Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 8, Juni 2017, hlm. 697-706 http://j-ptiik.ub.ac.id

Pengembangan Arduino *Code Builder* Pada Sistem Simulator *Smart Home* Dengan *Rule* Terdistribusi

Hery Julianto Situmorang[1], Edita Rosana Widasari[2], Sabriansyah Rizqika AKbar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]herisitumorang7@gmail.com, [2]editarosanaw@ub.ac.id, [3]sabrian@ub.ac.id

Abstrak

Penelitian dibidang rumah cerdas (*smart home*) pada saat ini sangat banyak dilakukan. Namun dalam perancangan dan implementasiannya, membutuhkan banyak perangkat keras dan memerlukan waktu yang cukup lama. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan simulator *smart home* berbasis aplikasi *desktop* sehingga dapat menghemat biaya dan waktu karena tidak perlu membeli perangkat keras dalam pengembangan suatu *smart home*. Sistem ini terdiri dari dua aplikasi yaitu aplikasi kontrol dan monitoring serta aplikasi *object* simulator. Pada aplikasi kontrol dan monitoring menampilkan informasi pengguna, daftar objek dan informasi *rule*. Pada sistem ini akan diimplementasikan metode *rule* terdistribusi dalam menghidupkan dan mematikan objek. Objek yang digunakan pada penelitian ini adalah sensor PIR dan lampu, *rule* berisi aturan-aturan lampu yang akan dinyalakan apabila sensor PIR mendeteksi keberadaan manusia. Pada sistem ini juga menghasilkan keluaran berupa kode program untuk *node* sensor PIR dan *node* lampu yang bertujuan untuk memvalidasi penggunaan *rule terdistribusi* pada perangkat keras. Berdasarkan hasil pengujian didapatkan bahwa *object* simulator dapat menerima *rule* dari aplikasi kontrol dan monitoring dengan tingkat keberhasilan 100% dari 28 kali pengiriman *rule* dan r*ule* juga dapat dieksekusi dengan tingkat keberhasilan 100% dari 28 kali percobaan. Aplikasi juga dapat menampilkan kode program sesuai dengan *rule* yang dipilih oleh pengguna.

Kata Kunci: *smart home, eksekusi rule terdistribusi, delphi,sensor PIR, komunikasi socket*

Abstract

Nowadays, there are numbers of researches on Smart Home that have been done by technicians even though they need a lot of relevant hardware and periods of time to design plan and ways of implementation for the Smart Home. Therefore, this research conducted with focus on developing Smart Home simulator based on desktop application which can improve the time and hardware needed since it enables the Smart Home to be used without utilizing lots of hardware. The system consists of two applications which are (i) Control and Monitoring application and (ii) Object Simulator application. The Control and Monitoring application provides user's information, list of object and information of rule. In this system, the method of distributed rule will be implemented to turn on and turn off the objects while the objects for this present research are PIR sensor and lamps. Rule contains patterns for which lamps that are going to on if the PIR sensor detects the human existence. This system also produces program code for PIR sensor and light nodes that aim to validate the use of distributed rules on hardware. The result of this research shows that simulator object can receive rule from Control and Monitoring application with the success rate of 100% out of 28 times rule transmissions and also can be executed with the success rate of 100% out of 28 times tests. The application can also display the program code according to the rule chosen by the user.

Keywords: *smart home, distributed rule execution, delphi, PIR sensor, socket communication*

1. PENDAHULUAN

Pemanasan global adalah salah satu masalah yang dihadapi oleh negara-negara di dunia, salah satu faktor penyebab pemanasan global adalah penggunaan energi listrik yang sangat boros. Sebagian besar energi listrik yang kita gunakan berasal dari pembakaran batu bara dan minyak

bumi yang hasil pembakarannya akan menghasilkan karbon dioksida. Perkembangan teknologi juga mempengaruhi penggunaan listrik, semakin banyaknya barang-barang elektronik yang digunakan juga mempengaruhi penggunaan listrik. Pemborosan daya listrik sering terjadi karena penggunanya yang tidak bijak menggunakannya, misalnya lupa mematikan lampu pada siang hari, lupa mematikan lampu pada saat meninggalkan rumah dan tidak mencabut steker pada saat tidak digunakan lagi. Salah satu cara untuk mengurangi pemborosan energi listrik adalah dengan menggunakan teknologi rumah cerdas (*smart home*).

Teknologi rumah cerdas (*smart home*) adalah sistem aplikasi yang merupakan gabungan antara teknologi dan pelayanan yang dikhususkan pada lingkungan rumah dengan fungsi tertentu yang bertujuan meningkatkan efesiensi, kenyamanan dan keamanan penghuninya. Sistem rumah cerdas biasanya terdiri dari perangkat kontrol, monitoring dan otomatisasi beberapa perangkat atau peralatan rumah yang dapat diakses melalui sebuah komputer (Yunarman & Azman, 2009). Dengan fitur otomatisasi pada *smart home* dapat membantu menurunkan pemborosan energi listrik pada suatu rumah.

Salah satu fitur *smart home* adalah lampu yang hemat daya. Lampu tersebut mampu menyala dan mati sesuai perintah dari pusat berdasarkan kondisi data *node-node* lain yang diperoleh pusat, sehingga membutuhkan waktu yang relatif lama dan daya yang relatif besar. Untuk mengatasi hal tersebut, maka *smart home* yang menggunakan *wireless sensor node* dapat menggunakan suatu proses eksekusi *rule* yang terdistribusi. Pada proses eksekusi *rule* terdistribusi sensor dan aktuator akan berkomunikasi secara *node to node* untuk mengeksekusi *rule* yang dikirim dari pusat dan disimpan di *node* sensor lalu mengirimkan hasil eksekusi ke pusat.

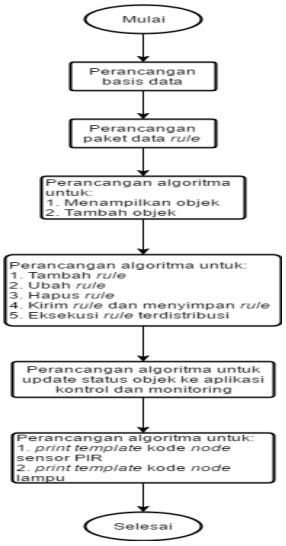
Pada penelitian sebelumnya yang berjudul "*Smart Home* Menggunakan *Wireless Sensor Network* Dengan Proses Eksekusi *Rule* Terdistribusi: Penerapan Pada Sensor PIR dan Lampu Di Dalam Ruangan" merancang sebuah sistem yang akan menyalakan lampu jika sensor PIR mendeteksi keberadaan manusia. Lampu-lampu yang akan dinyalakan disimpan ke dalam *rule* yang bisa dituliskan, disimpan dan dieksekusi oleh masing-masing modul sensor PIR secara terdistribusi. Proses penulisan *rule* dilakukan secara nirkabel dari *base station* (BS) ke masing-masing sensor PIR. Proses eksekusi secara terdistribusi dilakukan oleh masing-masing sensor PIR yang mengirim data secara nirkabel ke tiap lampu sesuai dengan *rule* masing-masing tanpa melalui *base station*. Kelemahan pada penelitian ini adalah dibutuhkan banyak perangkat keras yang harus dibeli, sehingga membutuhkan banyak biaya. Pada penelitian sebelumnya telah dibuat sebuah sistem *centralized smart home* yang dapat membantu pengguna memantau penggunaan listrik peralatan rumah dan mematikan secara otomatis apabila tidak digunakan. Selain itu sistem *centralized smart home* dapat digunakan dengan *mode* manual dan dikontrol dengan menggunakan *smart phone* yang terhubung dengan internet. Tetapi pada sistem yang sebelumnya fitur yang ada masih terbatas. (Setyawan, Setiabudi, Muttaqin, & Ichsan, 2016)

Oleh karena itu dalam penelitian ini, akan dirancang sebuah sistem simulator *centralized* pada *smart home* untuk memvalidasi penelitian sebelumnya yang telah menerapkan proses eksekusi *rule* terdistribusi pada perangkat keras, apakah *rule* terdistribusi dapat juga diterapkan pada perangkat lunak dengan cara memberikan keluaran berupa kode program untuk *node* sensor PIR dan *node* lampu yang siap digunakan pada arduino pro mini. Dengan dikembangkannya aplikasi *centralized smart home* simulator ini nantinya dapat digunakan untuk pengembangan algoritma dan fitur-fitur yang akan diaplikasikan di *smart home*.

2. PERANCANGAN DAN IMPLEMENTASI

2.1 Perancangan Perangkat Lunak

Terdapat dua tahap perancangan perangkat lunak dari sistem Smart Home Centralized Simulator yaitu perancangan aplikasi kontrol dan monitoring dan aplikasi objek simulator. Gambar 1 menunjukkan diagram alir perancangan perangkat lunak.



Gambar 1 Diagram alir perancangan perangkat lunak

2.2 Perancangan Paket Data *Rule*

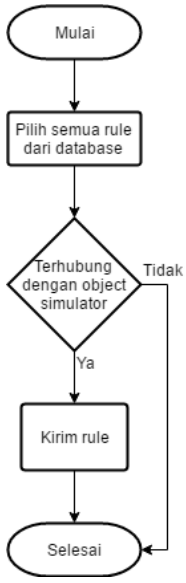
Paket data yang dimaksud dalam penelitian ini adalah *rule* yang dikirim dari aplikasi kontrol dan monitoring ke *object* simulator. Dengan adanya perancangan paket data, diharapkan *rule* yang dieksekusi tepat pada objek yang dituju. Tabel 1 menunjukkan keterangan paket data *rule*.

Tabel 1 Keterangan paket data rule

No	Nama Slot	Keterangan
1	@	Header_ untuk ID *rule*
2	RULE_ID	ID *rule*
3	#	Header untuk objek yang berperan sebagai input
4	RULE_OBJ_IN	ID objek yang berperan sebagai *input*
5	^	*Header* untuk nama objek yang berperan sebagai *input*
6	SObj_In_Name	Nama objek yang berperan sebagai *input*
7	$	*Header* untuk status objek yang berperan sebagai *input*
8	RULE_OBJ_IN_STATUS	Status objek yang berperan sebagai *input*
9	%	*Header* untuk ID objek yang berperan sebagai *output*
10	RULE_OBJ_OUT	ID objek yang berperan sebagai *output*
11	+	*Header* untuk nama objek yang berperan sebagai *output*
12	SObj_Out_Name	Nama objek yang berperan sebagai *output*
13	*	*Header* untuk status objek yang berperan sebagai *output*
14	RULE_OBJ_OUT_STATUS	Status objek yang berperan sebagai *output*
15	&	*Header* untuk ID user pemilik *rule*
16	RULE_USR_ID	ID *user* pemilik *rule*

2.3 Perancangan Algoritma Kirim dan Simpan *Rule*

Fungsi kirim *rule* digunakan untuk mengirim *rule* dari aplikasi kontrol dan monitoring ke aplikasi objek simulator. *Rule* harus dikirim ke objek simulator agar objek dapat melakukan eksekusi *rule*. Gambar *2* menunjukkan rancangan algoritma untuk mengirim *rule*.

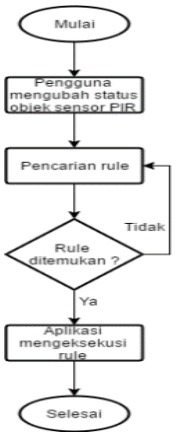


Gambar *2* Diagram alir kirim *rule*

Sebelum mengirim *rule* pengguna harus memastikan aplikasi kontrol dan monitoring telah terhubung dengan aplikasi objek monitoring. Untuk menghubungkannya dapat melalui *menu Object* Simulator IP *Address*, masukkan alamat IP dan *port* yang digunakan oleh objek simulator yaitu *localhost* dan 16209

setelah itu klik tombol *Connect*. Jika sudah terhubung pengguna dapat mengirim *rule*, *rule* yang diterima oleh objek simulator akan disimpan kedalam *file* konfigurasi bernama Rule.ini.

2.4 Perancangan Algoritma Eksekusi *Rule* Terdistribusi



Gambar 3 Diagram alir eksekusi *rule* terdistribusi

Pengguna dapat melakukan eksekusi *rule* melalui aplikasi objek simulator. Pengguna akan mengubah status objek sensor PIR, setiap ada perubahan pada status objek PIR aplikasi akan mencari apakah ada *rule* yang sesuai dengan status objek PIR yang terbaru. Apabila *rule* ditemukan aplikasi akan mengeksekusi perintah yang sesuai dengan *rule*. Gambar 3 menunjukkan rancangan algoritma eksekusi *rule*.

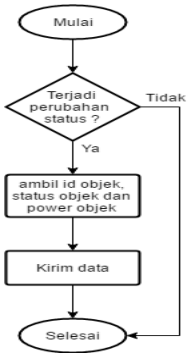
2.5 Perancangan Algoritma *Update* Status Objek ke *Base Station*

Setiap terjadi perubahan status pada objek maka objek tersebut akan mengirim status terbarunya pada aplikasi kontrol dan monitoring. Fitur ini digunakan untuk memenuhi kebutuhan monitoring dalam sistem ini. Data status dikirim dengan menggunakan format yang telah ditentukan. Tabel 2 menunjukkan keterangan pengiriman data objek

Tabel 2 Keterangan format pengiriman data objek

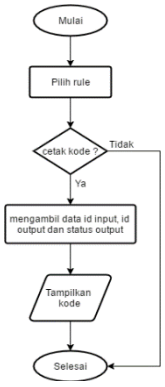
No	Nama	Keterangan
1	!	Header untuk ID objek pengirim
2	sRule_Id_Obj_Out	ID objek pengirim
3	@	Header untuk status terbaru objek
4	sRule_Obj_Out_Staus	Status terbaru objek
5	#	Header untuk daya objek
6	IntToStr(iPower)	Daya objek

Gambar 4 menunjukkan setiap terjadi perubahan status maka aplikasi *object* simulator akan mengirim status terbaru objek ke aplikasi kontrol dan monitoring.



Gambar 4 Diagram alir *update* status objek

2.6 Perancangan Algoritma *Print* Kode Program *Node* Sensor PIR



Gambar 5 Diagram alir *print* kode *node* sensor PIR

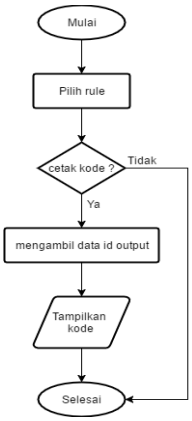
Fungsi *print template* kode untuk *node* sensor PIR adalah untuk memudahkan pengembang dalam membuat kode program apabila ingin mengaplikasikan *rule* terdistribusi pada perangkat keras *node* sensor PIR, sehingga *node* sensor PIR dapat menerima *rule* dari aplikasi kontrol dan monitoring, menyimpan *rule* dan mengirim perintah ke *node* lampu. Pengembang tidak perlu membuat kode program dari awal tetapi dapat mencetak kode dari aplikasi kontrol dan monitoring Untuk mencetak kode program yang akan digunakan pada *node* sensor PIR, pengguna harus memilih *rule* yang akan digunakan pada *node* sensor PIR yang ada pada aplikasi kontrol dan monitoring. Setelah itu pengguna mengklik tombol Print

Kode PIR kemudian aplikasi akan menampilkan kode program yang siap digunakan pada *node* sensor PIR. Gambar 5 menunjukkan diagram alir untuk *print* kode *node* sensor PIR.

2.7 Perancangan Algoritma *Print* Kode Program *Node* Lampu

Fungsi *print template* kode untuk *node* lampu adalah untuk memudahkan pengembang dalam membuat kode program sehingga node lampu dapat menerima perintah dari node sensor PIR dan dapat mengirim status terbaru *node* lampu ke aplikasi kontrol dan monitoring. Pengembang tidak perlu membuat kode program dari awal tetapi dapat mencetak kode dari aplikasi kontrol dan monitoring.

Untuk mencetak kode program yang akan digunakan pada *node* lampu, pengguna harus memilih *rule* yang akan digunakan pada *node* sensor PIR yang ada pada aplikasi kontrol dan monitoring. Setelah itu pengguna mengklik tombol *Print* Kode Lampu kemudian aplikasi akan menampilkan kode program yang siap digunakan pada *node* lampu. Gambar 6 menunjukkan diagram alir untuk *print* kode *node* lampu.



Gambar 6 Diagram alir *print* kode *node* lampu

2.8 Implementasi Algoritma Kirim *Rule* dan Simpan *Rule*

Fitur kirim *rule* digunakan untuk mengirim *rule* dari aplikasi kontrol dan monitoring ke aplikasi *object* simulator. *Rule* harus dikirim agar dapat dieksekusi oleh objek.

Objek simulator akan menerima data *rule* yang dikirim tersebut dan menyimpannya kedalam *file* Rule.ini dalam bentuk format yang telah ditentukan. Tabel 3 menunjukkan contoh penyimpanan *rule* pada file *Rule*.ini.

Tabel 3 Contoh penyimpanan *rule*

Section	Key	Value
!PIR_1_ON@10	*Rule*_ID	10
	*Rule*_Obj_In	10
	*Rule*_Obj_In_Name	PIR_1
	*Rule*_Obj_In_Status	100
	*Rule*_Obj_Out	1
	*Rule*_Obj_out_Name	Lampu_1
	*Rule*_Obj_Out_Status	100
	*Rule*_User_Id	1

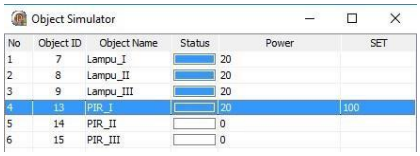
Keterangan:

1. Nama *section* adalah !PIR_1_ON@10. Karakter "!" adalah *header* untuk nama objek yang berperan sebagai *input*. "PIR_1" adalah nama objek, karakter "_" adalah *header* untuk status objek, "ON" adalah status objek, karakter "@" adalah *header* untuk ID *rule* dan angka "10" adalah ID *rule*.
2. *Rule*_ID adalah informasi ID *rule.*
3. *Rule*_Obj_In adalah ID objek yang berperan sebagai *input.*
4. *Rule*_Obj_In_Name adalah nama objek yang berperan sebagai *input*.
5. *Rule*_Obj_In_Status adalah status objek yang berperan sebagai *input*, 100 untuk ON dan 0 untuk OFF.
6. *Rule*_Obj_Out adalah ID objek yang berperan sebagai *output.*
7. *Rule*_Obj_Out_Name adalah nama objek yang berperan sebagai *output*.
8. *Rule*_Obj_Out_Status adalah status objek yang berperan sebagai *output*, 100 untuk ON dan 0 untuk OFF.

   *Rule*_User_Id adalah ID pengguna pemilik *rule.*

2.9 Implementasi Algoritma Eksekusi *Rule* Terdistribusi

Fitur eksekusi *rule* dilakukan pada aplikasi objek simulator. Pengguna akan memilih objek sensor PIR yang akan diubah statusnya, setelah itu akan dilakukan pencarian *rule* yang sesuai dengan status objek yang diubah. Apabila *rule* ditemukan maka *node* sensor PIR akan mengirim perintah ke objek lampu yang dipengaruhi oleh *rule* tersebut.



Gambar 7 Eksekusi *rule terdistribusi*

Pada Gambar 7 menunjukkan PIR_I diberikan status 100 yang berarti aktif. Sesuai dengan *rule* yang tersimpan pada *node* sensor PR_I maka lampu_I, lampu_II dan lampu_III akan menyala apabila PIR_I aktif.

2.10 Implementasi Algoritma *Update* Status Objek ke *Base Station*

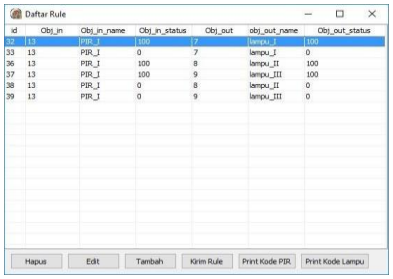
Fitur *update* status objek ke aplikasi kontrol dan monitoring digunakan untuk memenuhi fungsi monitoring pada sistem ini. Setiap terjadi perubahan status pada objek maka objek akan mengirim status terbarunya. Paket data objek dikirim dengan menggunakan format yang telah ditentukan.

Gambar 8 menunjukkan log pengiriman status objek dari aplikasi *object* simulator ke aplikasi kontrol dan monitoring. Informasi yang disimpan adalah ID objek pengirim, waktu penerimaan, status terbaru objek dan *power* objek.

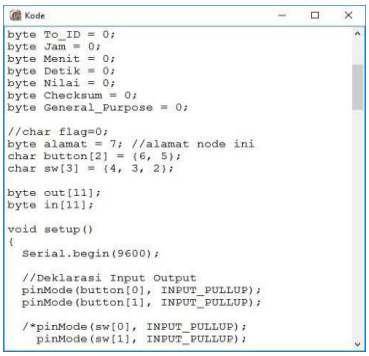
Gambar 8 Tabel logpenerimaan status objek

2.11 Implementasi Algoritma *Print* Kode Program *Node* Sensor PIR

Gambar 9 menunjukkan *rule* yang dipilih adalah *rule* dengan *id* 32 dengan penjelasan apabila PIR_I dengan *id* 13 mendeteksi manusia maka lampu_I dengan *id* 7 akan menyala.



Gambar 9 Pilih *rule*

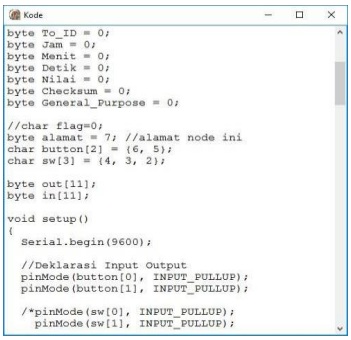


Gambar 10 Hasil kode *node* sensor PIR

Gambar 10 menunjukkan kode program untuk *node* sensor PIR, pada kode tersebut variabel To_ID, From_ID dan Nilai berhasil diubah berdasarkan *rule* yang ada pada halaman daftar *rule*.

2.12 Implementasi Algoritma *Print* Kode Program *Node* Lampu

Berdasarkan *rule* yang dipilih pada Gambar 9 objek lampu_I memiliki id 7. Gambar 11 menunjukkan variabel alamat berhasil diubah sesuai dengan id lampu_I yang ada pada halaman daftar *rule*.



Gambar 11 Hasil kode *node* lampu

3. PENGUJIAN DAN ANALISIS

3.1 Pengujian Algoritma Kirim *Rule* dan Simpan *Rule*

Tujuan dari pengujian ini adalah, untuk

mengetahui keberhasilan aplikasi kontrol dan monitoring dalam mengirim *rule* ke aplikasi objek simulator dan untuk mengetahui keberhasilan aplikasi objek simulator dalam menerima dan menyimpan *rule* yang diterima.

Dari Gambar 12, Gambar 13 dan Gambar 14 dapat dilihat bahwa setiap pengujian pengiriman *rule* dari aplikasi kontrol dan monitoring serta penerimaan dan penyimpanan *rule* pada aplikasi objek simulator berhasil dilakukan. Berdasarkan hasil didapatkan bahwa aplikasi kontrol dan monitoring serta objek simulator dapat mengirim *rule*, menerima dan menyimpan *rule* dengan tingkat keberhasilan 100% dari 6 kali pengiriman *rule* pada masing-masing objek sensor PIR. Algoritma penyimpanan *rule* berjalan sesuai perancangan, *rule* berhasil disimpan pada *file* Rule.ini.

NO	*Rule* dengan objek masukan PIR 1			
	Status Object In	Object Out	Status Object Out	Hasil
1	100	Lampu_I	100	Terima-Simpan
2	100	Lampu_II	100	Terima-Simpan
3	100	Lampu_III	100	Terima-Simpan
4	100	Lampu_I Lampu_II	100	Terima-Simpan
5	100	Lampu_I Lampu_III	100	Terima-Simpan
6	100	Lampu_II Lampu_III	100	Terima-Simpan

Gambar 12 Pengujian kirim dan simpan *rule* PIR 1

NO	*Rule* dengan objek masukan PIR 2			
	Status Object In	Object Out	Status Object Out	Hasil
1	100	Lampu_I	100	Terima-Simpan
2	100	Lampu_II	100	Terima-Simpan
3	100	Lampu_III	100	Terima-Simpan
4	100	Lampu_I Lampu_II	100	Terima-Simpan
5	100	Lampu_I Lampu_III	100	Terima-Simpan
6	100	Lampu_II Lampu_III	100	Terima-Simpan

Gambar 13 Pengujian kirim dan simpan *rule* PIR 2

NO	*Rule* dengan objek masukan PIR 3			
	Status Object In	Object Out	Status Object Out	Hasil
1	100	Lampu_I	100	Terima-Simpan
2	100	Lampu_II	100	Terima-Simpan
3	100	Lampu_III	100	Terima-Simpan
4	100	Lampu_I Lampu_II	100	Terima-Simpan
5	100	Lampu_I Lampu_III	100	Terima-Simpan
6	100	Lampu_II Lampu_III	100	Terima-Simpan

Gambar 14 Pengujian kirim dan simpan *rule* PIR 3

Tabel 4 menunjukkan deskripsi pengujian pengiriman dan penyimpanan *rule.*

Tabel 4 Deskripsi uji kirim dan simpan *rule*

Nama	Deskripsi
Object in	Objek yang diberi *rule*. PIR_I = Sensor PIR 1 PIR_II = Sensor PIR 2 PIR_III = Sensor PIR 3
Status *object in*	Status dari objek yang diberi *rule*. 100 = mendeteksi manusia 0 = tidak mendeteksi manusia
Object Out	Objek yang dieksekusi. Lampu_I = Objek lampu 1 Lampu_II = Objek lampu 2 Lampu_II = Obje lampu 3
Status *object out*	Status dari objek yang dieksekusi. 100 = Lampu menyala 0 = Lampu mati

3.2 Pengujian Eksekusi *Rule* Terdistribusi

Tujuan dari pengujian ini adalah untuk menguji eksekusi *rule* secara terdistribusi. Eksekusi *rule* dilakukan pada objek simulator oleh objek sensor PIR yang sudah menyimpan *rule*. Hal yang diuji adalah ketepatan objek sensor PIR dalam mengirim perintah status kepada objek lampu sesuai dengan *rule* yang diinginkan. Pengujian dilakukan secara bergantian kepada tiga objek sensor PIR.

NO	*Rule* dengan objek in PIR_I			
	Status Object In	Object Out	Status Object Out	Hasil
1	100	Lampu_I	100	Menyala
2	100	Lampu_II	100	Menyala
3	100	Lampu_III	100	Menyala
4	100	Lampu_I Lampu_II	100	Menyala
5	100	Lampu_I Lampu_III	100	Menyala
6	100	Lampu_II Lampu_III	100	Menyala

Gambar 15 Pengujian eksekusi *rule* terdistribusi PIR 1

NO	*Rule* dengan objek in PIR_II			
	Status Object In	Object Out	Status Object Out	Hasil
1	100	Lampu_I	100	Menyala
2	100	Lampu_II	100	Menyala
3	100	Lampu_III	100	Menyala
4	100	Lampu_I Lampu_II	100	Menyala
5	100	Lampu_I Lampu_III	100	Menyala
6	100	Lampu_II Lampu_III	100	Menyala

Gambar 16 Pengujan eksekusi *rule* terdistribusi PIR 2

NO	*Rule* dengan objek in PIR_III			
	Status Object In	Object Out	Status Object Out	Hasil
1	100	Lampu_I	100	Menyala
2	100	Lampu_II	100	Menyala
3	100	Lampu_III	100	Menyala
4	100	Lampu_I Lampu_II	100	Menyala
5	100	Lampu_I Lampu_III	100	Menyala
6	100	Lampu_II Lampu_III	100	Menyala

Gambar 17 Pengujian eksekusi *rule* terdistribusi PIR 3

Tabel 5 Deskripsi uji eksekusi *rule* terdistribusi

Nama	Deskripsi
Object in	Objek yang diberi *rule*. PIR_I = Sensor PIR 1 PIR_II = Sensor PIR 2 PIR_III = Sensor PIR 3
Status *object in*	Status dari objek yang diberi *rule*. 100 = mendeteks manusia 0 = tidak mendeteksi manusia
Object Out	Objek yang dieksekusi. Lampu_I = Objek lampu 1 Lampu_II = Objek lampu 2 Lampu_II = Obje lampu 3
Status *object out*	Status dari objek yang dieksekusi. 100 = Lampu menyala 0 = Lampu mati

Dari Gambar 15, Gambar 16 dan Gambar 17 dapat dilihat bahwa setiap pengujian eksekusi *rule* terdistribusi pada sistem berhasil dilakukan. Tabel 5 menunjukkan deskripsi pengujian eksekusi *rule* ter distribusi. Berdasarkan hasil pengujian dapat dikatakan bahwa objek Sensor PIR melakukan eksekusi *rule* terdistribusi dengan tingkat keberhasilan 100% dari 6 kali percobaan. Lampu 1, lampu 2 dan lampu 3 berjalan sesuai dengan *rule* yang diberikan yang menandakan *rule* terdistribusi dapat dijalankan.

3.3 Pengujian *Update* Status Objek ke *Base Station*

Tujuan dari pengujian ini adalah untuk menguji pengiriman data status objek setelah dilakukan eksekusi *rule* terdistribusi ke aplikasi kontrol dan monitoring. Pengujian dilakukan secara bergantian kepada tiga objek lampu. Pada saat objek lampu dipengaruhi oleh eksekusi rule dari objek sensor PIR, maka objek lampu harus dapat mengirimkan status terbarunya ke aplikasi kontrol dan monitoring untuk memenuhi fungsi monitoring dalam sistem ini.

Dari Tabel 6, Tabel 7 dan Tabel 8 dapat dilihat bahwa setiap pengiriman status objek dari aplikasi *object* simulator ke aplikasi kontrol dan monitoring berhasil dilakukan. Berdasarkan hasil pengujian didapatkan bahwa objek lampu dapat mengirim status kepada aplikas kontrol dan monitoring dengan tingkat keberhasilan 100% dari 13 kali pengiriman data pada masing-masing objek lampu. Algoritma pada aplikasi *object* simulator dapat berjalan sesuai dengan perancangan.

Tabel 6 Hasil pengujian objek lampu 1

No	Waktu	Status
1	2017-04-23 09:31:52	100
2	2017-04-23 10:27:00	0
3	2017-04-23 10:45:15	100
4	2017-04-23 10:50:36	0
5	2017-04-23 14:10:06	100
6	2017-04-23 14:12:35	0
7	2017-04-23 14:18:15	100
8	2017-04-23 14:35:51	0
9	2017-04-23 14:40:06	100
10	2017-04-23 14:46:03	0
11	2017-04-23 14:52:47	100
12	2017-04-23 14:58:32	0
13	2017-04-23 15:04:06	100

Tabel 7 Hasil pengujian objek lampu 2

No	Waktu	Status
1	2017-04-23 09:31:53	100
2	2017-04-23 10:27:01	0
3	2017-04-23 10:45:15	100
4	2017-04-23 10:50:37	0
5	2017-04-23 14:10:07	100
6	2017-04-23 14:12:36	0
7	2017-04-23 14:18:15	100
8	2017-04-23 14:35:51	0
9	2017-04-23 14:40:06	100
10	2017-04-23 14:46:03	0
11	2017-04-23 14:52:47	100
12	2017-04-23 14:58:32	0
13	2017-04-23 15:04:07	100

Tabel 8 Hasil pengujian objek lampu 3

No	Waktu	Status
1	2017-04-23 09:31:53	100
2	2017-04-23 10:27:01	0
3	2017-04-23 10:45:15	100
4	2017-04-23 10:50:37	0
5	2017-04-23 14:10:07	100
6	2017-04-23 14:12:36	0
7	2017-04-23 14:18:15	100
8	2017-04-23 14:35:52	0
9	2017-04-23 14:40:06	100
10	2017-04-23 14:46:03	0
11	2017-04-23 14:52:48	100
12	2017-04-23 14:58:33	0
13	2017-04-23 15:04:07	100

Tabel 9 menunjukkan deskripsi pengujian untuk *update* status objek ke aplikasi kontrol dan monitoring.

Tabel 9 Deskripsi tabel hasil pengujian

Nama	Deskripsi
Objek	Objek yang mengirimkan status ke aplikasi kontrol dan monitoring 1. Lampu_I = Lampu 1 2. Lampu_II = Lampu 2 3. Lampu_III = Lampu 3
Waktu	Menampilkan waktu pada saat status objek dikirim
Status	Status dari objek lampu 1. 0 = tidak menyala 2. 100 = menyala

4. KESIMPULAN

Berdasarkan rumusan masalah yang telah dibuat, hasil perancangan, implementasi dan pengujian yang dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Perancangan sistem simulator *centralized* pada *smart home* dengan metode eksekusi *rule* terdistribusi dirancang dengan menggunakan bahasa pemrograman Delphi dan sistem-sistem yang diimplementasikan dalam *Smart Home Centralized* Simulator ini dapat saling berintegrasi dengan memanfaatkan komponen Indiy sebagai media komunikasi *socket* dalam mengirim dan menerima data.
2. Aplikasi objek simulator dan apliasi kontrol dan monitoring dapat melakukan pengiriman dan penerimaan data dengan menggunakan komunikasi *socket.*
3. Aplikasi objek simulator dapat melakukan eksekusi *rule* secara terdistribusi dengan cara membaca *rule* yang telah tersimpan pada *file* Rule.ini ketika sensor PIR mendeteksi manusia. Berdasarkan data hasil pembacaan *rule* maka objek PIR akan mengeksekusinya dengan cara mengirim nilai status ke objek lampu secara langsung tanpa melalui aplikasi kontrol dan monitoring. Dengan demikian menunjukkan bahwa eksekusi *rule* terjadi secara terdistribusi pada setiap objek PIR dan tidak dieksekusi secara terpusat pada aplikasi kontrol dan monitoring.

5. DAFTAR PUSTAKA

Alam, M. R., Reaz, M. B., & Ali, M. A. (2012). A Review of Smart Homes – Past, Present, and Future. *IEEE Transactions*.

Devart. (2010). *MySQL Data Access Components*. Dipetik November 2016, dari http://www.devart.com/mydac/

Ellamonica, V. A. (2015). *PERANCANGAN APLIKASI HOME AUTOMATION.* Malang: Universitas Brawijaya.

eTutorials.org. (2015). *Building Socket Applications*. Dipetik November 2016, dari

http://etutorials.org/Programming/mastering+delphi+7/Part+IV+Delphi+the+Internet+and+a+.NET+Preview/Chapter+19+Internet+Programming+Sockets+and+Indy/Building+Socket+Applications/

Fernández, A. C., Secilla, J. M., Muñoz, J. M., Bueno, J. O., & García, F. L. (2013). Distributed Intelligent Rule-Based Wireless Sensor Network Architecture. *Ambient Intelligence, 219*, 187-194.

Juwita Utami Putri, S. M. (2005). *Universitas Gunadarma staffsite.* Dipetik April 13, 2017, dari http://jutami.staff.gunadarma.ac.id/Downloads/files/29491/Sejarah+Delphi.doc

mandalamaya.com. (2015). *Pengertian SQL Dan Jenis-Jenis Perintah SQL*. Dipetik April 13, 2017, dari http://www.mandalamaya.com/pengertian-sql-dan-jenis-jenis-perintah-sql/

Mustika, Y. P. (2015). *SMART HOME MENGGUNAKAN WIRELESS SENSOR NETWORK DENGAN PROSES EKSEKUSI RULE TERDISTRIBUSI: PENERAPAN PADA SENSOR PIR DAN LAMPU DI DALAM RUANG.* Malang: Universitas Brawijaya.

Nurcahyo, E. (2011). *Pengenalan Socket programming*. Dipetik April 13, 2017, dari http://egrit-nurcahyo-w.blog.ugm.ac.id/2011/09/25/pengenalan-socket-programming/

Oracle Corporation. (2017). *About MySQL*. Dipetik April 13, 2017, dari https://www.mysql.com/about/

Oracle Corporation. (t.thn.). *MySql*. Dipetik November 2016, dari MySql: https://www.mysql.com/why-mysql/white-papers/mysql-for-internet-of-things/

Setyawan, G. E., Setiabudi, A., Muttaqin, A., & Ichsan, M. H. (2016). Centralized Smart Home Framework Monitoring for Power Savings. *Jurnal of Information Technology and Compute Science*.

Synnott, J., Nugent, C., & Jeffers, P. (2015). Simulation of Smart Home ActivityDatasets. *Sensors*, 14162-14179.

Wahana Komputer. (2010). *Panduan Aplikatif dan Solusi Membuat Aplikasi Client Server dengan Visual Basic 2008.* Surabaya: Andi Publisher.

Yunarman, T. F., & Azman, N. (2009). PERANCANGAN SOFTWARE APLIKASI PERVASIVE SMART HOME. *Seminar Nasional Aplikasi Teknologi Informasi 2009*.