PENGEMBANGAN SOFTWARE MANAJEMEN DATA UNTUK ENERGY AWARENESS SYSTEMBERBASIS WIRELESS SENSOR-ACTUATOR NETWORK

Christian Sinatra [1*], Henry Hermawan [1]
[1] Jurusan Teknik Elektro, Fakultas Teknik, Universitas Surabaya, Raya Kalirungkut Surabaya
*corresponding author: s160114002@student.ubaya.ac.id

Abstrak – Penggunaan energi dunia terus meningkat dan sepertiganya digunakan oleh bangunan. Salah satu cara mengatasinya adalah menggunakan *energy awareness system* untuk mengontrol penggunaan energi bangunan. Pada skripsi ini, sistem tersebut diimplementasikan di ruang kelas Fakultas Teknik Universitas Surabaya. Sistem tersebut mengontrol lampu dan AC berdasarkan sensor cahaya, gerak, dan suhu. Sistem ini terdiri dari beberapa komponen: kontroler *slave*, kontroler master, *router node*, koordinator, dan server. Fokus skripsi ini adalah *software* manajemen pada koordinator, termasuk desain struktur *database*. Pengerjaan dibagi menjadi empat tahap: desain *software* manajemen data keseluruhan, pembuatan *driver* AT86RF233, desain *database*, dan pembuatan *software* manajemen data secara utuh. Pengujian meliputi uji SPI, uji *transceiver*, uji akses *database*, uji *software* manajemen data, dan uji integrasi sistem besar. Hasil menunjukkan bahwa *software* yang dibuat berfungsi dan mampu diintegrasikan dengan sistem besar.
Kata kunci**:** WSAN, manajemen data, *database*

Abstract – World energy consumption continues to rise and a third of it is used for buildings. One way to overcome it is to implement energy awareness systems to control energy usage of buildings. In this study, such system is implemented at a classroom on a building of the Engineering Department of Universitas Surabaya. The system controls AC and lamps based on movement, temperature, and light sensors. The system consists of several components: slave controller, master controller, router node, coordinator, and server. The focus of this study is the data management software on the coordinator, which includes database development. The work is divided into four steps: software design, communications driver development, database design, and software integration. Testing includes SPI communications test, database access test, integrated data management software test, and system integration test. The result shows that the software made in this study is functional and is successfully integrated into the system.

Keywords**:** WSAN, data management, database

PENDAHULUAN

Menurut data *International Energy Agency* (IEA) [1], konsumsi energi dunia terus bertambah dan sepertiganya digunakan untuk bangunan. Menurut beberapa studi [2, 3], bangunan baru bisa didesain dengan prinsip *green architecture* agar lebih hemat energi. Hal itu tidak bisa diterapkan pada bangunan lama tanpa renovasi.

Studi yang dilakukan oleh Timilehin Labeodan, dkk. [4] menunjukkan penggunaan *energy awareness system* dapat menjadi salah satu cara agar bangunan lama mampu menghemat energi. Studi tersebut menggunakan sistem yang terbuat dari *wireless sensor-actuator network* (WSAN) untuk mengontrol lampu berdasarkan keberadaan orang. Data penggunaan listrik menunjukkan penurunan 24% menggunakan sistem WSAN dibandingkan dengan kontrol manual.

Pada tahun 2014, ada sebuah penelitian [5] tentang pengembangan sebuah model purwarupa *energy awareness system* yang bersifat *low-cost*, *low-power*, dengan proses instalasi yang mudah. Purwarupa tersebut mampu mengontrol lampu dan AC serta memiliki *web-server* untuk proses *monitoring* dan kontrol jarak jauh. Purwarupa tersebut memiliki model komunikasi yang sangat sederhana tapi andal. Purwarupa tersebut juga terbukti berhasil mengontrol lampu dan AC dalam satu bangunan.

Namun, purwarupa tersebut punya beberapa hal yang bisa dikembangkan lebih lanjut. Salah satunya adalah algoritma yang kurang efisien. Semua kontroler master memulai proses kontrol dengan *request* data ke server dalam bentuk *file* "stat.php" yang hanya berisi dua *byte*: satu *byte* alamat dan satu *byte* perintah. Isi "stat.php" yang dibaca belum tentu milik master yang sesuai.

Mikrokontroler yang digunakan juga dasarnya tidak mendukung teknologi Ethernet. Mikrokontroler Arduino yang digunakan harus ditambahi *shield* khusus Ethernet agar bisa berkomunikasi dengan koordinator.

Hal lain yang kurang pada purwarupa ini terletak pada struktur *database* yang digunakan. *Database* pada purwarupa ini memiliki tabel "data" yang menyimpan data terbaru dari masing-masing kontroler, dan tabel "log" yang menyimpan *history* pengaturan yang dibuat oleh *user*. Dua tabel ini sudah cukup, namun akan lebih baik jika ditambah tabel seperti *log* data kontroler untuk penelitian lanjutan.

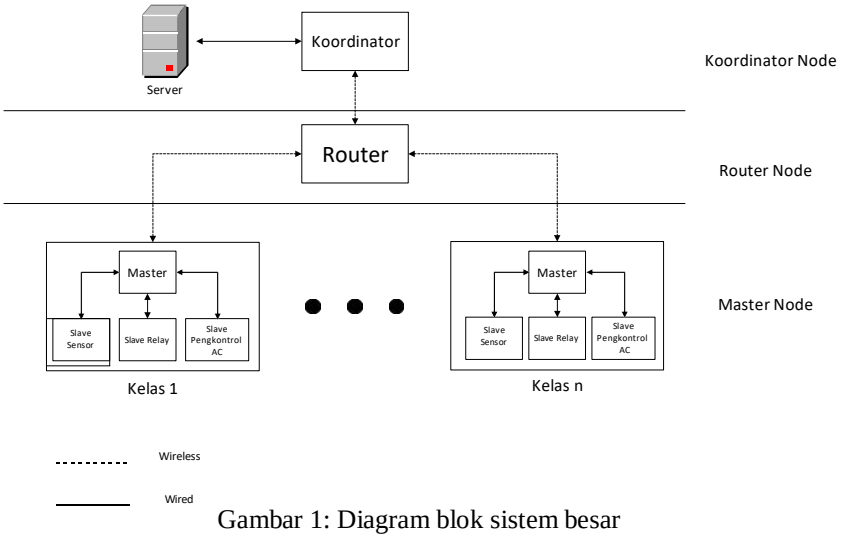
Kontrol otomatis yang ada pada purwarupa ini juga terbatas. Kontrol AC yang disediakan oleh purwarupa ini berupa pengganti *remote control* yang diakses melalui *web-server*. Kontrol lampu otomatis yang ada juga terbatas karena nilai *setpoint* untuk keputusan mati-nyala lampu menggunakan nilai yang *hard-coded*.

Dengan berdasar pada penelitian tersebut, sebuah *software* manajemen data untuk *energy awareness system* berbasis WSAN akan dikembangkan pada skripsi

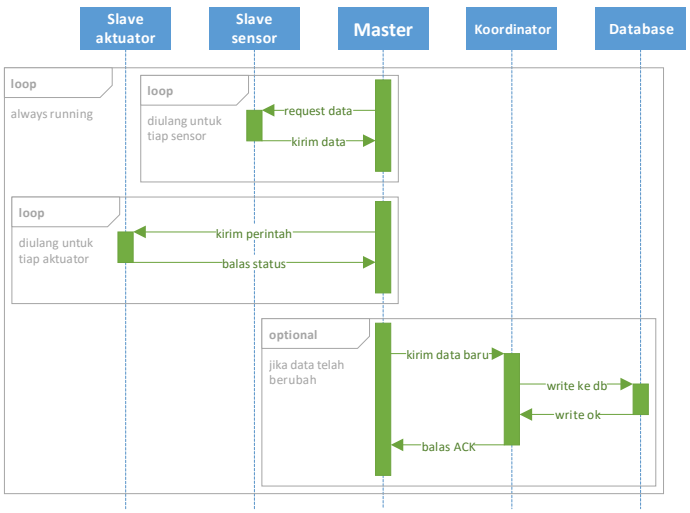
ini. *Software* yang dirancang menambahkan *data logging* untuk nilai sensor dan aktuator, desain prosedur komunikasi yang lebih efisien, dan desain kontrol otomatis meliputi perubahan *setpoint* dari input *user* dan kontrol otomatis sederhana untuk AC. Selain itu, *software* ini harus tetap andal dan sederhana seperti purwarupa penelitian di atas.

METODE PENELITIAN

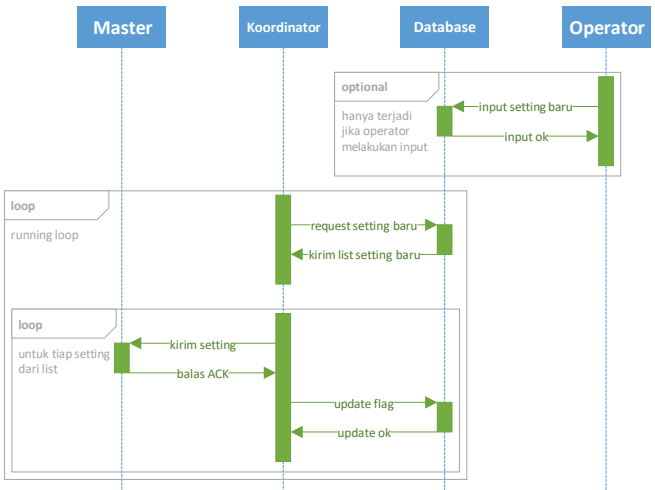
Skripsi yang dibuat oleh penulis adalah satu bagian dari sebuah sistem besar. Gambaran struktur sistem besar tersebut dapat dilihat pada Gambar 1, sedangkan alur interaksi antar komponen sistem besar dapat dilihat pada Gambar 2 dan Gambar 3.



Gambar 1: Diagram blok sistem besar



Gambar 2: *Sequence diagram* dari master ke koordinator



Gambar 3: *Sequence diagram* dari koordinator ke master

Tiap kelas memiliki satu *master node*. *Master node* ini terdiri dari sebuah kontroler master yang terhubung ke beberapa kontroler *slave* menggunakan satu jalur komunikasi serial RS485 dalam bentuk *multidrop bus*. Kontroler master juga memiliki modul AT86RF233 untuk berkomunikasi dengan koordinator. Kontroler master bertugas mengambil data dari *slave*-*slave*-nya secara periodik, mengirimkan data baru ke koordinator, menerima *setpoint* baru dari koordinator, dan juga mengatur AC dan lampu berdasarkan *setpoint*.

Kontroler *slave* yang digunakan dalam sistem besar ini ada 3 macam: sensor, aktuator lampu, dan aktuator AC. Jumlah dan jenis *slave* disesuaikan dengan kebutuhan masing-masing kelas. *Slave* jenis sensor selalu memiliki 3 sensor: cahaya, suhu, dan gerak. *Slave* aktuator lampu berisi *relay* yang digunakan untuk mengontrol lampu. *Slave* aktuator AC berisi *transmitter* inframerah yang berfungsi sebagai pengganti *remote control* untuk AC.

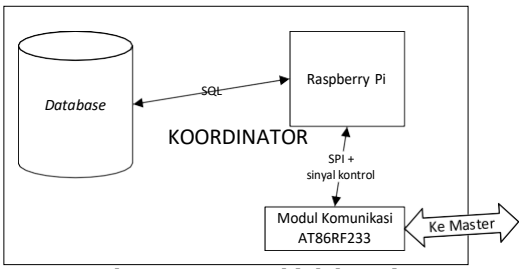
Semua kontroler master terhubung ke satu koordinator. Koordinator ini menjembatani alur data antara server dan master di tiap kelas. Koordinator ini terbuat dari sebuah Raspberry Pi yang dilengkapi dengan modul AT86RF233. Tugas koordinator adalah menerima data dari master, mencatat data tersebut ke dalam *database* di server, memproses hasil input operator menjadi *setpoint* yang bisa dikirim, serta mengirimkan *setpoint*-*setpoint* tersebut ke master yang sesuai.

Di antara master dan koordinator, terdapat jaringan *router node* yang membantu trans*port*asi *packet*. *Router node* ini terdiri dari sebuah kontroler dan

modul AT86RF233. Tugas router adalah melakukan *routing* untuk *packet-packet* yang lewat di jaringan antara master dan koordinator.

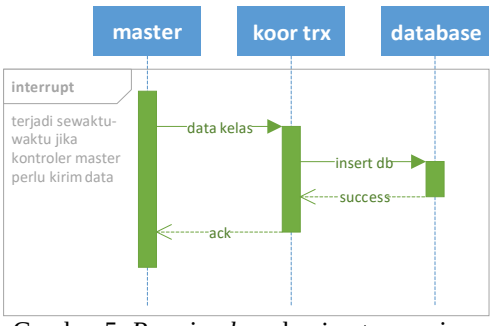
Server pada sistem ini memiliki sebuah *database*. *Database* tersebut mencatat data-data berikut: informasi kondisi kelas dari master-master yang terpasang, *setpoint* yang dibaca oleh koordinator untuk dikirim ke kelas, dan hasil input oleh operator. Server hanya bisa diakses oleh koordinator dan operator.

Fokus skripsi penulis dalam sistem besar ini adalah pengembangan *software* manajemen data yang mencakup keseluruhan *software* di koordinator, termasuk *driver* modul komunikasi AT86RF233 untuk koordinator. Tugas penulis juga termasuk desain *database* yang digunakan dalam sistem besar. *Database* tersebut juga akan terletak di koordinator karena server belum selesai ketika skripsi ini dikerjakan. Diagram blok koordinator dapat dilihat pada Gambar 4.



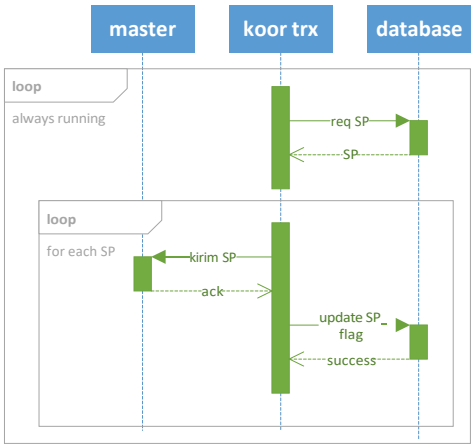
Gambar 4: Diagram blok koordinator

Untuk memenuhi tugas-tugas koordinator, rancangan *software* manajemen data pada koordinator yang dibuat pada skripsi ini akan dibagi menjadi dua bagian: bagian yang berkomunikasi dengan master (bagian *transceiver*) dan bagian yang menafsirkan input operator *(preset)* menjadi *setpoint* kontrol (bagian konversi *preset*).



Gambar 5: *Running loop* bagian *transceiver*

Software koordinator bagian *transceiver* bertugas menangani *transmit* dan *receive* menggunakan AT86RF233, termasuk akses *database* terkait *transmit* atau *receive*. Rancangan interaksi antara koordinator dengan *database* dan kontroler master dapat dilihat pada *sequence diagram* Gambar 5 dan Gambar 6.

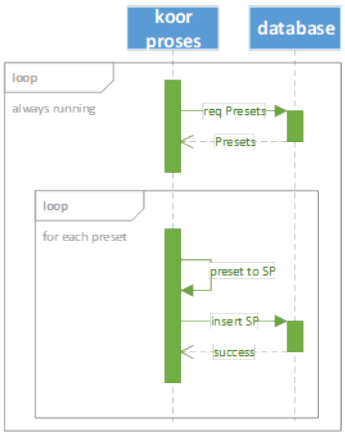


Gambar 6: *Interrupt* bagian *transceiver*

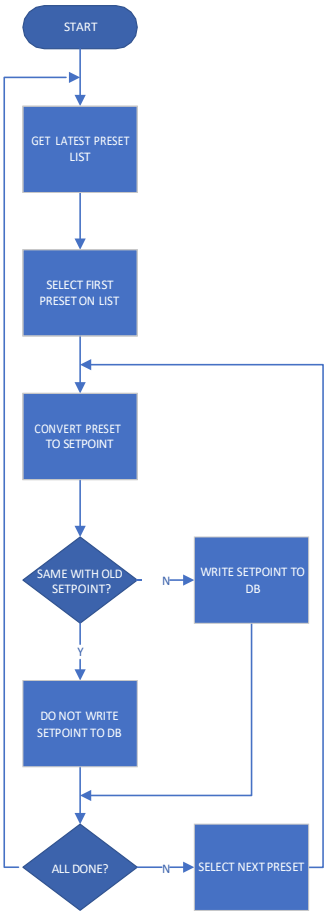
Software bagian *transceiver* akan menjalankan sebuah *loop*. Pertama, koordinator melakukan *request* daftar *setpoint* terbaru ke *database*. Untuk setiap *setpoint* pada daftar tersebut, koordinator akan melakukan pengiriman *packet setpoint* tersebut kepada master yang sesuai. Jika kontroler master tersebut membalas ACK dalam jangka waktu yang ditentukan, koordinator mengupdate *flag* pengiriman *setpoint* tersebut menjadi sudah berhasil dikirim. Proses kirim dan tunggu ACK ini diulang untuk tiap *setpoint* pada daftar tersebut. Setelah semua *setpoint* dalam daftar tersebut selesai diproses, *loop* diulang.

Loop akan disela sebentar oleh *interrupt* jika ada *packet* masuk dari kontroler master. Jika koordinator menerima *packet* dari master berupa data info kelas, data tersebut segera di-*insert* ke *database*. Setelah itu, koordinator mengirimkan balasan ACK kepada master pengirim lalu kembali menjalankan *loop* seperti biasa.

Software koordinator bagian konversi *preset* input operator menjadi *setpoint* siap kirim juga berupa *loop*. *Loop* dimulai dengan *request* daftar *preset* terbaru untuk semua kelas yang terdaftar di *database*. Tiap *preset* pada daftar tersebut akan dikonversi ke *setpoint* dan dituliskan ke tabel *setpoint*. Setelah semua *preset* diproses, *loop* ini diulang. Ilustrasi untuk alur kerja *software* ini dapat dilihat pada Gambar 7 dan Gambar 8, sedangkan detil proses konversi dilihat pada Gambar 9.



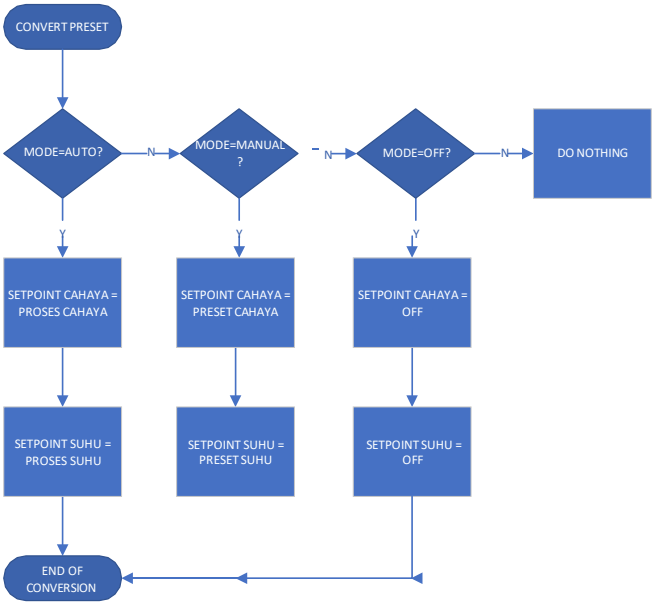
Gambar 7: *Loop* konversi *preset*



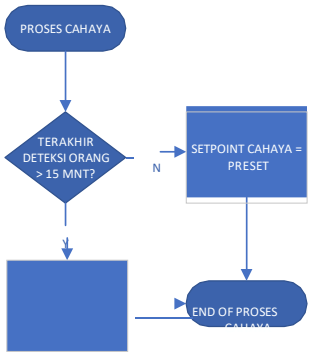
Gambar 8: *Loop* untuk *software* bagian konversi *preset*

Rancangan konversi *preset* yang dibuat menghasilkan *setpoint* berdasarkan mode *preset* yang diproses. Mode otomatis menghasilkan *setpoint* berdasarkan data sensor. Mode manual menghasilkan *setpoint* yang sama persis dengan nilai *preset*. Mode paksa *off* ini akan menghasilkan *setpoint* cahaya 0 lux (mematikan lampu)

dan kode AC *off*. Selain itu, *preset* dianggap *invalid* dan tidak diproses. Penjelasan proses otomatis untuk *setpoint* cahaya dapat dilihat pada Gambar 10, sedangkan penjelasan untuk proses *setpoint* suhu terdapat pada Gambar 11.



Gambar 9: *Flowchart* detail proses konversi *preset*



Gambar 10: *Flowchart* pemrosesan *setpoint* cahaya otomatis

Untuk pemrosesan *setpoint* otomatis, *setpoint* cahaya dihasilkan berdasarkan data sensor PIR. Jika deteksi gerak terakhir sudah lebih 15 menit, *setpoint* cahaya yang dihasilkan proses ini adalah *setpoint* mematikan lampu (0 lux). Selain itu, nilai *setpoint* cahaya menggunakan nilai *preset* dari operator. Durasi 15 menit yang dicontohkan bisa diganti sesuai kebutuhan.

Pemrosesan otomatis *setpoint* suhu lebih rumit. Jika pergantian *setpoint* terakhir belum lewat dari 5 menit, *setpoint* suhu tidak diubah. Selain itu, *setpoint* suhu akan dihitung berdasarkan deteksi sensor PIR dan suhu.

Jika tidak ada sensor PIR yang mendeteksi gerakan, bobot poin PIR adalah 0. Jika ada satu sensor yang mendeteksi gerakan, bobot poinnya satu. Selain itu, bobot poin PIR adalah dua.

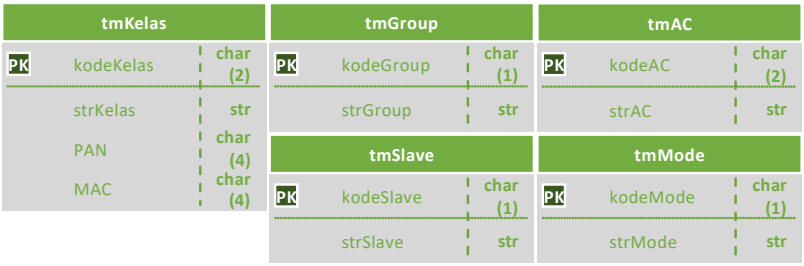
Untuk sensor suhu, nilai suhu ruang di antara 24 dan 28 derajat celsius berbobot satu poin. Suhu lebih dari *range* tersebut berbobot dua poin, sedangkan kurang dari *range* tersebut diberi bobot 0 poin.

Total poin sensor PIR dan suhu digunakan untuk menentukan *setpoint* suhu. Total poin satu atau nol menghasilkan *setpoint* 25 derajat celsius. Total poin dua menghasilkan 24 derajat, total poin tiga menghasilkan 23 derajat, dan selebihnya menghasilkan *setpoint* 22 derajat celsius.

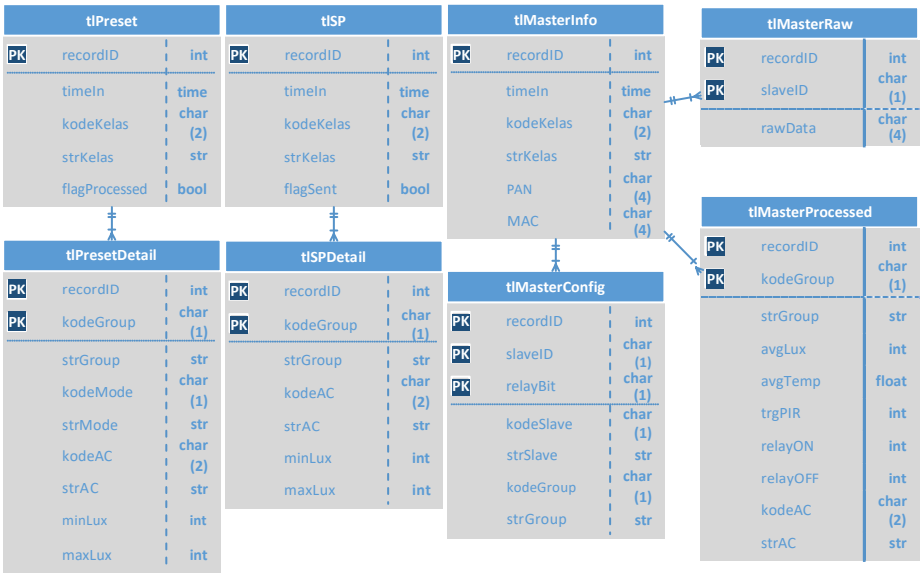
```
IF TERAKHIR GANTI SETPOINT > 5 MENIT:
    TRG=JUMLAH SENSOR PIR YANG MENDETEKSI GERAK
    IF TRG<=0:
        P_TRG=0
    ELIF TRG<=1:
        P_TRG=1
    ELSE:
        P_TRG=2
    END IF

    TMP=SUHU RUANG
    IF TMP<24:
        P_TMP=0
    ELIF TMP<=28:
        P_TMP=1
    ELSE:
        P_TMP=2
    END IF

    PP=P_TMP+P_TRG #TOTAL POINT
    IF PP<=1:
        SP_SUHU=25 C
    ELIF PP<=2:
        SP_SUHU=24 C
    ELIF PP<=3:
        SP_SUHU=23 C
    ELSE:
        SP_SUHU=22 C
    END IF
ELSE:
    SP_SUHU=SP_SUHU_LAMA
END IF
RETURN SP_SUHU
```

Gambar 11: *Pseudocode* pemrosesan otomatis *setpoint* suhu

Data yang disimpan pada *database* dibagi menjadi 3 macam: data info kelas, data *preset*, dan data *setpoint*. Data info kelas adalah data dari kelas yang dikirimkan oleh master meliputi informasi konfigurasi *slave*, status aktuator, dan nilai sensor. Bentuk data kedua yang dicatat oleh *database* adalah data *preset*. Data *preset* ini adalah data input dari operator untuk proses penentuan *setpoint*. Bentuk data ketiga adalah data *setpoint* yang mengandung kode AC, nilai lux minimum, dan nilai lux maksimum. Data *setpoint* dikirim koordinator kepada master-master di kelas. Struktur yang dibuat dapat dilihat pada Gambar 12 dan Gambar 13.



Gambar 12: Tabel-tabel master



Gambar 13: Tabel-tabel transaksi

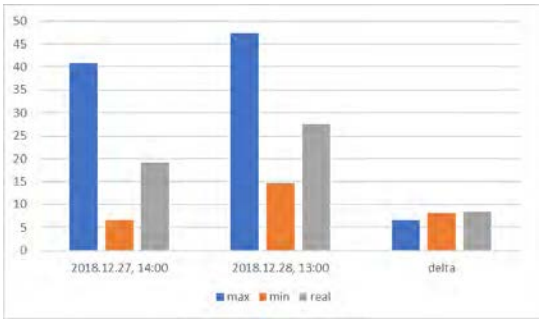
HASIL DAN PEMBAHASAN

Software manajemen data dibagi menjadi dua bagian: bagian *transceiver* dan bagian konversi *preset*. Dua kode tersebut akan berjalan bersama-sama terus-menerus saat operasi normal.

Untuk pengujian *software*, kode yang dijalankan pada koordinator adalah dua kode tersebut, ditambah kode *insert* ke *database* yang menulis satu *preset* ke *database* tiap 2 detik, dan kode *logging* persentase penggunaan CPU dan RAM dengan pencatatan tiap 0.1 detik. Satu Pi lagi bertindak sebagai imitasi kontroler master, mengirim *packet* info kelas tiap 2 detik.

Skenario yang digunakan adalah simulasi kondisi yang sangat sibuk. Pada praktiknya, beban kerja sistem tidak seberat ini. *Preset* hanya akan dimasukkan oleh operator jika perlu diganti saja. Selain itu, data info kelas hanya dikirim oleh master jika ada perubahan data saja.

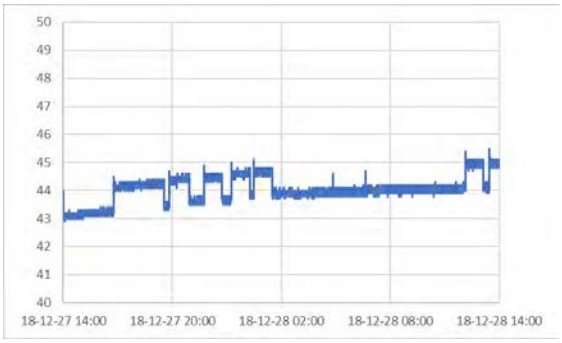
Skenario pengujian ini dimulai tanggal 27 Desember 2018 sekitar 13:00 WIB dan diakhiri tanggal 28 Desember sekitar 16:00 WIB. Data yang yang digunakan adalah data mulai 14:00 WIB tanggal 27 Desember 2018 hingga 14:00 WIB tanggal 28 Desember 2018.



Gambar 14: Rata-rata penggunaan CPU selama satu jam

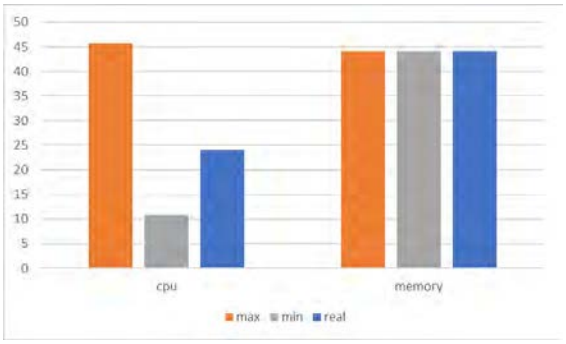
Gambar 14 menunjukkan rata-rata penggunaan CPU untuk pengukuran selama satu jam. Data di awal pengujian menunjukkan rata-rata penggunaan CPU maksimum sekitar 41%, minimum sekitar 7%, dan rata-rata penggunaan CPU secara keseluruhan sekitar 19%. Setelah satu hari, hasil pengukuran menunjukkan rata-rata penggunaan CPU maksimum sekitar 47%, minimum sekitar 14%, dan rata-rata penggunaan CPU keseluruhan sekitar 28%. Rata-rata penggunaan CPU maksimum naik 6%, minimum naik 7%, dan rata-rata keseluruhan naik 9%. Penggunaan CPU meningkat terus selama beroperasi.

Untuk penggunaaan RAM, nilai pengukuran selama 24 jam cukup stabil dan hanya naik sedikit. Selama perekaman nilai *RAM usage* terendah adalah 43%, sedangkan nilai tertinggi adalah 45%. Grafik penggunaan RAM selama 24 jam dapat dilihat pada Gambar 15.



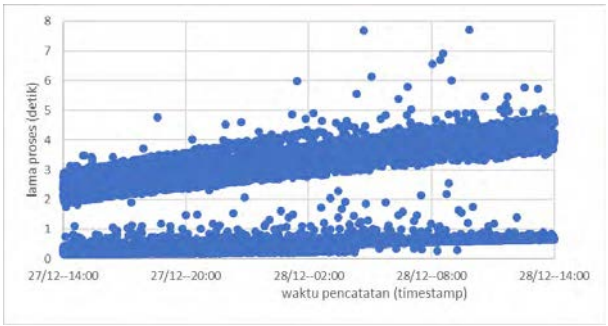
Gambar 15: Penggunaan RAM selama 24 jam

Untuk keseluruhan selama 24 jam, rata-rata penggunaan CPU maksimum adalah 46%, minimum 11%, dan rata-rata total adalah 24%. Penggunaan RAM konstan dengan rata-rata 44%. Grafik yang menunjukkan nilai-nilai tersebut ada pada Gambar 16.



Gambar 16: Rata-rata penggunaan CPU dan RAM selama 24 jam

Kode-kode yang dijalankan juga memiliki *log* yang mencatat seberapa lama kode-kode tersebut melakukan prosesnya. Kode konversi *preset* mencatat waktu yang diperlukan untuk melakukan satu kali iterasi *loop*, mulai dari *request* daftar *preset* ke *database*, memproses *preset-preset* tersebut satu per satu, memasukkan *setpoint-setpoint* baru ke *database*, dan mengupdate *flag-flag preset* yang telah diproses. Data waktu proses per iterasi yang diperlukan kode ini disajikan pada Gambar 17.

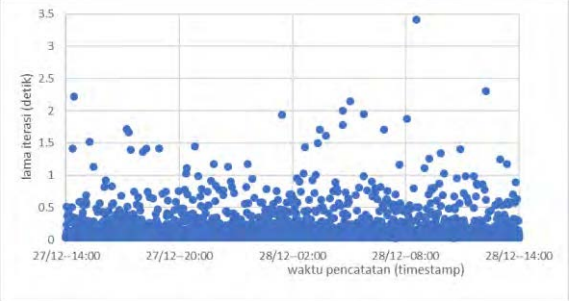


Gambar 17: Grafik durasi proses konversi *preset*

Berdasarkan Gambar 17, durasi satu iterasi proses konversi *preset* ke *setpoint* membentuk dua garis berbeda. Kode konversi *preset* ini menghabiskan waktu sekitar 1 detik per iterasi jika tidak memproses *preset* (garis bawah) namun

menghabiskan waktu sekitar 2 – 5 detik per iterasi jika ada yang perlu diproses (garis atas).

Hal yang sebaliknya dapat diamati pada proses *insert preset*. Kode *insert preset* yang digunakan pada pengujian ini adalah *loop* yang mengisi tabel *preset* setiap 2 detik sekali. Durasi satu iterasi yang dihitung oleh kode ini dimulai dari proses *insert* ke tabel tlPreset hingga proses *insert* ke tlPresetDetail selesai. Hampir semua (97.3%) iterasi proses ini diselesaikan dalam waktu kurang dari 0.1 detik. Durasi paling lama yang tercatat adalah 3.4 detik dan hanya terjadi sekali. Data untuk proses *insert* ini disajikan pada Gambar 18 dan Tabel 1.



Gambar 18: Grafik waktu iterasi proses *insert preset*

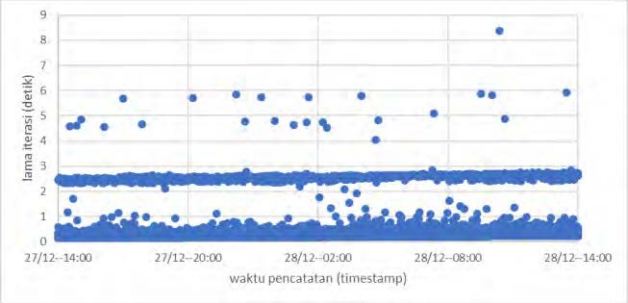
Tabel 1: Tabel frekuensi waktu iterasi proses *insert preset*

batas nilai	frekuensi	persentase
0.1	40972	97.291%
0.2	571	1.356%
0.3	224	0.532%
0.4	91	0.216%
0.5	72	0.171%
3.5	183	0.435%
total	**42113**	**100.000%**

Log untuk proses *transceiver* dibedakan menjadi dua macam: pengiriman *setpoint* dan penerimaan *packet*. Untuk pengiriman *setpoint*, runtutan prosesnya meliputi *request* daftar *setpoint* dari *database* , proses masing-masing *setpoint* tersebut ke bentuk *packet* yang siap kirim, kirim *packet* tersebut satu per satu ke tujuan yang sesuai, menunggu datangnya ACK hingga *timeout*, lalu mengupdate *flag* pengirman setpoint jika ACK diterima. Durasi satu iterasi yang dicatat pada *log* meliputi keseluruhan proses tersebut, termasuk menunggu datangnya ACK. Setelah keseluruhan proses tersebut selesai, koordinator akan diam selama satu

detik sebelum mengulang proses ini dari awal. Data *log* proses pengiriman *setpoint* ditampilkan pada Gambar 19 dan Tabel 2.

Grafik durasi untuk proses ini sekilas tampak mirip dengan grafik durasi untuk proses konversi *preset* karena membentuk dua garis. Garis atas memiliki durasi sekitar 2.5 detik, sedangkan garis bawah memiliki durasi sekitar satu detik atau kurang. Namun, dua garis tersebut tidak disebabkan oleh hal yang sama. Pada pengujian ini, lama waktu *timeout* menunggu ACK masuk adalah 2 detik.Garis atas pada grafik tersebut disebabkan karena balasan ACK tidak diterima hingga *timeout*. Menurut tabel frekuensi data, hampir 60% total iterasi diselesaikan dalam waktu kurang dari 250 milisekon, 35% dalam waktu kurang dari setengah detik.



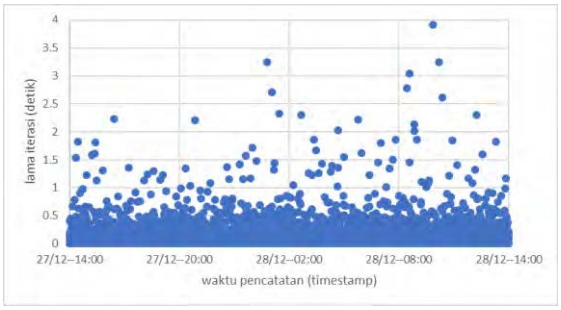
Gambar 19: Grafik waktu iterasi pengiriman *setpoint*

Tabel 2: Tabel frekuensi waktu iterasi pengiriman *setpoint*

batas nilai	frekuensi data	persentase
0.25	39360	59.836%
0.5	23344	35.488%
0.75	1751	2.662%
1	47	0.071%
1.5	16	0.024%
2	5	0.008%
2.5	433	0.658%
3	799	1.215%
9	25	0.038%
Total	**65780**	**100.000%**

Untuk proses terima *packet*, durasi yang dihitung meliputi proses *interrupt handling*, dekapsulasi *packet* yang diterima, kirim ACK balik, dan proses *write* ke *database*. *Log* hanya dicatat jika *packet* diterima dan proses dekapsulasi dilakukan, sehingga tidak sembarang *interrupt* masuk yang langsung ditulis pada log. Sebagian besar penanganan *packet* masuk diatasi dalam waktu setengah detik atau kurang. Menurut Tabel 3, 42% iterasi terselesaikan dalam waktu 0.1 detik dan 56% terjadi

dalam waktu 0.2 detik. Gambar 20 menunjukkan bahwa ada 1 iterasi yang menghabiskan waktu hampir 4 detik.



Gambar 20: Grafik waktu iterasi proses penerimaan *packet* dari master

Tabel 3: Tabel frekuensi waktu iterasi penerimaan *packet* dari master

batas nilai	frekuensi	persentase
0.1	28760	42.210%
0.2	38007	55.781%
0.3	706	1.036%
0.4	261	0.383%
0.5	123	0.181%
4	279	0.409%
Total	**68136**	100.000%

Pengujian berikutnya berupa gabungan keseluruhan *sistem energy awareness* berbasis WSAN. *Software* manajemen data yang dibuat penulis diuji bersama dengan kontroler master dan *slave* buatan rekan Ivan Taufan. Transmisi antara master dan koordinator dilakukan menggunakan *transceiver* AT86RF233 *custom* buatan Randi Eriko. Pengujian dilakukan di ruang TF21 Gedung TF Fakultas Teknik Universitas Surabaya.

Pada skenario ini, ruang kelas dipasangi sebuah master yang terhubung dengan 5 *slave*: dua *slave* sensor, sebuah *slave* relay, dan dua *slave* kontroler AC. Wilayah kontrol ruangan ini dibagi menjadi dua. Bagian belakang ruang disebut sebagai grup A, bagian depan ruang disebut sebagai grup B. Masing-masing memiliki satu *slave* sensor dan satu *slave* kontroler AC. Karena *slave relay* yang didesain oleh rekan Ivan Taufan memiliki 2 unit *relay*, satu unit *relay* tersebut digunakan untuk grup A dan satu yang lain digunakan untuk grup B. Koordinator diletakkan pada ruang kelas TF22 karena tidak ada komponen *router*. Skenario ini dilaksanakan pada tanggal 17 Desember 2018 sekitar pukul 20:00 WIB

Preset yang digunakan dalam pengujian menggunakan mode otomatis untuk grup A dan grup B. *Preset* tersebut dapat dilihat pada Tabel 4. Pada tabel tersebut, recordID yang menunjukkan *preset* yang dipakai adalah recordID 148. Data recordID 147 tentu tidak akan dipakai karena lebih tua daripada 148, recordID 198 pada waktu malam pengujian masih belum ada.

Tabel 4: *Preset* yang digunakan selama uji integrasi sistem besar

recordID	kodeGroup	kodeMode	kodeAC	minLux	maxLux	timeIn	kodeKelas
00000000000000000147	0	O	80	300	1000	2018-12-13 00:25:48	21
00000000000000000147	1	O	40	300	1000	2018-12-13 00:25:48	21
00000000000000000147	2	O	20	1000	8000	2018-12-13 00:25:48	21
00000000000000000147	3	O	10	0	1000	2018-12-13 00:25:48	21
00000000000000000148	0	A	80	300	1000	2018-12-14 19:51:05	21
00000000000000000148	1	A	40	300	1000	2018-12-14 19:51:05	21
00000000000000000148	2	O	20	1000	8000	2018-12-14 19:51:05	21
00000000000000000148	3	O	10	0	1000	2018-12-14 19:51:05	21
00000000000000000198	0	A	80	300	1000	2018-12-18 06:58:39	21
00000000000000000198	1	A	40	300	1000	2018-12-18 06:58:39	21
00000000000000000198	2	O	20	1000	8000	2018-12-18 06:58:39	21
00000000000000000198	3	O	10	0	1000	2018-12-18 06:58:39	21

Tabel 5: Data kelas TF21 grup A dari master

timeIn	kodeKelas	recordID	kodeGroup	avgLux	avgTemp	trgPIR	relayON	relayOFF	kodeAC
2018-12-17 20:02:14	21	00000000000000019881	0	464	31.7383	0	0	1	CC
2018-12-17 20:02:41	21	00000000000000019884	0	0	31.25	0	0	1	CC
2018-12-17 20:02:45	21	00000000000000019885	0	0	31.7383	0	0	1	CC
2018-12-17 20:03:01	21	00000000000000019887	0	0	31.7383	0	0	1	CC
2018-12-17 20:03:05	21	00000000000000019888	0	0	31.25	0	0	1	CC
2018-12-17 20:04:10	21	00000000000000019900	0	0	31.25	0	0	1	CC
2018-12-17 20:04:36	21	00000000000000019905	0	0	31.7383	0	0	1	CC
2018-12-17 20:04:40	21	00000000000000019906	0	0	31.25	0	0	1	CC
2018-12-17 20:05:02	21	00000000000000019911	0	0	31.7383	0	0	1	CC
2018-12-17 20:05:46	21	00000000000000019920	0	0	31.7383	0	0	1	CC
2018-12-17 20:05:50	21	00000000000000019922	0	0	31.25	0	0	1	CC
2018-12-17 20:05:53	21	00000000000000019924	0	0	31.7383	0	0	1	CC
2018-12-17 20:06:19	21	00000000000000019930	0	0	31.25	0	0	1	CC
2018-12-17 20:08:40	21	00000000000000019952	0	0	31.25	0	0	1	CC
2018-12-17 20:10:51	21	00000000000000019975	0	0	31.25	1	0	1	CC
2018-12-17 20:10:54	21	00000000000000019976	0	0	31.25	1	0	1	CC
2018-12-17 20:11:12	21	00000000000000019979	0	0	31.7383	1	0	1	CC
2018-12-17 20:11:16	21	00000000000000019980	0	0	31.25	1	0	1	CC
2018-12-17 20:11:28	21	00000000000000019983	0	0	31.25	0	0	1	CC
2018-12-17 20:11:35	21	00000000000000019984	0	0	31.25	0	1	0	CC
2018-12-17 20:11:42	21	00000000000000019985	0	480	31.25	1	1	0	CC
2018-12-17 20:11:45	21	00000000000000019987	0	472	31.25	0	1	0	CC
2018-12-17 20:11:56	21	00000000000000019988	0	480	31.25	1	1	0	CC
2018-12-17 20:11:59	21	00000000000000019989	0	0	31.25	0	1	0	CC
2018-12-17 20:16:18	21	00000000000000020038	0	0	31.7383	0	1	0	CC
2018-12-17 20:18:46	21	00000000000000020063	0	0	31.25	0	1	0	CC
2018-12-17 20:21:53	21	00000000000000020090	0	0	31.25	0	1	0	CC

Dari Tabel 5, diketahui bahwa tak lama setelah PIR mendeteksi keberadaan orang di dalam ruangan (recordID 19975, pukul 20:10:51 WIB) lampu pada grup tersebut berubah dari kondisi mati ke kondisi menyala (recordID 19984, pukul 20:11:35 WIB). Kejadian ini menunjukkan bahwa koordinator mengirimkan *packet setpoint* baru ke kontroler master setelah mendeteksi perubahan nilai sensor PIR tersebut. Hal tersebut terbukti pada Tabel 6 yang menunjukkan data *setpoint* di sekitar waktu itu. Jika dilihat pada tabel tersebut, isian dengan recordID 936 pada pukul 20:11:31 menunjukkan bahwa *setpoint* cahaya yang dikirim untuk grup A

telah berubah dari *setpoint* minimum 0 lux dan maksimum 0 lux menjadi minimum 300 lux dan maksimum 1000 lux, persis dengan aturan *preset* untuk grup tersebut.

Tabel 6: Perubahan *setpoint* dari *off* menjadi sesuai *preset*

recordid	kodegroup	kodeac	minlux	maxlux	kodekelas	timein	flagsent
00000000000000000930	0	10	0	0	21	2018-12-17 20:02:07	1
00000000000000000934	0	80	0	0	21	2018-12-17 20:06:26	1
00000000000000000936	0	02	300	1000	21	2018-12-17 20:11:31	1
00000000000000000937	0	10	300	1000	21	2018-12-17 20:12:33	1
00000000000000000938	0	80	300	1000	21	2018-12-17 20:13:32	1

Tak lama kemudian, lampu ruang kelas TF21 mati karena sudah 15 menit berlalu tanpa orang di ruangan tersebut. Tabel 7 menunjukkan bahwa deteksi orang oleh sensor PIR yang terakhir adalah pukul 20:11:56 (recordID 19988). Relay pada grup tersebut berubah dari kondisi menyala menjadi kondisi mati pada pukul 20:27:50. Bila dilihat pada Tabel 8, memang tercatat bahwa pada recordID 940 pukul 20:27:48 terjadi perubahan setpoint cahaya menjadi minimum 0 lux dan maksimum 0 lux.

Tabel 7: Data kelas TF21 grup A, pukul 20:27:50 lampu mati

timeIn	kodeKelas	recordID	kodeGroup	avgLux	avgTemp	trgPIR	relayON	relayOFF	kodeAC
2018-12-17 20:11:28	21	00000000000000019983	0	0	31.25	0	0	1	CC
2018-12-17 20:11:35	21	00000000000000019984	0	0	31.25	0	1	0	CC
2018-12-17 20:11:42	21	00000000000000019985	0	480	31.25	1	1	0	CC
2018-12-17 20:11:45	21	00000000000000019987	0	472	31.25	0	1	0	CC
2018-12-17 20:11:56	21	00000000000000019988	0	480	31.25	1	1	0	CC
2018-12-17 20:11:59	21	00000000000000019989	0	0	31.25	0	1	0	CC
2018-12-17 20:16:18	21	00000000000000020038	0	0	31.7383	0	1	0	CC
2018-12-17 20:18:46	21	00000000000000020063	0	0	31.25	0	1	0	CC
2018-12-17 20:21:53	21	00000000000000020090	0	0	31.25	0	1	0	CC
2018-12-17 20:22:01	21	00000000000000020091	0	0	31.7383	0	1	0	CC
2018-12-17 20:22:21	21	00000000000000020096	0	0	31.7383	0	1	0	CC
2018-12-17 20:22:25	21	00000000000000020097	0	0	31.25	0	1	0	CC
2018-12-17 20:22:41	21	00000000000000020100	0	0	31.25	0	1	0	CC
2018-12-17 20:22:45	21	00000000000000020101	0	0	31.25	0	1	0	CC
2018-12-17 20:23:54	21	00000000000000020111	0	0	31.7383	0	1	0	CC
2018-12-17 20:24:42	21	00000000000000020121	0	0	31.25	0	1	0	CC
2018-12-17 20:25:14	21	00000000000000020127	0	0	31.7383	0	1	0	CC
2018-12-17 20:25:22	21	00000000000000020129	0	0	31.25	0	1	0	CC
2018-12-17 20:25:32	21	00000000000000020132	0	0	31.25	0	1	0	CC
2018-12-17 20:26:43	21	00000000000000020144	0	0	31.25	0	1	0	CC
2018-12-17 20:27:04	21	00000000000000020148	0	0	31.25	0	1	0	CC
2018-12-17 20:27:35	21	00000000000000020153	0	0	31.7383	0	1	0	CC
2018-12-17 20:27:39	21	00000000000000020154	0	0	31.25	0	1	0	CC
2018-12-17 20:27:50	21	00000000000000020157	0	0	31.25	0	0	1	CC
2018-12-17 20:27:59	21	00000000000000020159	0	0	31.25	0	0	1	CC
2018-12-17 20:28:25	21	00000000000000020164	0	0	31.7383	0	0	1	CC
2018-12-17 20:28:32	21	00000000000000020165	0	0	31.25	0	0	1	CC
2018-12-17 20:28:52	21	00000000000000020169	0	0	31.25	0	0	1	CC
2018-12-17 20:29:02	21	00000000000000020172	0	0	31.7383	0	0	1	CC
2018-12-17 20:29:21	21	00000000000000020175	0	0	31.25	0	0	1	CC
2018-12-17 20:29:26	21	00000000000000020177	0	0	31.7383	0	0	1	CC
2018-12-17 20:29:35	21	00000000000000020178	0	0	31.25	0	0	1	CC
2018-12-17 20:29:48	21	00000000000000020182	0	0	31.7383	0	0	1	CC
2018-12-17 20:29:53	21	00000000000000020183	0	0	31.25	0	0	1	CC
2018-12-17 20:30:04	21	00000000000000020186	0	0	31.25	0	0	1	CC

Tabel 8: Setpoint TF21 grup A, lampu dimatikan pukul 20:27:48

recordid	kodegroup	kodeac	minlux	maxlux	kodekelas	timein	flagsent
00000000000000000936	0	02	300	1000	21	2018-12-17 20:11:31	1
00000000000000000937	0	10	300	1000	21	2018-12-17 20:12:33	1
00000000000000000938	0	80	300	1000	21	2018-12-17 20:13:32	1
00000000000000000939	0	10	300	1000	21	2018-12-17 20:18:40	1
00000000000000000940	0	10	0	0	21	2018-12-17 20:27:48	1
00000000000000000943	0	20	0	0	21	2018-12-18 03:11:07	1
00000000000000000944	0	10	0	0	21	2018-12-18 03:19:14	1
00000000000000000948	0	80	0	0	21	2018-12-18 06:58:33	1

Pengujian ini membuktikan bahwa *software* manajemen data yang telah dibuat pada skripsi ini telah melaksanakan tugasnya dengan baik. Pada pengujian ini, masing-masing bagian kode yang dibuat terbukti berfungsi. Kode akses *database* terbukti berfungsi dari isi *database* yang ditampilkan pada penggalan-penggalan tabel yang dibahas di subbab ini. Kode *transceiver* terbukti berfungsi dari adanya data *packet* info kelas yang masuk dan tercatat pada tabel info kelas. Kode konversi *preset* beserta algoritma kontrol otomatis yang terdapat di dalamnya juga terbukti berfungsi dengan munculnya *setpoint-setpoint* baru di *database* meskipun *preset* terbaru saat pengujian tercatat 3 hari sebelumnya. Pengujian ini juga membuktikan bahwa *software* manajemen data yang dibuat tak hanya berfungsi dengan baik ketika disimulasikan, tetapi juga mampu diintegrasikan dengan komponen-komponen sistem besar lainnya.

KESIMPULAN DAN SARAN

Software manajemen data pada koordinator yang dibuat pada skripsi ini berfungsi dengan baik. Semua bagian kode *software* manajemen data telah dibuktikan bisa berfungsi dengan baik pada kondisi simulasi maupun kondisi nyata di lapangan. Integrasi *software* manajemen data dengan sistem besar berhasil.

Penggunaan RAM selama pengujian 24 jam hampir konstan 44%. Penggunaan CPU rata-rata selama pengujian 24 jam adalah 24% secara keseluruhan. Penggunaan rata-rata per jam CPU mengalami kenaikan 9% dalam waktu operasi 24 jam, dari rata-rata 19% pada jam pertama menjadi rata-rata 28% pada jam ke-24.

Implementasi sistem WSAN gabungan yang dibuat dari skripsi penulis dan rekan-rekan tidak termasuk *routing*. Jarak koordinator dan master-masternya tidak bisa terlalu jauh sehingga ketika pengujian sistem besar. Saran untuk penerus adalah mengembangkan protokol komunikasi antara koordinator dan master yang mencakup proses *routing packet* untuk memperbesar jangkauan jaringan sistem.

Model *database* yang digunakan penulis *(relational database)* kurang tepat. Saran dari penulis adalah mengembangkan *database* yang modelnya *document-oriented*, misalnya MongoDB.

DAFTAR PUSTAKA

[1] "World Energy Balances 2018: Overview," International Energy Agency, 2018.

[2] Prof. Dr. Osama Ahmed Ibrahim Masood, Dr. Eng. Mohamed Ibrahim Abd Al-Hady and Ahmed Khamies Mohamed Ali, "Applying the Principles of Green Architecture for Saving Energy in Buildings," *Energy Procedia,* vol. 115, pp. 369-382, Juni 2017.

[3] Luay N. Dwaikat and Kherun N. Ali, "The Economic Benefits of a Green Building - Evidence from Malaysia," *Journal of Building Engineering,* vol. 18, pp. 448-453, 2018.

[4] Timilehin Labeodan, Christel de Bakker, Alexander Rosemann and Wim Zeiler, "On the Application of Wireless Sensors and Actuators Network in Existing Buildings for Occupancy Detection and Occupancy-driven Lighting Control," *Energy and Buildings,* vol. 127, pp. 75-83, 2016.

[5] Henry Hermawan, Edward Oesnawi and Albert Darmaliputra, "A Reliable, Low-Cost, and Low-Power Base Platform for Energy Management System," in *Proceedings of Second International Conference on Electrical Systems, Technology and Information 2015 (ICESTI 2015)*, Singapore, 2016.