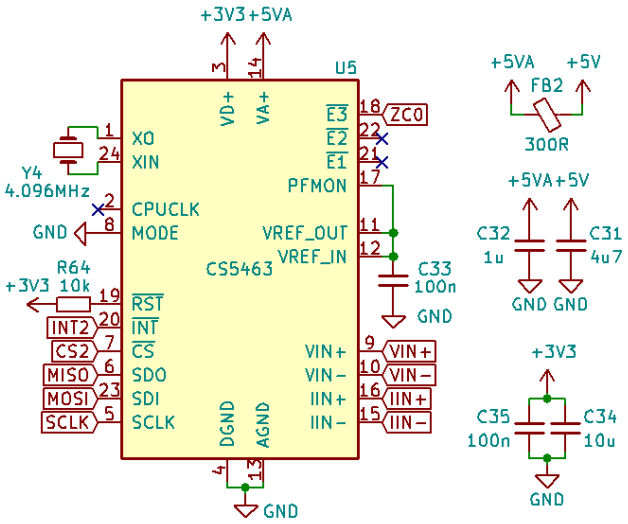
Tabel 3.4 Daftar hasil konversi sensor daya.

No.	I/O	Nama	Tipe	Sensor Terhubung
1	I/O18	V_{RMS}	Masukan analog	Jala-jala PLN
2	I/O19	I_{RMS}	Masukan analog	Trafo arus
3	I/O20	Frekuensi	Masukan analog	-
4	I/O21	Daya aktif (P)	Masukan analog	-
5	I/O22	Daya reaktif (Q)	Masukan analog	-
6	I/O23	Daya semu (S)	Masukan analog	-
7	I/O24	Faktor daya	Masukan analog	-
8	I/O25	*Current lagging*	Masukan diskrit	-

Sensor daya digunakan untuk mengawasi penggunaan daya pada beban yang terhubung dengan relay. Dengan sensor ini, pengelola gedung dapat apakah ada penggunaan daya yang abnormal pada suatu area dan dapat ditangani segera; atau aplikasi yang lebih sederhana dimana di saat faktor daya kurang dari batas tertentu, pengelola gedung dapat menyalakan *capacitor bank* untuk memperbaikinya. Terdapat banyak tipe dan kelas sensor daya yang beredar di pasaran, mulai dari sensor yang hanya dapat mengukur arus hingga sensor yang dapat melihat penggunaan daya pada beban tiga fasa dan harmonisa pada tegangan dan arus. Sensor daya yang terdapat di pasaran diantaranya adalah ACS712, HLW8012, CS5460A dan CS5463. Sensor daya yang penulis gunakan dalam penelitian ini adalah Cirrus Logic CS5463. Cirrus Logic CS5463 memiliki dua unit 24-bit ADC yang dapat mengukur tegangan dan arus bersamaan dan menghitung tegangan sesaat, arus sesaat, V_{rms}, I_{rms}, daya sesaat, daya nyata, daya semu, energi dan arah energi untuk fasa tunggal. IC ini dipilih untuk mempermudah perhitungan daya jika dibandingkan dengan menggunakan ADC pada MCU dan menghitung tegangan, arus, daya dan energi secara manual. Komunukasi dengan MCU dilakukan dengan SPI (*serial peripheral interface*).

SPI terdiri dari empat jalur, yaitu CLK (*clock*), MISO (*master in, slave out*), MOSI (*master out, slave in*) dan CS (*chip select*). Transmisi data pada SPI bersifat sinkron, dimana data bergerak bersamaan dengan perubahan *clock* pada CLK yang

disuplai oleh master atau MCU. MISO dan MOSI adalah dua jalur data dari dan menuju master atau MCU. Dengan dua jalur data maka komunikasi dua arah dapat terjadi bersamaan (*full duplex*). Ketiga jalur ini dapat disambungkan paralel dengan *slave* lain dengan jalur CS terpisah digunakan untuk memilih IC tujuan yang bersifat *active low*.



Gambar 3.10 Skema *pinout* CS5463.

Pada CS5463 tegangan dan arus akan masuk ke *sigma-delta modulator* yang akan mengukur kedua sinyal dengan frekuensi sampel:

$$F_S = \frac{\frac{F_{OSC}}{K}}{8} \tag{3.1}$$

$$F_s = \frac{4096000/_1}{8} = 512000\ sample\ /\ second \tag{3.2}$$

Kemudian mealui digital filter dengan hasil keluaran filter:

$$F_D = \frac{\frac{F_{OSC}}{K}}{1024} \tag{3.3}$$

$$F_D = \frac{4096000/_1}{1024} = 4000\ value\ /\ second \tag{3.4}$$

Keterangan:

F_S : Frekuensi *sampling* (*sample* / *second*)

F_{OSC} : Frekuensi osilator (4,096MHz)

K : *Prescaler* (*Default* = 1)

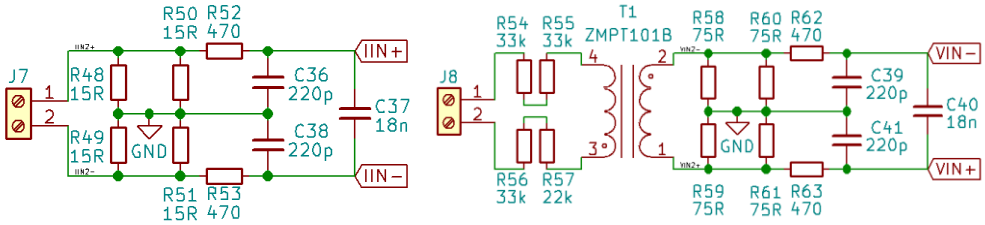
F_D : Frekuensi keluaran *filter* (*value* / *second*)

Data dari filter ini kemudian masuk ke *high-pass filter* untuk menghilangkan komponen DC yang terdapat pada sinyal. Kemudian data disesuaikan untuk menghilangkan *offset* dan rasio *gain* dan menghasilkan tegangan, arus dan daya sesaat. V_{rms} dan I_{rms} dihitung dari kumupulan tegangan / arus sebanyak N. Daya semu dihitung dengan mengalikan V_{rms} dan I_{rms}. Dengan nilai N = 4000, maka V_{rms}, I_{rms} dan energi dapat dihitung setiap 1 detik.

Tegangan 220V tidak dapat langsung dihubungkan dengan CS5463 yang hanya dapat menerima tegangan ±250mV, sehingga dibutuhkan transformer untuk menurunkan tegangan. Transformer yang digunakan adalah ZMPT101B yang pada dasarnya adalah trafo arus dengan perbandingan kumparan 1:1, sehingga membutuhkan resistor pada sisi primer untuk membatasi arus dan resistor di sisi sekunder sebagai resistor beban. Tegangan pada resistor sisi sekunder ini yang akan terhubung dengan CS5463.

ZMPT101B memiliki kumparan dengan rating 2mA, sehingga resistor yang digunakan harus dipilih agar tidak melebihi nilai ini. Toleransi tegangan yang penulis tentukan adalah 220V + 10% = 242V, sehingga dibutuhkan resistor sebesar 121kΩ. Nilai ini dibagi menjadi empat resistor, yaitu 33kΩ, 33kΩ, 33kΩ dan 22kΩ. Pada sisi sekunder akan dipasang resistor dengan besar 75Ω, sehingga di saat tegangan di sisi primer 242V_{rms}, tegangan di sisi sekunder sebesar 150mV_{rms} atau 212,132mV_{peak} dengan asumsi tegangan sinusoidal murni dengan frekuensi 50Hz.

Untuk mengukur arus akan digunakan trafo arus dengan rasio 1:2500 atau 5A:2mA dengan kapasitas arus hingga 60A dan resistor beban sebesar 10Ω, sehingga saat arus di sisi primer sebesar 37,5A, arus di sisi sekunder sebesar 10mA dan tegangan sebesar 150mV_{rms} atau 212,132mV_{peak} dengan asumsi tegangan sinusoidal murni dengan frekuensi 50Hz.

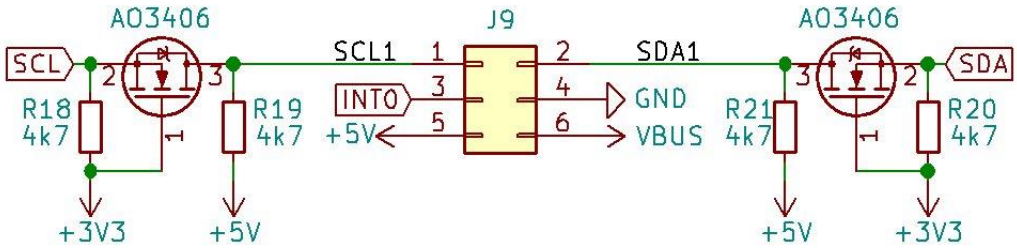


Gambar 3.11 Skema *wiring* CS5463 pada jalur arus(a) dan tegangan(b).

3.3.5 Ekspansi I/O

Terdapat total delapan keluaran diskrit, enam masukan diskrit dan enam masukan analog yang terpasang pada PCB, tetapi pada pengembangan sistem selanjutnya, jumlah tersebut mungkin tidak cukup, sehingga penulis menambahkan potensial ekspansi berbasis I2C. Terdapat banyak I/O *expander* yang menggunakan I2C, diantaranya PCF8574 untuk I/O diskrit atau ADS1115 untuk masukan analog.

I2C terdiri dari dua jalur, yaitu SCL (*clock*) dan SDA (data). I2C hanya memiliki satu jalur data, sehingga komunikasi data hanya bisa terjadi searah di saat bersamaan (*half-duplex*) dengan frekuensi 100KHz. I2C membutuhkan *pull-up* resistor, karena komponen yang terhubung bersifat *open drain* atau *open collector*. Besar resistor yang dibutuhkan bergantung pada frekuensi I2C dan total kapasitansi pada jaringan. Penulis memilih ukuran resistor yang umum yaitu 4,7KΩ. Karena STM32 beroperasi pada tegangan 3,3V sedangkan mayoritas *slave* I2C beroperasi pada tegangan 5V, maka ditambahkan *logic converter* dua arah menggunakan N-MOSFET. Skema rangkaian *convereter* I2C dapat dilihat pada gambar berkut.



Gambar 3.12 *Logic converter* dua arah untuk I2C.

3.3.6 EEPROM

EEPROM (*electrically erasable programmable read-only memory*) adalah salah satu tipe memori yang bersifat *non-volatile* dimana data yang tersimpan dalam memorinya tidak hilang saat *power supply unit* terputus. EEPROM pada penelitian ini bersifat eksternal karena MCU yang digunakan tidak memiliki EEPROM internal. IC yang digunakan adalah 24C32 yang menggunakan protokol I2C dengan ukuran memori sebesar 4KB. 24C32 digunakan untuk menyimpan data berikut:

a. Jumlah I/O (1 byte) pada alamat 0,

b. Jumlah grup I/O (1 byte) pada alamat 1,

c. Data kalibrasi CS5463 (16 byte) pada alamat 6,
d. Alamat MAC (*media access control*) (6 byte) pada alamat 22,
e. Alamat IP (*internet protocol*) (4 byte) pada alamat 28,
f. Data jadwal (80 byte) pada alamat 32,
g. Data I/O (masing-masing sebesar 56 byte) pada alamat 128.

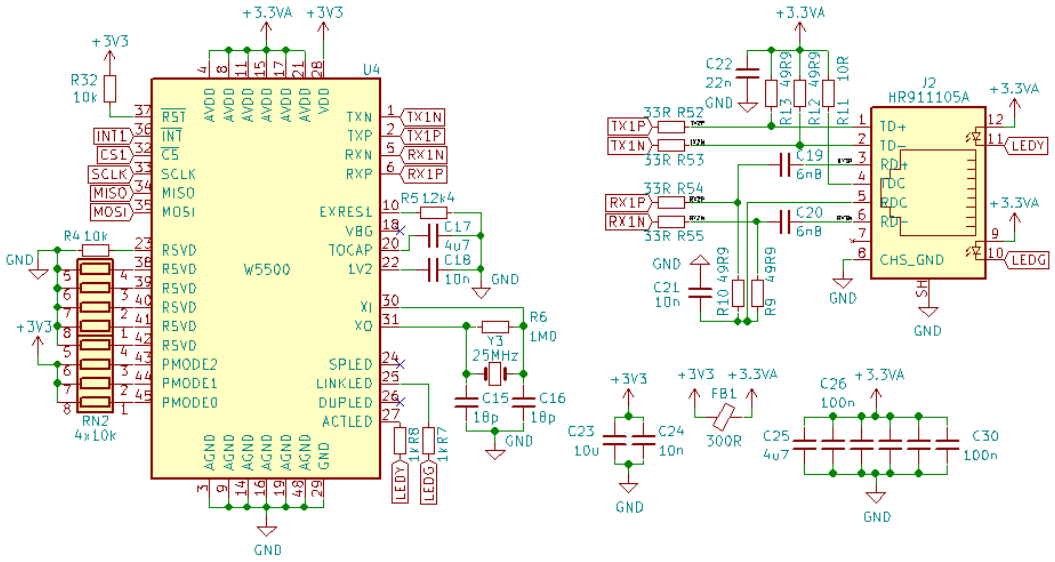
Pada penelitian ini jumlah I/O adalah 22, maka memori yang dibutuhkan adalah 1232 byte. 24C32 terhubung dengan MCU setelah melalui *converter* pada gambar 3.12 di titik SDA1 dan SCL1 karena 24C32 beroperasi pada tegangan 5V.

3.3.7 *Ethernet Controller*

Penulis memilih untuk menggunakan *ethernet* karena gedung modern sudah memiliki jaringan *ethernet*, sehingga tidak dibutuhkan instalasi baru. Sedangkan jika menggunakan serial seperti RS-485 dibutuhkan instalasi kabel baru. Untuk dapat terhubung dengan *ethernet*, MCU membutuhkan unit *ethernet controller*. Beberapa IC yang umum digunakan untuk purwarupa seperti ENC28J60, W5100 dan W5500. IC tersebut mengubah sinyal *ethernet* menjadi sinyal yang dapat dibaca oleh MCU melalui protokol seperti SPI (*serial peripheral interface*).

Unit *etheret controller* yang penulis pilih adalah Wiznet W5500, karena memiliki delapan kanal independen dengan *buffer* data yang lebih dari cukup untuk menampung format modbus, memiliki sudah memiliki *library* yang memudahkan penulis untuk memprogram dan beroperasi dengan SPI yang dapat dibagi dengan sensor daya CS5463. Berdasarkan datasheet, Wiznet W5500 dapat beroperasi pada jaringan *ethernet* 10/100Mbps.

Wiznet W5500 sudah memiliki PHY (*physical layer*) terintegrasi dimana sinyal diferensial dari konektor menjadi sinyal yang dapat dibaca oleh MCU dan juga sebaliknya. Kemudian data diteruskan ke MII (*Media independent interface*), selanjutnya ke inti *processor*. Data dan konfigurasi IC terhubung dengan MCU melalui SPI. Jalur SPI terbagi dengan CS5463 dengan CS (*chip select*) dan *interrupt* terpisah. Berikut skema rangkaian pada Wiznet W5500:



Gambar 3.13 Koneksi W5500 dengan konektor RJ45.

3.3.8 *Power Supply Unit* (PSU)

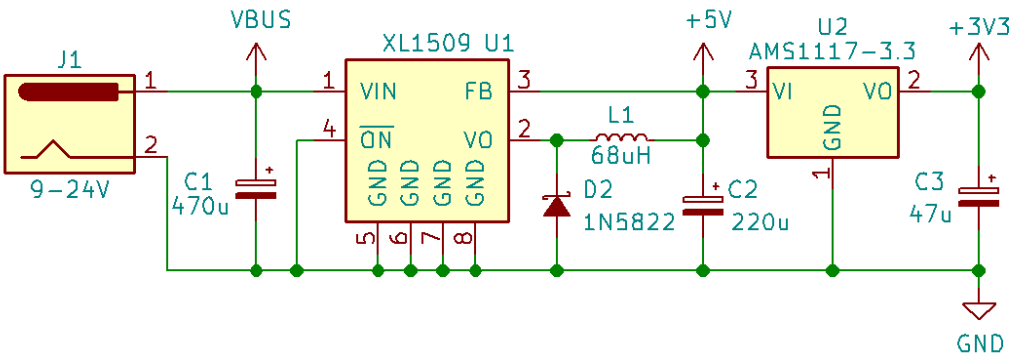
Power supply unit (PSU) diperlukan untuk setiap perangkat agar dapat bekerja. Skema PSU ditentukan oleh tegangan dan arus yang diperlukan. Dalam alat yang penulis kembangkan terdapat dua tegangan berbeda, yaitu 3,3V untuk MCU, sensor daya dan *ethernet controller*; 5V untuk sensor daya dan relay. Penggunaan arus sebenarnya akan bergantung pada fungsi dan *peripheral* yang aktif, tetapi arus yang akan dikalkulasi adalah arus maksimum, sehingga PSU dapat menyuplai cukup arus di saat beban maksimal. Penggunaan arus maksimum dapat dilihat pada *datasheet* masing-masing komponen.

Tabel 3.5 Arus maksimum setiap komponen berdasarkan *datasheet*.

Komponen	Jumlah	Arus (3,3V)	Arus (5V)
STM32	1	150mA	-
24C32	1	5mA	
W5500	1	132mA	-
CS5463	1	1,7mA	1,3mA
ELD207	6	6mA	-
ELQ3H7	8	40mA	
Relay	8	-	550mA
Total		334,7mA	551,3mA

Penulis memilih AMS1117-3.3 untuk regulator 3,3V yang memiliki kapasitas arus sebesar 1A dan XL1509-5.0 untuk regulator 5V yang memiliki kapasitas arus sebesar 2A. XL1509-5.0 mendapatkan tegangan dari DC adaptor, sedangkan AMS1117-3.3 mendapatkan tegangan dari hasil keluaran XL1509-5.0.

Regulator 5V digunakan adalah tipe *buck converter* XLSEMI XL1509-5.0 dengan kapasitas arus 2A dengan sumber mengunakan adaptor eksternal dengan tegangan hingga 40V. Penulis memilih *buck converter* karena efisiensi yang lebih tinggi dibandingkan dengan menggunakan *linear drop out* (LDO) regulator. Berdasarkan *datasheet*, XL1509-5.0 memiliki tingkat efisiensi dari 74% hingga 92% tergantung dari arus beban dan perbedaan tegangan antara input dan output. Regulator 3,3V yang digunakan adalah tipe LDO AMS1117-3.3 dengan kapasitass arus 1A dengan sumber tegangan dari keluaran XL1509-5.0. Penulis tidak menghubungkan AMS1117-3.3 dengan adaptor karena perbedaan tegangan yang terlalu besar yang akan menyebabkan panas terlalu tinggi. Dengan demikian arus yang harus disuplai oleh XL1509-5.0 adalah 551,3mA + 334,7mA = 886mA.



Gambar 3.14 Rangkaian regulator 3,3V dan 5V.

XL1509-5.0 adalah *buck converter* dimana transistor di dalamnya akan menyala dan mati dengan frekuensi tetap 150KHz dengan *duty cycle* (waktu transistor menyala) dari 0% atau mati sepenuhnya hingga 100% atau menyala sepenuhnya. Berdasarkan *datasheet*, XL1509-5.0 dapat menerima tegangan masukan dari 7V hingga 40V, dengan tegangan adaptor yang digunakan pada penelitian ini adalah 9V. XL1509-5.0 seperti *buck converter* lainnya membutuhkan komponen eksternal, yaitu kapasitor input, dioda *free-wheel*, induktor dan kapasitor

output. *Datasheet* sudah memiliki tipe dan nilai yang direkomendasikan untuk komponen tersebut dan penulis cukup mengikutinya.

AMS1117-3.3 adalah regulator *linear drop out* (LDO) dengan tegangan input maksiumum 15V dan tegangan *drop out* hingga 1,3V. Pada regulator LDO, arus yang mengalir pada sumber sama dengan pada beban, tidak seperti *buck regulator*. Karena itu, perbedaan tegangan yang terlalu besar akan menghasilkan panas dari daya terbuang P_D dapat dihitung dengan persamaan:

$$P_D = (V_{in} - V_{out}) \times I_{load} \quad (3.5)$$

Dengan V_{in} = 5V dan I_{load} = 0,3347A, maka disipasi daya P_D saat beban maksimum adalah:

$$P_D = (5V - 3{,}3V) \times 0{,}3347\text{A} = 0{,}56899W \quad (3.6)$$

Selanjutnya suhu regulator dapat ditentukan dengan persamaan:

$$T = \left(\varphi_{JA} \times P_D\right) + T_{Ambient} \quad (3.7)$$

$$T = (90℃/W \times 0{,}56899W) + 25℃ = 76{,}2091℃ \quad (3.8)$$

Perlu diperhatikan bahwa nilai φ_{ja} yang digunakan adalah nilai nominal untuk IC dengan tipe SOT-223 sperti yang penulis gunakan dalam penelitian ini. Nilai φ_{ja} dapat berubah tergantung dari pendinginan dan area tembaga PCB di bawah IC. Dengan asumsi tersebut, suhu operasional hasil kalkulasi di atas masih berada di bawah ambang batas suhu operasional maksimum yang bernilai 125^{o}C, sehingga penggunaan AMS1117-3.3 dapat dinyatakan aman.

3.3.9 *Printed Circuit Board* (PCB)

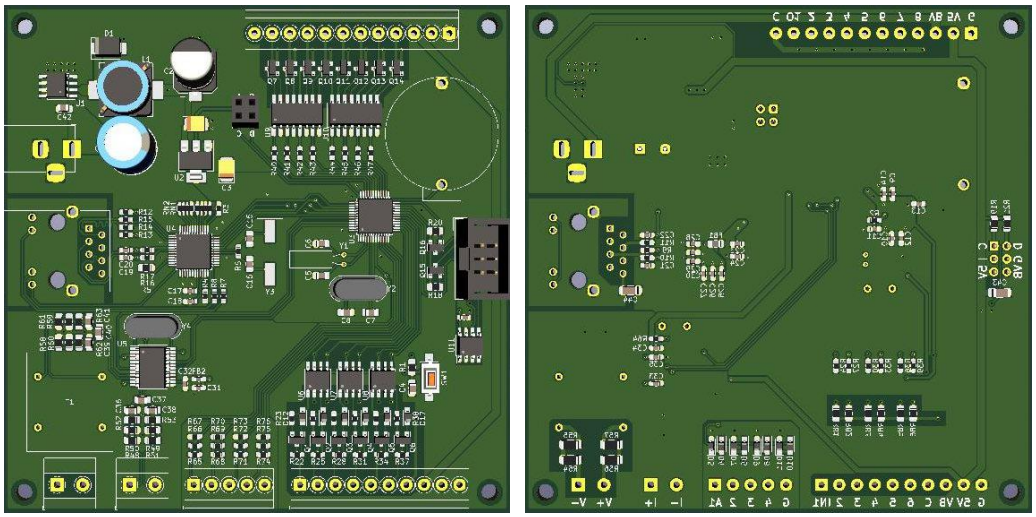
Printed circuit board (PCB) adalah media yang umum digunakan untuk membangun suatu rangkaian elektronik. Perancangan PCB menggunakan aplikasi KiCAD. Hal yang perlu diperhatikan dalam perancangan PCB adalah jarak antar komponen yang tidak terlalu dekat, pengelompokkan komponen, penempatan letak terminal I/O agar mudah di akses.

Jenis PCB yang penulis gunakan adalah tipe *double layer* dengan substrat FR4 atau *fiber* dengan ketebalan 1,6mm. PCB jenis ini dipilih karena PCB jenis ini adalah jenis yang umum dipakai pada aplikasi dimana terdapat beberapa jenis sinyal pada saat perangkat beroperasi. *Double layer* berarti PCB memiliki lapisan tembaga

pada bagian atas dan bawah, sehingga terdapat ruang untuk meletakkan komponen dan jalur baik pada bagian atas dan bawah.

Tipe komponen yang dipakai adalah tipe SMD (*solder mounted device*) untuk menghemat ruang karena komponen dengan format SMD memiliki ukuran yang jauh lebih kecil dibandingkan dengan komponen dengan format THT (*through hole technology*). Namun demikian penulis masih menggunakan komponen THT untuk komponen mekanik seperti *terminal block* untuk *wiring* I/O, DC Jack untuk adaptor, konektor RJ45 untuk *ethernet* dan komponen lain yang tidak memiliki format SMD.

Seluruh rangkaian MCU, keluaran diskrit, masukan diskrit, masukan analog, sensor daya dan *ethernet controller* kemudian dibuat dalam satu keping PCB. Modul relay yang digunakan akan terpisah dengan unit kontroler karena penulis menggunakan modul relay delapan jalur yang sudah tersedia di pasaran. Perancangan PCB didesain menggunakan KiCAD. Hasil desain berupa PCB dengan ukuran 10cm x 10cm seperti pada gambar 3.15. Hasil desain kemudian dikerjakan oleh *manufacturer*, sedangkan proses penyolderan komponen pada PCB dikerjakan oleh penulis.



Gambar 3.15 Hasil *render* 3D PCB pada *layer* atas dan bawah.

3.4 Penulisan Software

3.4.1 Program MCU

Aplikasi yang digunakan untuk menulis program pada mikrokontroler menggunakan IAR Embedded Workbench for ARM (EWARM). IAR EWARM adalah aplikasi berbasis *shareware* dimana pengguna dapat mencoba secara gratis selama 30 hari atau pembatasan besar aplikasi sebesar 32kB. Bahasa pemrograman yang digunakan adalah bahasa C dengan *library* CMSIS (*Cortex-M Standard Interface Ssytem*) dari ARM, HAL (*hardware abstraction layer*) dari STMicroelectronics dan ioLibrary (*Internet Offload Library*) dari Wiznet. *Library* tersebut berfungsi sebagai standar untuk berinteraksi dengan MCU serta menyediakan fungsi dan *subroutine* dasar yang mempersingkat waktu penulisan program secara keseluruhan sehingga penulis tidak perlu menulis program dari nol.

Aplikasi yang digunakan untuk menulis program HMI adalah menggunakan Microsoft Visual Studio *Community* 2015 dan AdvancedHMI 3.99x. Microsoft Visual Studio memiliki beberapa tingkat dari *community* yang bersifat gratis hingga *enterprise* yang bersifat *shareware*. Microsoft Visual Studio mendukung aplikasi berbasis .NET dengan beberapa jenis bahasa pemrograman, yaitu VB.NET, C# dan Visual C++. Dengan aplikasi berbasis .NET, penulis dapat membuat aplikasi dengan kombinasi dari ketiga bahasa di atas. Bagian driver AdvancedHMI yang sudah disediakan ditulis dengan C#, sedangkan komponen visual dan algoritma program yang penulis rancang ditulis menggunakan VB.NET.

Program MCU dimulai dengan konfigurasi MCU, seperti *clock*, NVIC dan *peripheral*. Penulis menggunakan STM32CubeMX, aplikasi dari ST untuk pengembang untuk mempermudah konfigurasi dasar perangkat keras MCU untuk selanjutnya program dikembangkan pada IDE (*integrated development environments*). IDE yang didukung diantaranya adalah IAR EWARM, TrueStudio, MDK ARM, dan Eclipse gcc. Pada penelitian ini penulis memilih IAR EWARM sebagai IDE dan menggunakan ST HAL (*hardware abstraction layer*) API (*application programming interfaces*) dari STM32CubeMX sebagai standar antarmuka dengan MCU, sehingga penulis tidak perlu mengakses register dan memori secara langsung.

Pertama penulis mengatur *clock* pada MCU sebesar 32MHz. Kemudian mengatur konfigurasi GPIO yang diperlukan seperti pada gambar 3.16. Pada pin keluaran, konfigurasi yang digunakan adalah mode *push-pull* dengan kecepatan rendah. Pada pin masukan, konfigurasi yang digunakan adalah koneksi dengan EXTI dengan *interrupt* saat *falling* dan *rising* serta *pull-down* resistor internal. Pada jalur *interrupt* SPI dan I2C, konfigurasi yang digunakan adalah koneksi dengan EXTI dengan *interrupt* saat *falling* dan *pull-up* resistor internal.

Pin Name	GPIO mode	GPIO Pull-up/Pull-down	User Label
PB6	Output Push Pull	Pull-up	CS5460_CS
PA15	External Interrupt Mode with Falling edge trigger detection	Pull-up	CS5460_INT
PB8	External Interrupt Mode with Rising/Falling edge trigger detection	No pull-up and no pull-down	CS5460_ZC
PA4	External Interrupt Mode with Rising/Falling edge trigger detection	Pull-down	GPIO_IN0
PA5	External Interrupt Mode with Rising/Falling edge trigger detection	Pull-down	GPIO_IN1
PA6	External Interrupt Mode with Rising/Falling edge trigger detection	Pull-down	GPIO_IN2
PA7	External Interrupt Mode with Rising/Falling edge trigger detection	Pull-down	GPIO_IN3
PB0	External Interrupt Mode with Rising/Falling edge trigger detection	Pull-down	GPIO_IN4
PB1	External Interrupt Mode with Rising/Falling edge trigger detection	Pull-down	GPIO_IN5
PA12	Output Push Pull	No pull-up and no pull-down	GPIO_OUT0
PA11	Output Push Pull	No pull-up and no pull-down	GPIO_OUT1
PA10	Output Push Pull	No pull-up and no pull-down	GPIO_OUT2
PA9	Output Push Pull	No pull-up and no pull-down	GPIO_OUT3
PA8	Output Push Pull	No pull-up and no pull-down	GPIO_OUT4
PB15	Output Push Pull	No pull-up and no pull-down	GPIO_OUT5
PB14	Output Push Pull	No pull-up and no pull-down	GPIO_OUT6
PB13	Output Push Pull	No pull-up and no pull-down	GPIO_OUT7
PB2	External Interrupt Mode with Falling edge trigger detection	Pull-up	I2C_INT
PB7	Output Push Pull	Pull-up	W5500_CS
PB9	External Interrupt Mode with Falling edge trigger detection	Pull-up	W5500_INT

Gambar 3.16 Konfigurasi GPIO pada MCU.

STM32 berkomunikasi dengan CS5463 dan W5500 dengan SPI. Frekuensi maksimal SPI yang didukung oleh kedua IC ini berbeda, dengan CS5463 hanya sebesar 2MHz dan W5500 sebesar 25MHz. Dengan keuda batas frekuensi yang jauh berbeda, penulis menerapkan kode dimana frekuensi SPI akan diubah sesuai dengan batas IC tujuan. Dengan mengandalkan *prescaler* pada SPI *clock*, frekuensi yang dapat dihasilkan adalah 16MHz, 8MHz, 2MHz, 1MHz hingga 125kHz. Penulis memilih 8MHz untuk frekuensi SPI saat berkomunikasi dengan W5500 dan 2MHz untuk CS5463. Sedangkan komunikasi dengan I/O eksternal menggunakan I2C dengan frekuensi terkecil yang didukung oleh IC *slave*, yaitu mode standar dengan frekuensi 100KHz.

Konfigurasi pada NVIC adalah menentukan jalur *interrupt* yang aktif dan prioritasnya. Sistem prioritas pada STM32F103 terdiri dari 16 tingkat dengan nilai

yang lebih rendah menandakan prioritas yang lebih penting. Konfigurasi NVIC yang penulis terapkan dapat dilihat pada gambar 3.17.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
RTC global interrupt	☑	1	0
ADC1 and ADC2 global interrupts	☑	2	0
EXTI line0 interrupt	☑	3	0
EXTI line1 interrupt	☑	3	0
EXTI line4 interrupt	☑	3	0
EXTI line[9:5] interrupts	☑	4	0
SPI1 global interrupt	☑	4	0
EXTI line[15:10] interrupts	☑	4	0
EXTI line2 interrupt	☑	5	0
I2C2 event interrupt	☑	5	0
I2C2 error interrupt	☑	5	0

Gambar 3.17 Konfigurasi NVIC *interrupt table*.

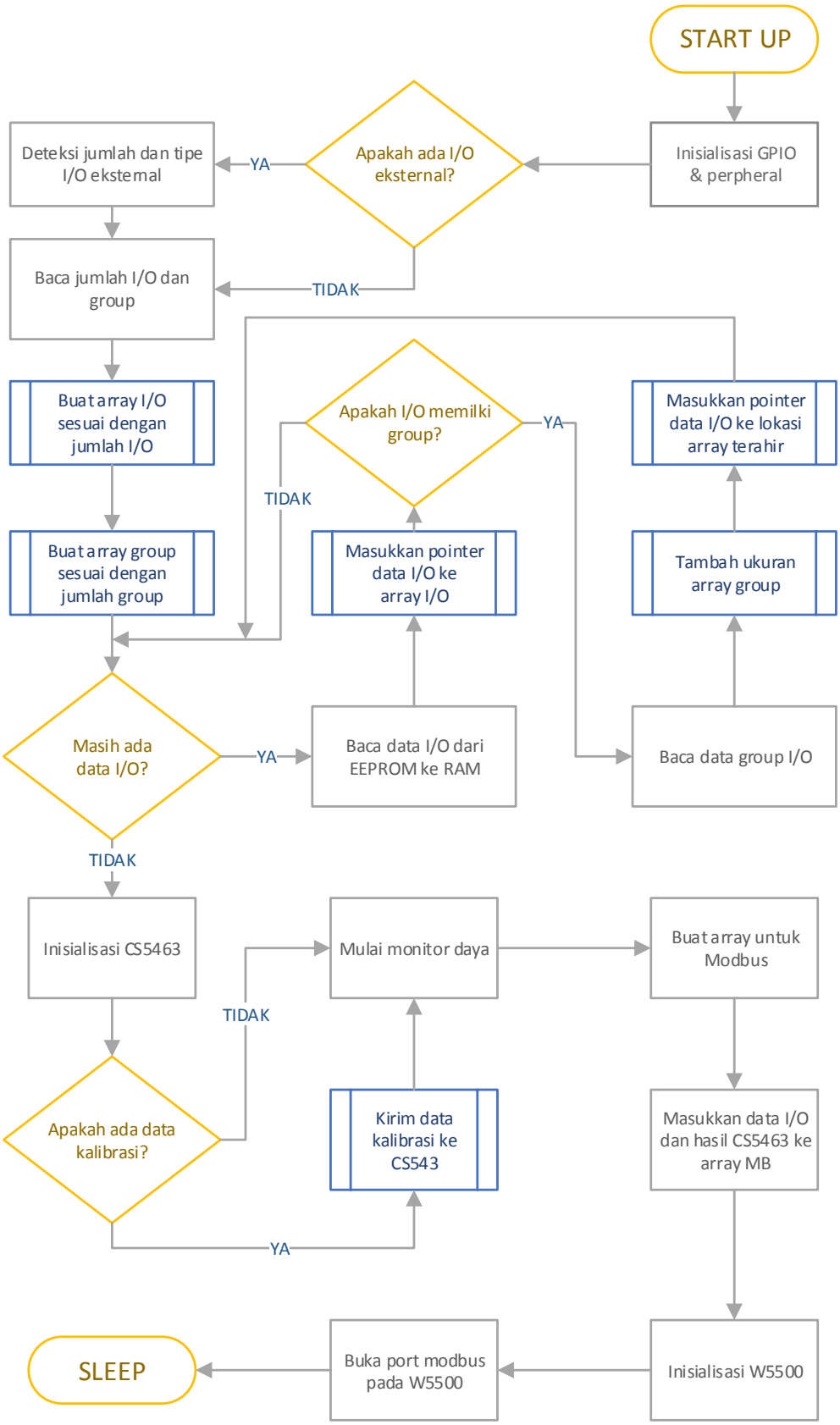
Program hasil STM32CubeMX hanya sampai tahap inisialisasi *hardware*. Program kemudian dikembangkan menggunakan EWARM. Setelah inisialisasi, program akan membaca data I/O yang sudah terimpan, banyaknya I/O dan grup, kemudian membuat array untuk I/O tersebut. Data I/O terdiri atas parameter berkut:

a. Tipe I/O, apakah input atau output;
b. Grup, apakah I/O bagian dari suatu grup I/O;
c. *Link* atau hubungan, di saat input I/O ini berubah keadannya, I/O atau grup yang terhubung dengannya akan berubah pula;
d. Jadwal, apkah I/O ini harus menyala atau mati di waktu tertentu;
e. Kondisi, apakah I/O dalam kondisi *active*, *inactive*, *above*, *within* atau *below*.

Ukuran array untuk I/O adalah banyaknya I/O yang dideteksi MCU. Setelah daftar I/O terbentuk, MCU membuat array untuk grup yang berisi jumlah I/O dan alamat memori I/O pada group tersebut dan jadwal I/O. Diagram alir program MCU saat *start-up* dapat dilihat pada gambar 3.18.

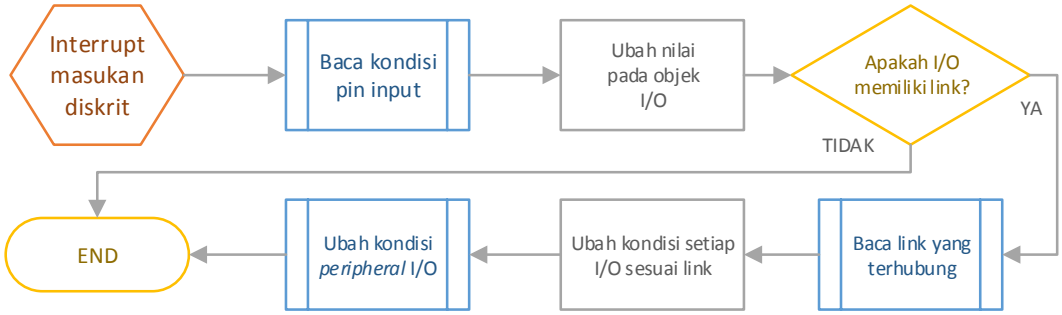
Setelah tahap ini, MCU siap menerima *interrupt* dari EXTI, I2C, CS5463 dan W5500. Saat *interrupt* terjadi, MCU menjalankan program ISR (*interrupt service routine*) sesuai dengan jalur *interrupt* yang mengaktifkannya. Jika beberapa

interrupt terjadi bersamaan atau MCU sedang memproses *interrupt* lain, maka MCU akan memilih ISR sesuai dengan tingkat prioritas (*preemption*) atau menunda *interrupt* yang datang sampai ISR yang sedang dijalankan selesai (*tailgate*). Saat MCU tidak menjalankan program ISR manapun, MCU akan masuk *sleep mode*, dimana seluruh *peripheral* masih aktif, tetapi CPU tidak menjalankan program apapun.



Gambar 3.18 Diagram alir program MCU saat *start-up*.

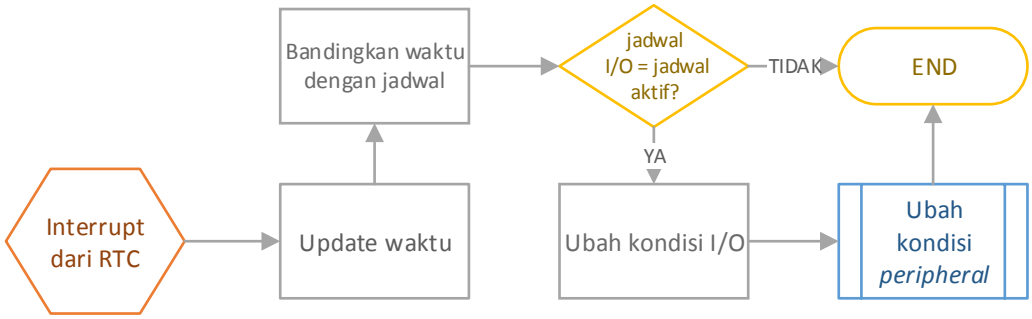
Jalur EXTI yang digunakan adalah delapan dengan enam masukan diskrit, satu W5500, satu CS5463 dan satu I2C. Saat *interrupt* aktif, MCU harus melihat jalur mana yang mengaktifkan EXTI untuk dapat menentukan program yang akan dijalankan. Berikut alur program ISR untuk masukan diskrit:



Gambar 3.19 Alur program ISR untuk masukan diskrit.

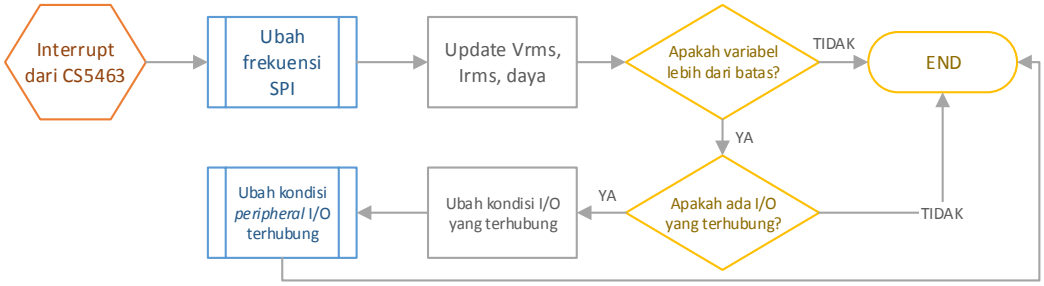
Dapat dilihat di diagram di atas, masukan diskrit dapat terhubung dengan individaual I/O atau group I/O jika memiliki link yang kemudian terhubung dengan keluaran diskrit. Alur program seperti ini memungkinkan reaksi yang cepat dan fleksibel antara input dan output tanpa adanya campur tangan dari luar. Operator gedung cukup mengubah parameter link pada input menggunakan HMI tanpa memprogram ulang MCU. Aplikasi dari fitur ini diantaranya adalah menyalakan lampu saat sensor mendeteksi orang, menyalakan alarm saat sensor mendeteksi api, mematikan katup pipa air saat ada kebocoran dan lain sebagainya.

Pada digital output, terdapat fitur *schedule* atau penjadwalan, dengan fitur ini I/O dapat menyala dan mati di waktu terrtentu. Pengecekan jadwal dilakukan setiap detik dari *interrupt* RTC. Berikut alur program ISR pada *interrupt* RTC:



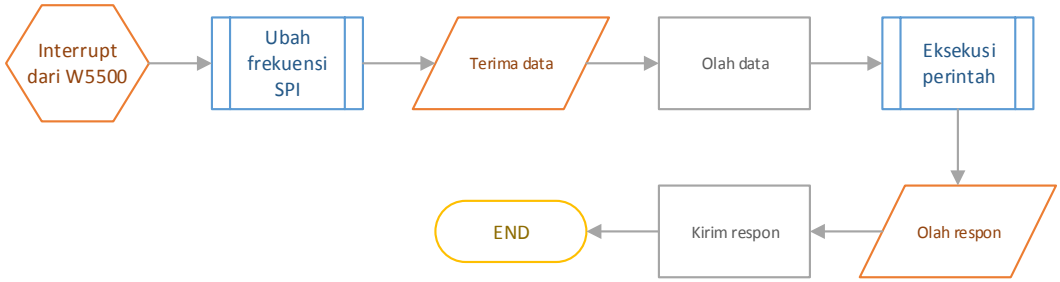
Gambar 3.20 Alur program ISR untuk RTC.

Interrupt dari CS5463 terjadi setiap nilai RMS dan daya telah diperbaharui. ISR akan memperbaharui setiap variabel dan melihat apakah ada nilai yang melebihi batas atas atau kurang dari batas bawah, kemudian merespon sesuai dengan konfigurasi. Berikut alur program ISR pada *interrupt* CS5463:



Gambar 3.21 Alur program ISR untuk CS5463.

Interrupt dari W5500 terjadi setiap data masuk dari HMI. Data dari HMI terdiri dari permintaan data MCU, perubahan konfigurasi *hardware* dan I/O atau peintah lain. Setelah data diterima kemudian MCU memproses data, melakukan perintah dan memberikan data respon kembali ke HMI. Berikut alur program ISR pada *interrupt* W5500:



Gambar 3.22 Alur program ISR untuk W5500.

3.4.2 Aplikasi HMI

Aplikasi HMI yang penulis gunakan adalah AdvancedHMI versi 3.99x, yang dapat menggunakan Visual Studio *Community* 2015 dengan bahasa VB.NET dan protokol berbasis Modbus TCP. *Function code* pada protokol modbus yang digunakan dalam penelitian ini adalah *user-defined function code*, yaitu:

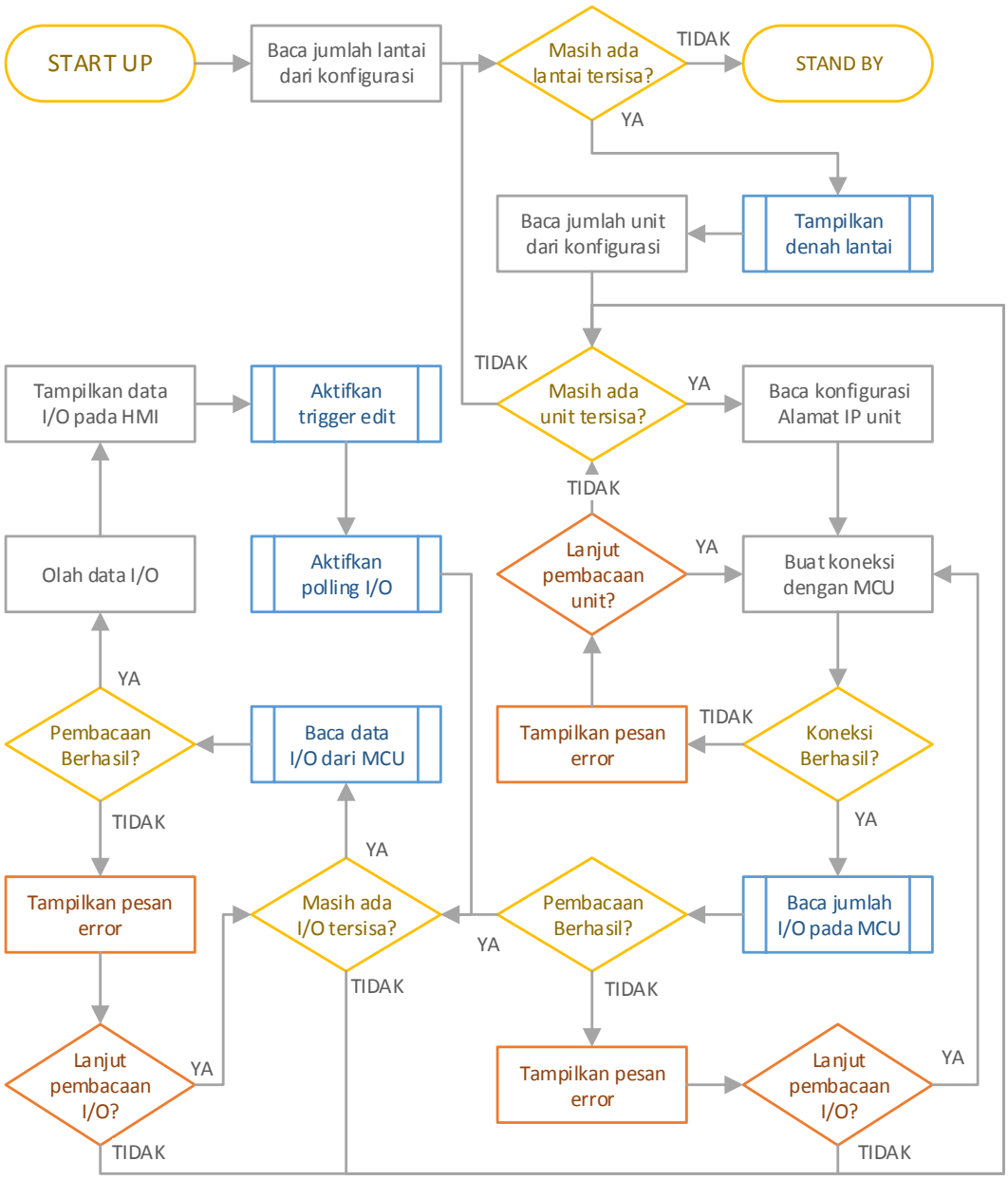
66: Baca data I/O
67: Tulis data I/O
68: Baca data jadwal
69: Tulis data jadwal
70: Baca waktu
71: Tulis waktu

Pada HMI yang dirancang terdapat objek ‘Lantai’ yang terdiri atas denah lantai dan kumpulan objek ‘Unit’ yang terdiri atas kumpulan I/O dan jadwal. Pada penelitian ini penulis hanya membangun satu unit kontroler, sehingga konfigurasi pada program HMI adalah 1 lantai yang terdiri atas 1 unit.

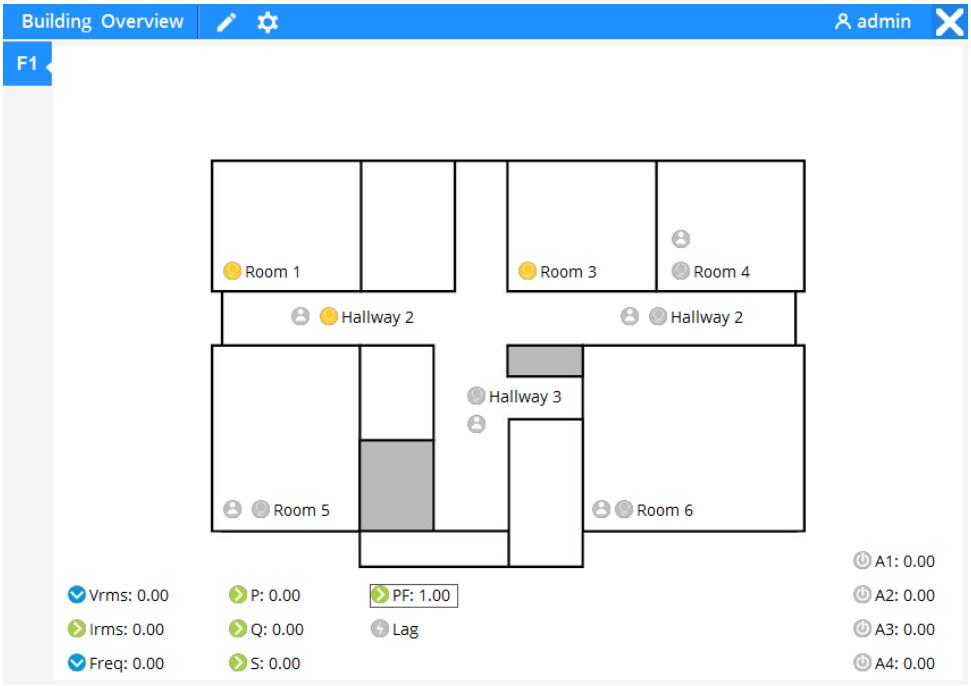
Program dimulai dengan membaca konfigurasi tersebut dan memulai *loop* untuk mendapatkan data seluruh ‘Unit’. Pada proses ini dilakukan cek pada setiap pembacaan untuk menghindari pembacaan dari unit kontroler yang tidak tersambung, malfungsi atau pembacaan dari alamat IP yang salah. Jika terjadi error, program akan memunculkan pesan bahwa terjadi error dan meminta respon pengguna apakah pembacaan dilanjutkan atau dilewat dan menampilkan data yang ada. Pengaturan jumlah lantai, alamat IP unit kontroler dan lain sebagainya dapat diatur setelah aplikasi terbuka. Diagram alir pada saat *start-up* aplikasi dapat dilihat pada gambar 3.23.

Setelah program terbuka maka akan muncul tampilan denah lantai dan tombol yang melambangkan setiap I/O pada *tab* yang melambangkan posisi lantai di sisi kiri. Denah lantai yang ditampilkan dapat diubah oleh pengguna dengan mengganti gambar pada *folder* tempat aplikasi berada. Untuk mengganti tampilan lantai yang ditampilkan, pengguna cukup memilih *tab* lantai yang diinginkan. Pada kondisi ini pengguna dapat memulai interaksi dengan I/O, atau jika belum dibutuhkan, tampilan aplikasi dapat dihilangkan tanpa menutup program. Pada bagian *backgroud*, HMI melakukan request data I/O setiap 500ms, kemudian me-*refresh* tampilan tombol.

Setiap tombol yang berada pada denah lantai melambangkan I/O yang ditampilkan sebagai *icon* yang mengindikasikan kondisi I/O pada sisi kiri diiringi dengan nama I/O dan nilainya untuk masukan analog. Pada keluaran diskrit, tombol tersebut dapat ditekan untuk mengubah kondisi keluaran, tetapi pada masukan fungsi ini tidak tersedia. Untuk setiap tombol pada HMI, pengguna dapat mengatur posisinya sesuai dengan keinginan mengikuti denah lantai dengan memilih *icon* yang berbentuk pensil pada bagian atas aplikasi kemudian menggeser tombol ke posisi yang dikehendaki. Tampilan program HMI dapat dilihat pada gambar 3.23.

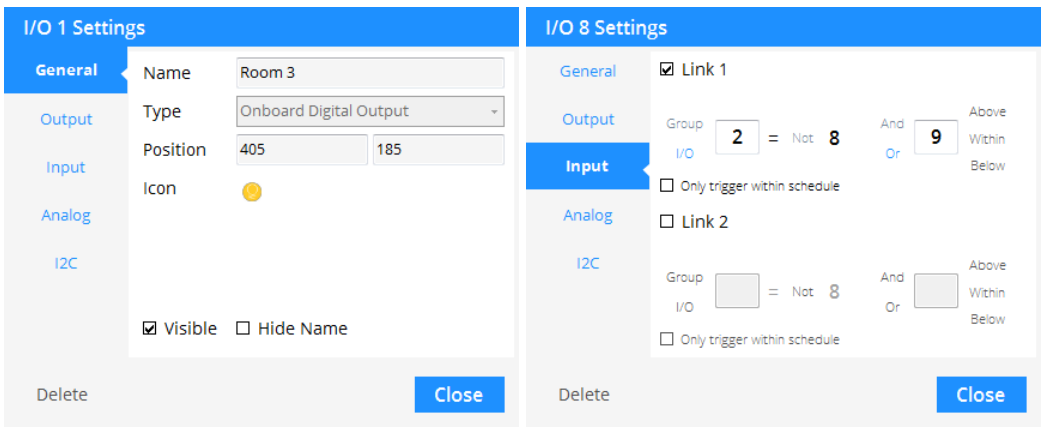


Gambar 3.23 Diagram alir program HMI saat *start-up*.



Gambar 3.24 Tampilan HMI setelah *start-up*.

Untuk mengatur parameter I/O, pengguna dapat menekan tombol F3 saat kursor berada di I/O yang ingin diubah yang akan memunculkan *window form* seperti pada gambar 3.25. Pada bagian ini, terdapat beberapa *tab* yang berisi parameter I/O sesuai dengan jenisnya. Pada *tab* General berisi konfigurasi nama, lokasi tombol I/O, jenis *icon* dan visibilitas tombol I/O. Pada *tab* Input berisi konfigurasi *link*. Pada *tab* Output berisi konfigurasi *schedule* dan grup. Pada *tab* Analog berisi batas atas, batas bawah, pengali dan *offset*. Setiap perubahan yang dilkakukan oleh pengguna langsung disimpan ke MCU dan EEPROM.



Gambar 3.25 Contoh tampilan konfigurasi I/O.

3.5 Pengujian Alat

Setelah program ditulis, untuk mengecek apakah sistem berfungsi dengan baik atau tidak, maka dilakukan pengujian berdasarkan spesifikasi sistem pada bagian 3.2. Dengan melihat hasil pengujian, penulis dapat menentukan apakah sistem berfungsi dengan baik atau tidak. Pengujian alat dilakukan dengan metode *black box testing*. Berikut pengujian yang penulis lakukan:

3.5.1 Pengujian PSU

Pengujian PSU dilakukan dengan menghubungkan unit kontroler dengan adaptor 9V, kemudian mengecek tegangan hasil keluaran XL1509-5.0 dan AMS1117-3.3. Toleransi tegangan untuk 5V adalah 4,75V – 5,25V mengikuti batas atas dan batas bawah dari CS5463. Sedangkan toleransi untuk 3,3V adalah 3,135V – 3,6V mengikuti batas bawah dari CS5463 dan batas atas dari STM32. Pengukuran tegangan dilakukan menggunakan multimeter pada mode DC voltage. Pengujian dinyatakan berhasil jika tegangan yang terukur berada dalam toleransi.

Tabel 3.6 Tabel pengujian PSU.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	Menyalakan regulator XL1509-5.0.	Input tegangan menggunakan adaptor 9V.	XL1509-5.0 menghasilkan tegangan 4,75V – 5,25V.
2	Menyalakan regulator AMS1117-3.3.	Input tegangan menggunakan keluaran XL1509-5.0.	AMS1117-3.3 menghasilkan tegangan 3,135V – 3,6V.

3.5.2 Pengujian RTC

Pengujian RTC dilakukan dengan menggunakan *debugger* pada EWARM. RTC diatur terlebih dahulu agar menghasilkan *interrupt* setiap detik dan memasang *breakpoint* pada ISR yang akan menampilkan waktu MCU, kemudian program akan dibiarkan berjalan. Jika RTC, EXTI, dan kode program berfungsi dengan baik, maka *breakpoint* akan tercapai dan program MCU akan terhenti pada titik *breakpoint* tersebut. Pengujian dinyatakan berhasil jika *breakpoint* terjadi dan RTC menghasilkan *interrupt* setiap detik.

Tabel 3.7 Tabel pengujian RTC.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	MCU beroperasi normal.	Data RTC *default*	RTC aktif setiap detik tanpa mengecek jadwal.
2	MCU mengubah data RTC.	Data RTC *default*	RTC langsung mengecek jadwal.
3	MCU beroperasi normal.	Data RTC telah diubah	RTC aktif setiap detik dan mengecek jadwal.
4	MCU mati.	Data RTC *default*	RTC tetap aktif setiap detik tanpa mengecek jadwal.
5	MCU mati.	Data RTC telah diubah	RTC tetap aktif setiap detik.

3.5.3 Pengujian Keluaran Diskrit

Pengujian keluaran diskrit dengan menulis tes kode yang mengaktifkan keluaran satu persatu, kemudian dimatikan satu persatu dengan *breakpoint* pada masing-masing baris. Pengukuran dilakukan dengan melihat kondisi pada pin MCU dengan multimeter pada mode DC voltage dan transistor pada *optocoupler* dengan multimeter pada mode dioda. Pengujian dinyatakan berhasil jika pada kondisi keluaran *high*, tegangan pada pin MCU sama dengan tegangan sumber dan drop tegangan trasistor pada *optocoupler* terbaca oleh multimeter; pada kondisi *low*, tegangan pada pin MCU adalah nol dan tegangan trasistor pada *optocoupler* tidak dapat terbaca pada multimeter.

Contoh kode untuk menyalakan pin keluaran:

```
GPIO_OUT0_GPIO_Port->BSRR = GPIO_OUT0_Pin;
GPIO_OUT1_GPIO_Port->BSRR = GPIO_OUT1_Pin;
```

Contoh kode untuk mematikan pin keluaran:

```
GPIO_OUT6_GPIO_Port->BRR = GPIO_OUT6_Pin;
GPIO_OUT7_GPIO_Port->BRR = GPIO_OUT7_Pin;
```

Tabel 3.8 Tabel pengujian keluaran diskrit.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	Menyalakan keluaran diskrit.	Keluaran diskrit *low* Terminal keluaran terhubung dengan relay	Tegangan 3,3V di pin keluaran Transistor aktif Relay menyala
2	Mematikan keluaran diskrit.	Keluaran diskrit *high* Terminal keluaran terhubung dengan relay	Tegangan 0V di pin keluaran Transistor tidak aktif Relay mati

3.5.4 Pengujian Masukan Diskrit

Pengujian masukan diskrit yang dilakukan menggunakan *debugger* pada EWARM dengan memasang *breakpoint* pada setiap ISR agar EXTI yang aktif dapat terlihat, kemudian memberi tegangan pada masing-masing terminal masukan dimulai dari masukan diskrit 1 hingga 6 atau I/O8 hingga I/O13. Pengujian dinyatakan berhasil jika *breakpoint* terjadi dan program MCU tertahan.

Tabel 3.9 Hasil pengujian masukan diskrit.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	Trigger manual terminal input dengan tegangan 5V.	*Breakpoint debugger* pada setiap ISR masukan diskrit.	*Breakpoint* terjadi program MCU tertahan.
2	Trigger masukan dengan sensor PIR (3,3V).	*Breakpoint debugger* pada setiap ISR masukan diskrit.	*Breakpoint* terjadi program MCU tertahan.

3.5.5 Pengujian Sensor Daya

Pengujian nilai konversi CS5463 dengan menggunakan STMStudio untuk mengetahui nilai variabel pada MCU secara real time. Pengujian dilakukan dengan menulis tujuh *dummy* variable untuk menampung setiap nilai konversi CS5463, menghubungkan tegangan jala-jala pada masukan tegangan dan trafo arus dengan

beban dan melihat hasil konversi pada STMStudio. CS5463 dinyatakan berfungsi jika hasil konversi menunjukkan nilai yang sesuai.

Tabel 3.10 Tabel pengujian sensor daya.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	Pembacaan sensor saat tidak ada input	Konfigurasi sensor *default*	Nilai pembacaan tegangan = 0 Nilai pembacaan arus = 0
2	Pembacaan sensor saat tegangan jala-jala tanpa beban.	Konfigurasi sensor *default*	Nilai pembacaan tegangan = tegangan jala-jala. Nilai pembacaan arus = 0
3	Pembacaan sensor saat tegangan jala-jala dengan beban	Konfigurasi sensor *default*	Nilai pembacaan tegangan = tegangan jala-jala. Nilai pembacaan arus = pembacaan *clamp meter*

3.5.6 Pengujian *Trigger* Otomatis Menggunakan Jadwal

Pengujian fitur *schedule* dilakukan dengan mengubah data jadwal pada HMI, kemudian mengatur keluaran diskrit agar terhubung dengan jadwal yang tersebut. Setelah itu, unit kontroler akan didiamkan hingga mencapai waktu jawal yang telah ditentukan dan melihat apakah keluaran diskrit aktif atau tidak. Jika aktif, pengujian dilanjutkan dengan melakukan *restart* pada unit kontroler untuk melihat apakah jadwal dapat tetap mengaktifkan keluaran. Pengujian dilanjutkan hingga waktu jadwal berakhir dan melihat apakah keluaran mati atau tidak.

Tabel 3.11 Tabel pengujian jadwal.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	Keluaran diskrit dengan jadwal aktif	Keluaran diskrit terhubung dengan jawal aktif. Keluaran diskrit mati.	Keluaran diskrit *high*.
2	Keluaran diskrit dengan jadwal tidak aktif	Keluaran diskrit terhubung dengan jawal tidak aktif. Keluaran diskrit menyala.	Keluaran diskrit *low*.
3	Mengaktifkan jadwal tidak aktif	Keluaran diskrit terhubung dengan jadwal tidak aktif. Keluaran diskrit mati.	Keluaran diskrit *high*.

4	Mematikan jadwal aktif	Keluaran diskrit terhubung dengan jadwal aktif. Keluaran diskrit menyala.	Keluaran diskrit *low*.

3.5.7 Pengujian *Trigger* Otomatis Menggunakan Masukan Diskrit

Pengujian ini dilakukan menggunakan HMI untuk mengatur parameter *link* pada masukan diskrit agar terhubung dengan keluaran diskrit, kemudian menghubungkan terminal input dengan tegangan agar dapat mengatur input secara manual. Pengujian dapat dinyatakan berhasil jika masukan diskrit dapat mengktifkan keluaran diskrit sesuai dengan parameter yang ditentukan.

Tabel 3.9 Tabel pengujian *link*.

No.	Skenario Pengujian	Prekondisi	Hasil yang Diharapkan
1	Masukan *low*	Keluaran diskrit terhubung dengan masukan. Konfigurasi *link* normal.	Keluaran diskrit *low*.
2	Masukan *low*	Keluaran diskrit terhubung dengan masukan. Konfigurasi *link* Not.	Keluaran diskrit *high*.
3	Masukan *high*	Keluaran diskrit terhubung dengan masukan. Konfigurasi *link* normal.	Keluaran diskrit *high*.
4	Masukan *high*	Keluaran diskrit terhubung dengan masukan. Konfigurasi *link* Not.	Keluaran diskrit *low*.
5	Masukan *low*	Keluaran diskrit terhubung dengan masukan. Keluaran diskrit terhubung dengan jadwal aktif. Konfigurasi *link* menghormati jadwal.	Keluaran diskrit *low*
6	Masukan *high*	Keluaran diskrit terhubung dengan masukan. Keluaran diskrit terhubung dengan jadwal aktif. Konfigurasi *link* menghormati jadwal.	Keluaran diskrit *high*.

7	Masukan *low*	Keluaran diskrit terhubung dengan masukan. Keluaran diskrit terhubung dengan jadwal tidak aktif. Konfigurasi *link* menghormati jadwal.	Keluaran diskrit tidak berubah.
8	Masukan *high*	Keluaran diskrit terhubung dengan masukan. Keluaran diskrit terhubung dengan jadwal tidak aktif. Konfigurasi *link* menghormati jadwal.	Keluaran diskrit tidak berubah.