**BAB IV**

**IMPLEMENTAS DAN PEMBAHASAN SISTEM**

1. **Implementasi**

Tahap implementasi meliputi penjelasan mengenai implementasi perangkat lunak (*software*) dan implementasi perangkat keras (*hardware*) serta implementasi antarmuka (*interface*).

1. **Ruang Lingkup Implementasi**

Ruang lingkup implementasi dimaksudkan agar implementasi menjadi lebih jelas. Ruang lingkup implementasi sistem keamanan rumah menggunakan arduino mega 2560 dengan metode *fuzzy logic* berbasis android ini adalah sebagai berikut:

1. Pengujian Sistem Keamanan Rumah Menggunakan Arduino Mega 2560 Dengan Metode *Fuzzy Logic* Berbasis Android menggunakan perangkat *smartphone android*.
2. Perangkat implementasi menggunakan perangkat *smartphone android* dan rangkaian perangkat keras.
3. Implementasi sistem keamanan rumah dalam penelitian ini difokuskan pada monitoring terhadap keadaan rumah.
4. **Implementasi Perangkat Keras**

Implementasi sistem keamanan rumah menggunakan arduino mega 2560 dengan metode *fuzzy logic* ini, perangkat keras yang digunakan adalah sebagai berikut:

**Tabel 4.1 Spesifikasi Perangkat Keras (*Hardware*)**

|  |  |
| --- | --- |
| Perangkat Kontroler | Smartphone Android |
| Arduino Mega 2560 | CPU : 1GHz |
| Arduino Ethernet Sheld | Ram : 512 MB |
| Adaptor 500A | Free Space : 20 MB |
| Sensor PIR |  |
| Sensor *Ultrasonic* |  |
| *Magnetic Switch* |  |
| *Buzzer* |  |

1. **Implementasi Perangkat Lunak**

Untuk implementasi sistem keamanan rumah menggunakan Arduino Mega 2560 dengan metode fuzzy logic ini, perangkat lunak yang digunakan untuk mendukung berjalannya sistem ditunjukan pada tabel 4.2 berikut ini.

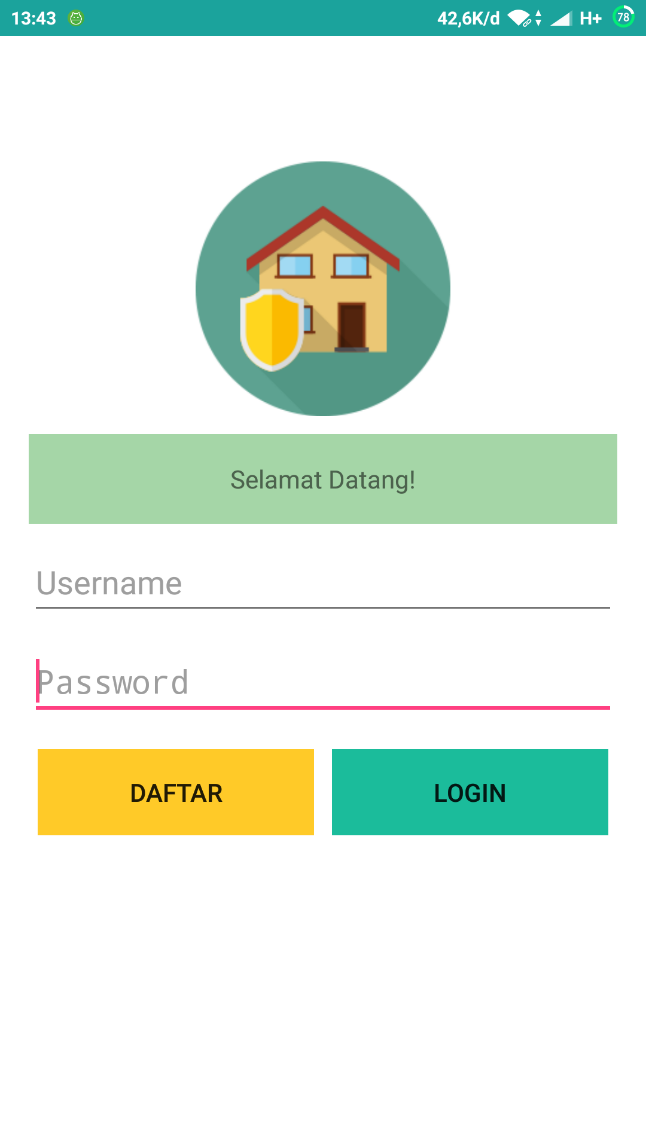
**Tabel 4.2 Spesikasi Perangkat Lunak (*Software*)**

|  |  |
| --- | --- |
| Sistem Operasi Android | KitKat 4.4 |
| Database | MySql |
| Sistem Operasi Windows | Windows 7 |
| Web Browser | Google Chrome |

1. **Implementasi Antarmuka**

Pada tahapan ini pengujian sistem yang dibahas berdasarkan objek yang akan diuji berupa *form*-*form* sebagai berikut:

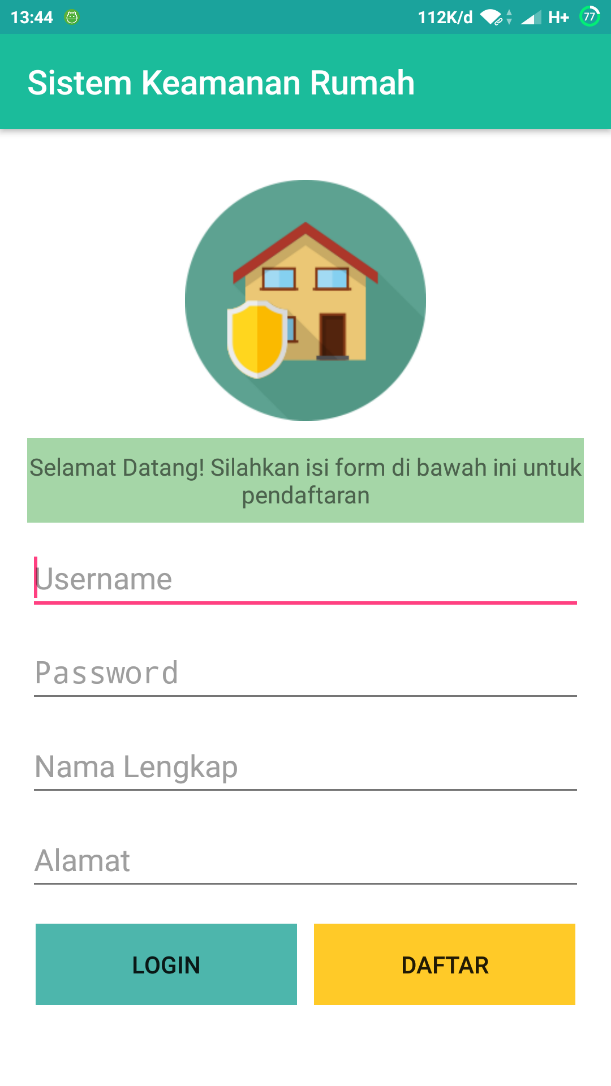
1. **Antarmuka Aplikasi Android**
2. *Form* Login



**Gambar 4.1 *Form* Login**

Pada *form* ini pengguna dihadapkan dengan tampilan login yang dimana tampilan akan muncul ketika pengguna membuka aplikasi untuk pertama kali dan kemudian pengguna diminta untuk memasukan *username* dan *password*.

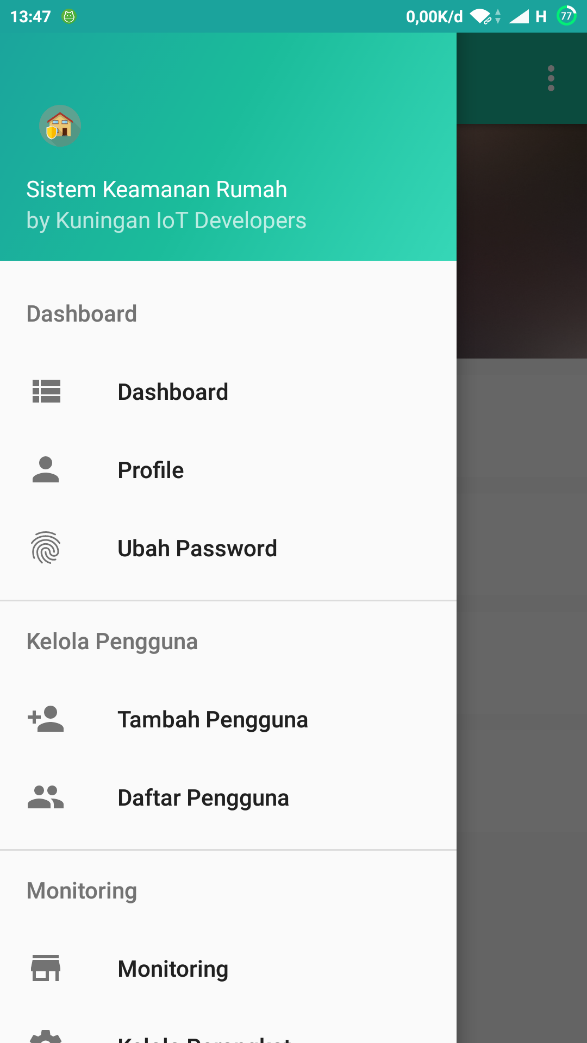
1. *Form* Daftar



**Gambar 4.2 *Form* Daftar**

Pada *form* ini pengguna dihadapkan dengan tampilan daftar yang dimana tampilan akan muncul ketika pengguna menekan tombol daftar pada halaman login kemudian pengguna diminta untuk memasukan data sesuai isian yang ditampilkan.

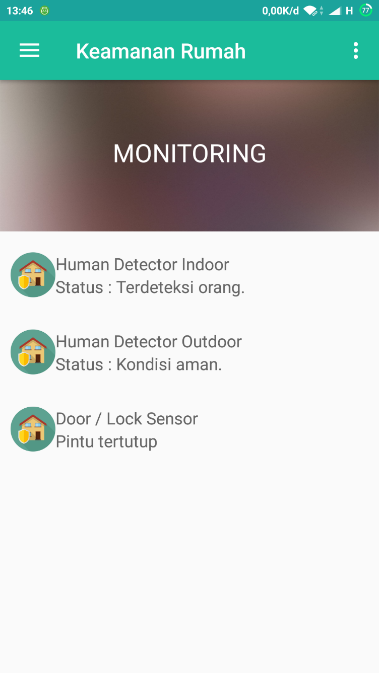
1. *Form* Menu Utama



**Gambar 4.3 Tampilan *Form* Menu Utama**

Pengguna akan dihadapkan pada tampilan awal yaitu *form* Menu Utama. Terdiri dari :

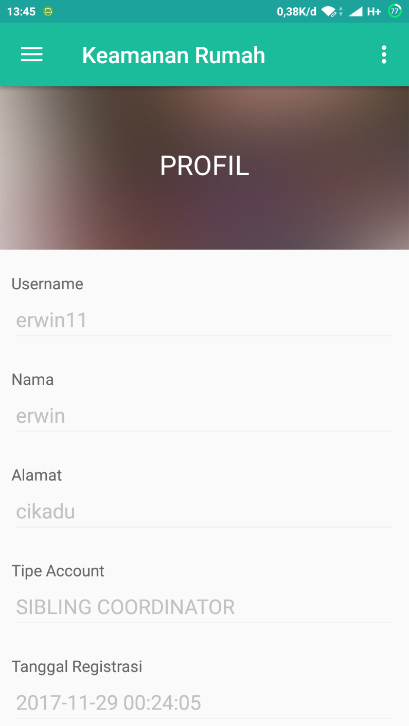
1. Monitoring : untuk memantau kondisi terkini rumah .
2. Kelola Profil : berfungsi untuk mengubah profil user yang sedang *login*
3. Kelola *Password* : berfungsi untuk mengubah password user yang sedang *login*
4. Kelola Pengguna : berfungsi untuk mengelola data pengguna (fitur ini hanya bisa diakses oleh root / koordinator keluarga)
5. *Form* Monitoring



**Gambar 4.4 Tampilan *Form* Monitoring**

Pada *form* ini menampilkan status terkini dari sensor yang terdapat pada rumah pengguna.

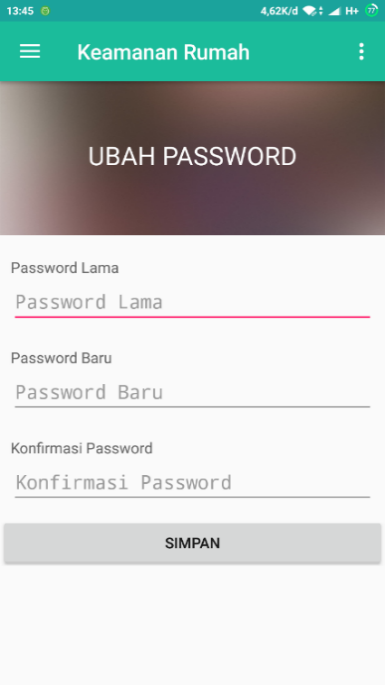
1. *Form* Kelola Profil



**Gambar 4.5 *Form* Kelola Profil**

Pada *form* ini pengguna dihadapkan dengan tampilan ubah profil yang dimana terdapat beberapa *field* berisi data pengguna yang dapat dirubah.

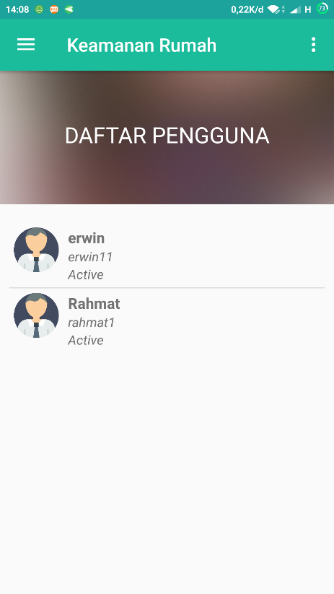
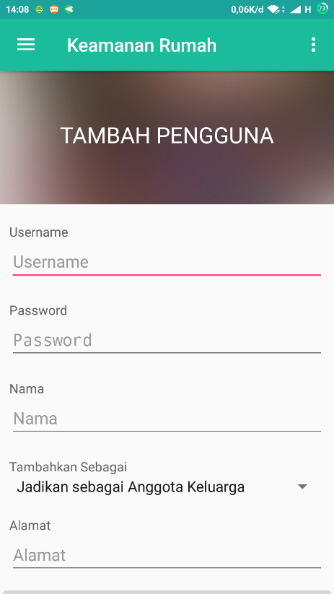
1. *Form* Kelola Password



**Gambar 4.6 *Form* Kelola Password**

Pada *form* ini pengguna dihadapkan dengan tampilan ubah *password* yang dimana terdapat tiga *field* yang harus diisi ketika pengguna ingin merubah *password* miliknya.

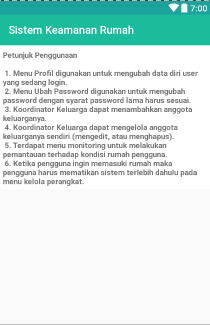
1. *Form* Kelola Pengguna

**Gambar 4.7 *Form* Kelola Pengguna**

Pada *form* ini pengguna dihadapkan dengan submenu yaitu tambah pengguna dan daftar pengguna, jika pengguna mengklik daftar pengguna maka pengguna (root / koordinator keluarga) dapat mengedit atau menghapus pengguna tersebut.

1. *Form* Petunjuk



**Gambar 4.8 Tampilan Petunjuk**

Pada halaman antarmuka ini pengguna akan diperlihatkan petunjuk penggunaan aplikasi ini.

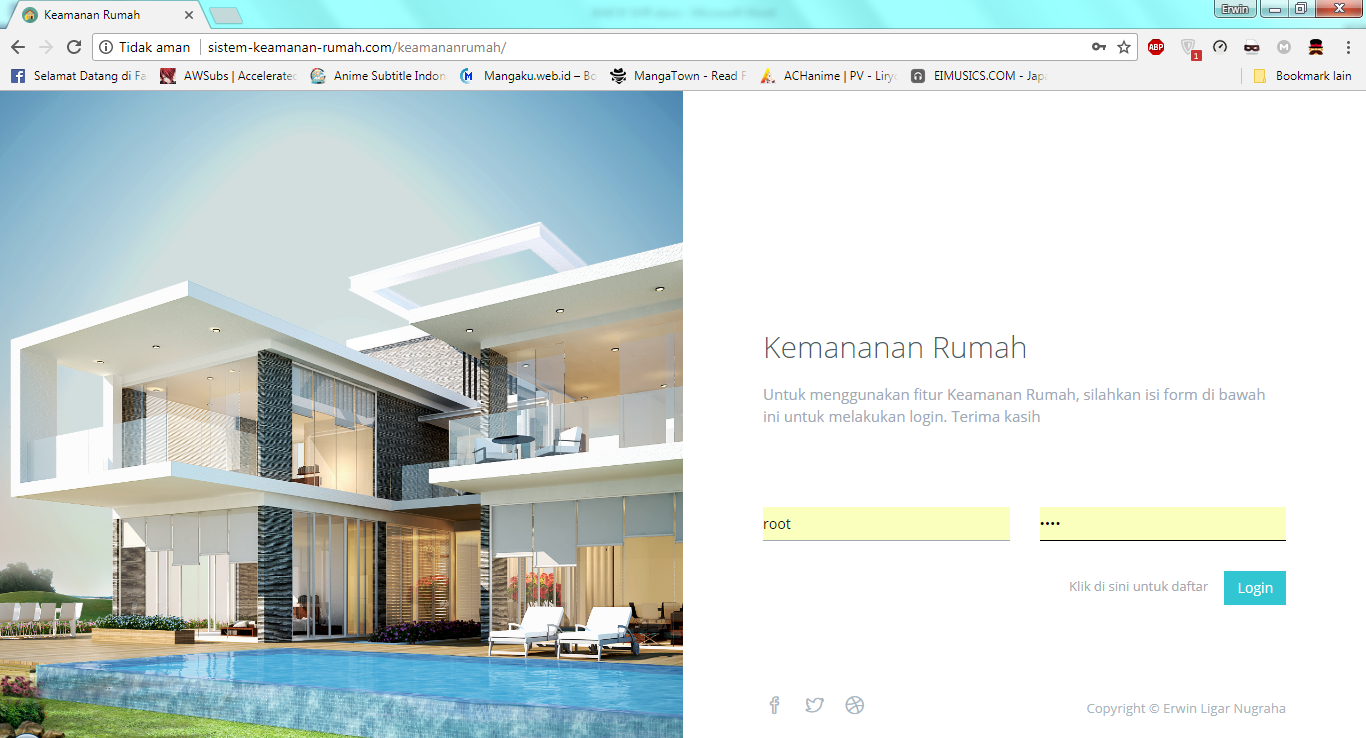
1. *Form* Tentang



**Gambar 4.9 Tampilan Tentang**

Pada halaman antarmuka pengguna akan diperlihatkan info pengembang.

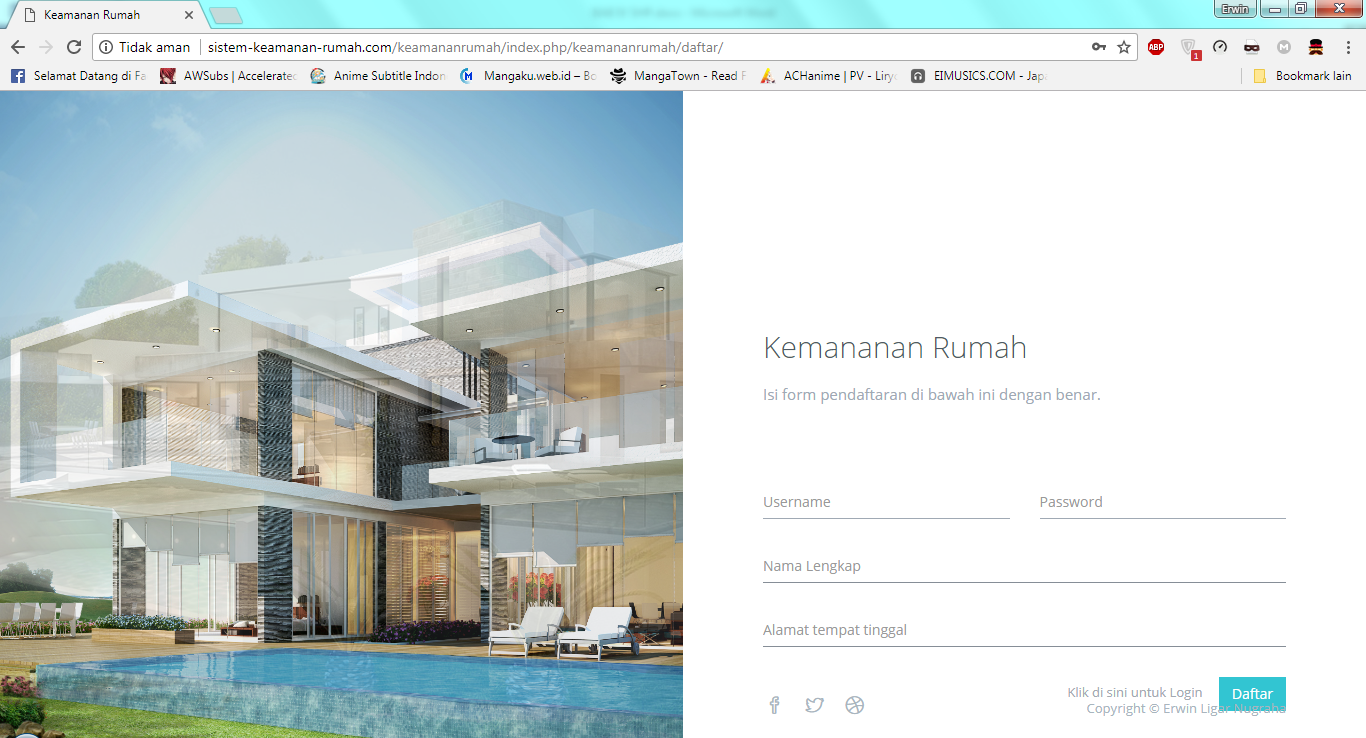
1. **Antarmuka *Web Service***
2. *Form* Login



**Gambar 4.12 *Form* Login**

Pada *form* ini pengguna dihadapkan dengan tampilan login yang dimana tampilan akan muncul ketika pengguna membuka aplikasi untuk pertama kali dan kemudian pengguna diminta untuk memasukan *username* dan *password*.

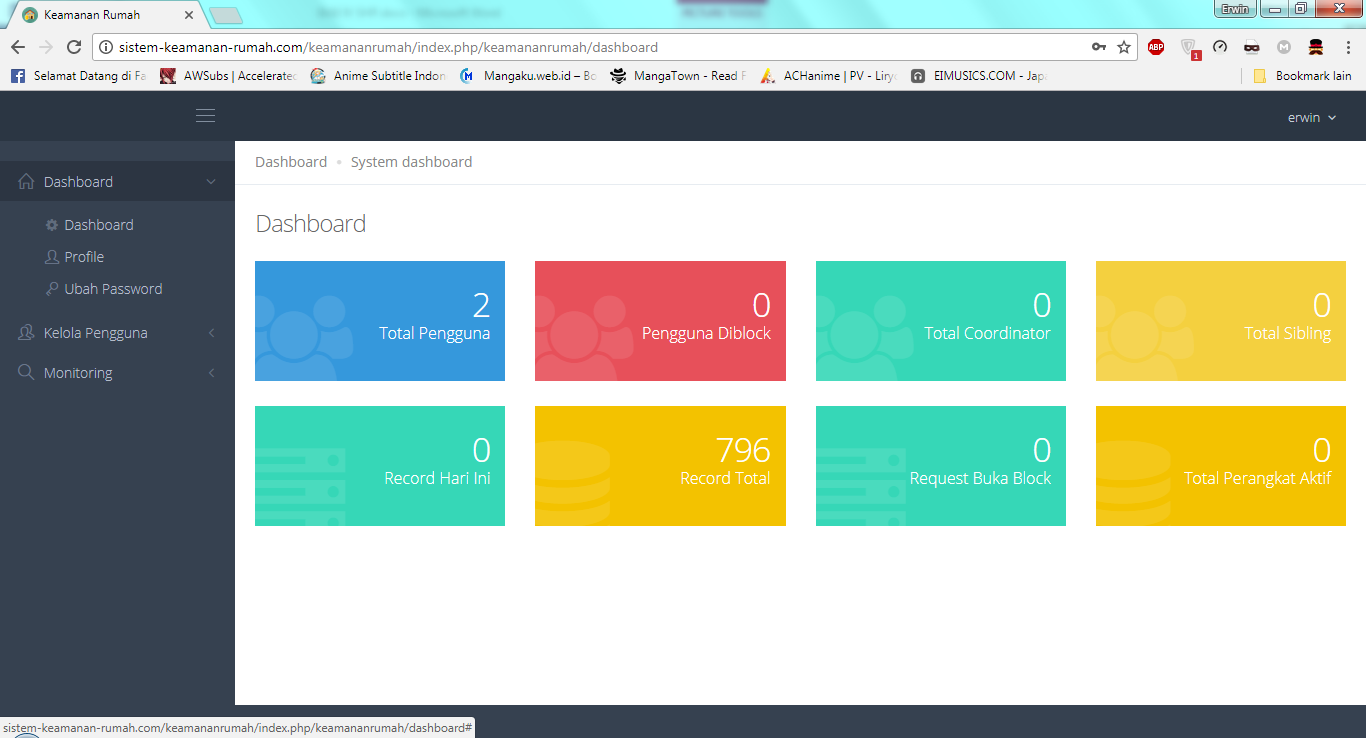
1. *Form* Daftar



**Gambar 4.13 *Form* Daftar**

Pada *form* ini pengguna dihadapkan dengan tampilan daftar yang dimana tampilan akan muncul ketika pengguna menekan tombol daftar pada halaman login kemudian pengguna diminta untuk memasukan data sesuai isian yang ditampilkan.

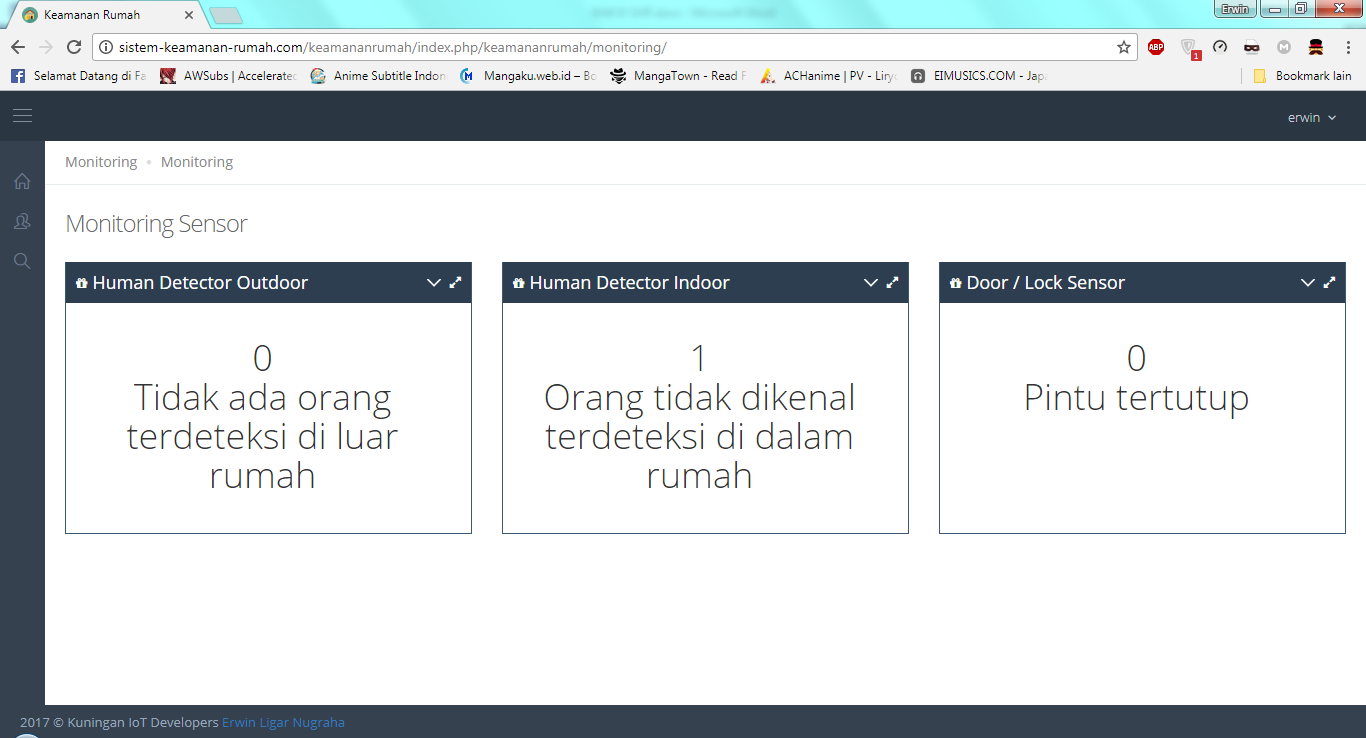
1. *Form* Menu Utama



**Gambar 4.14 Tampilan *Form* Menu Utama**

Pengguna akan dihadapkan pada tampilan awal yaitu *form* Menu Utama. Terdiri dari :

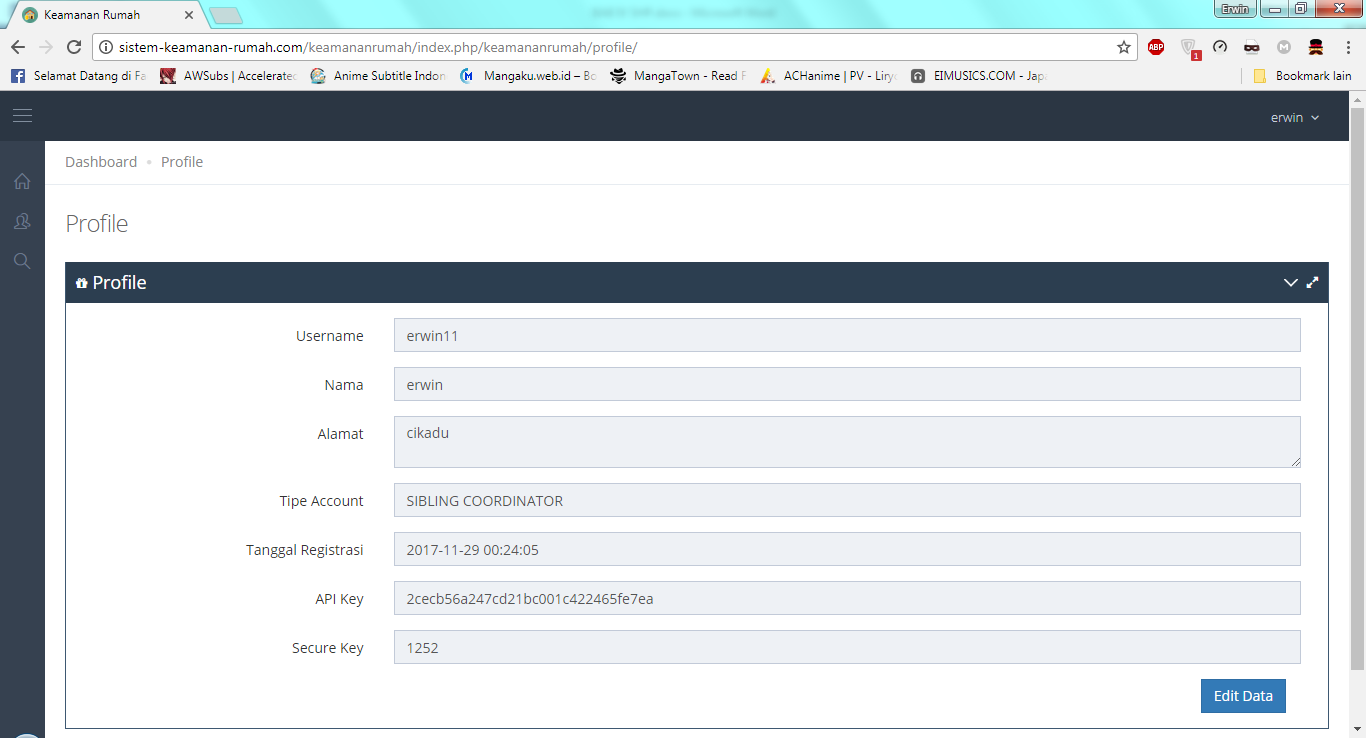
1. Monitoring : untuk memantau kondisi terkini rumah .
2. Kelola Profil : berfungsi untuk mengubah profil user yang sedang *login*
3. Kelola *Password* : berfungsi untuk mengubah password user yang sedang *login*
4. Kelola Pengguna : berfungsi untuk mengelola data pengguna (fitur ini hanya bisa diakses oleh root / koordinator keluarga)
5. *Form* Monitoring



**Gambar 4.15 Tampilan *Form* Monitoring**

Pada *form* ini menampilkan status terkini dari sensor yang terdapat pada rumah pengguna.

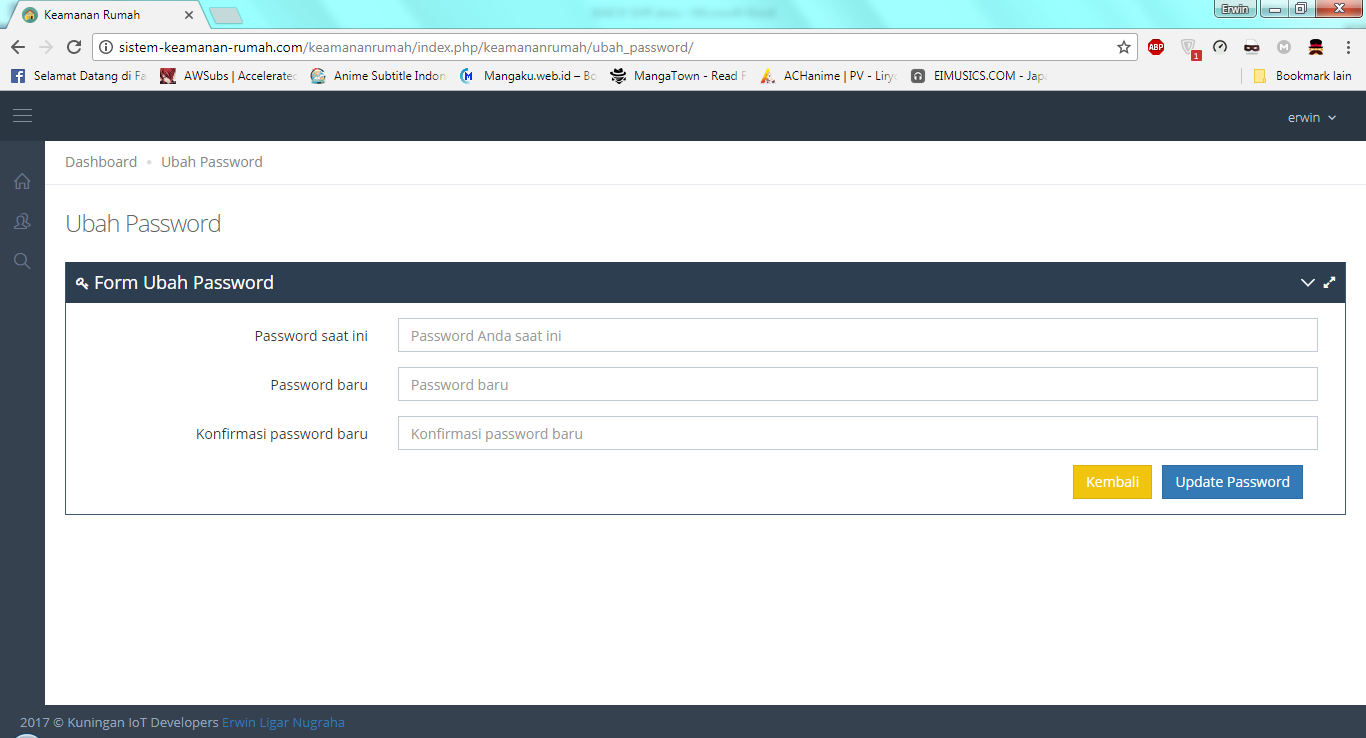
1. *Form* Kelola Profil



**Gambar 4.16 *Form* Kelola Profil**

Pada *form* ini pengguna dihadapkan dengan tampilan ubah profil yang dimana terdapat beberapa *field* berisi data pengguna yang dapat dirubah.

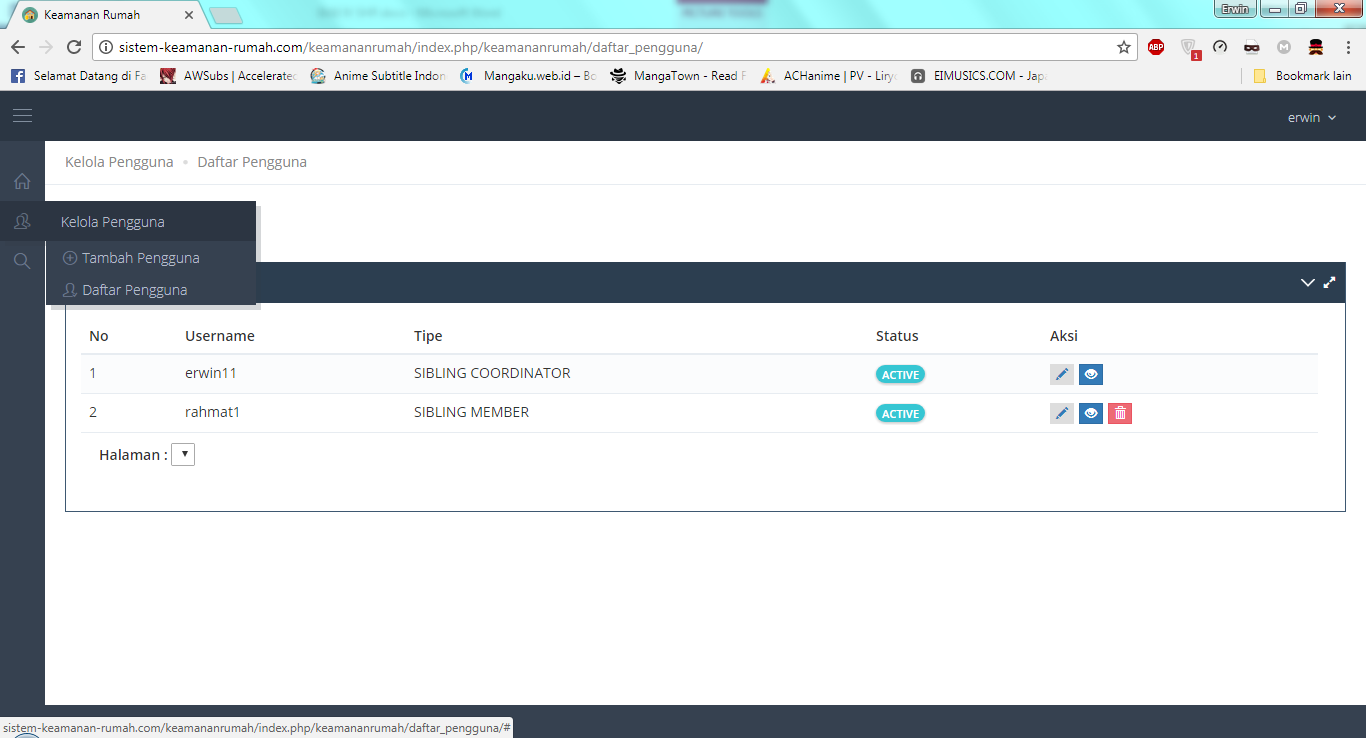
1. *Form* Kelola Password



**Gambar 4.17 *Form* Kelola Password**

Pada *form* ini pengguna dihadapkan dengan tampilan ubah *password* yang dimana terdapat tiga *field* yang harus diisi ketika pengguna ingin merubah *password* miliknya.

1. *Form* Kelola Pengguna

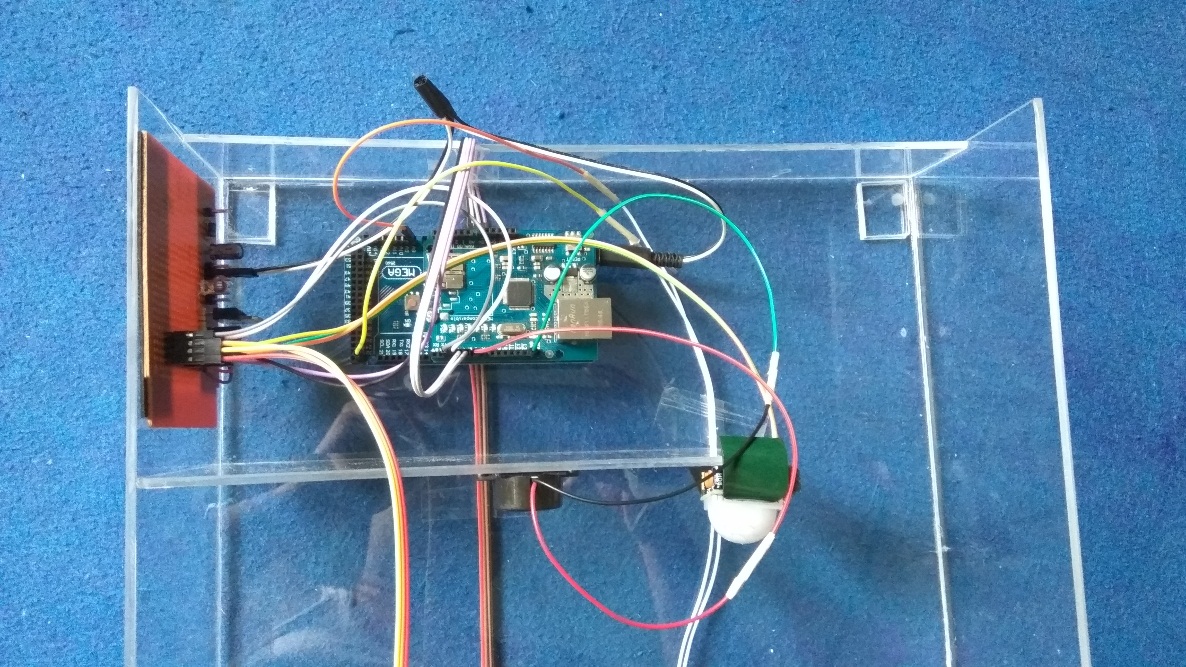


**Gambar 4.18 *Form* Kelola Pengguna**

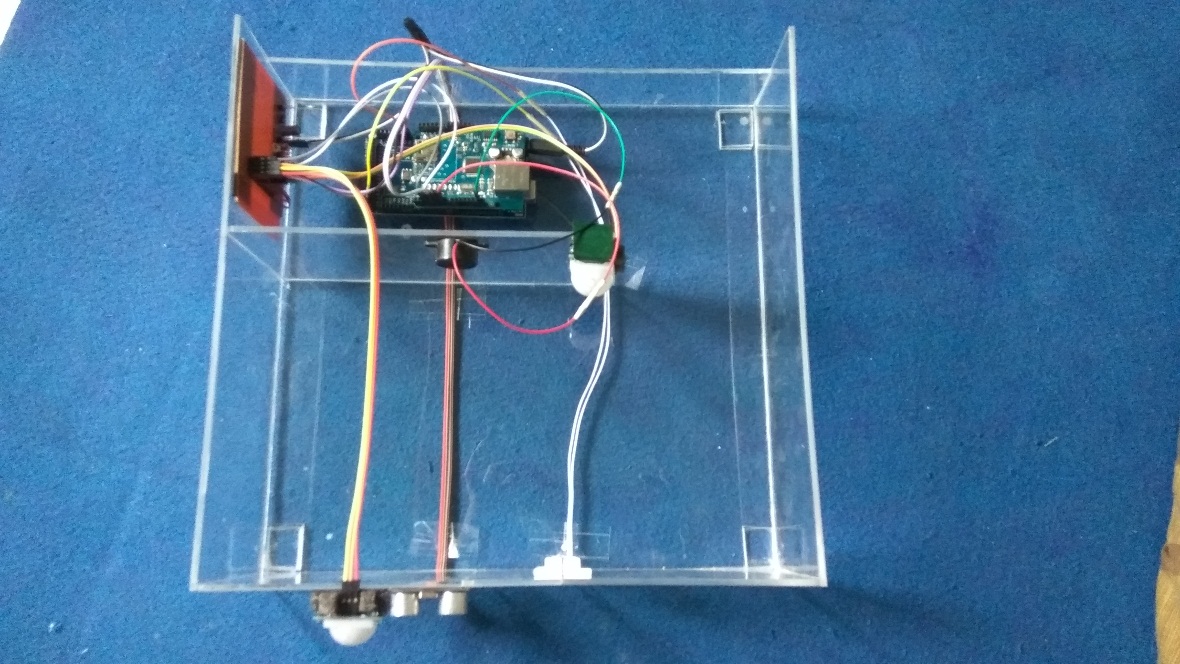
Pada *form* ini pengguna dihadapkan dengan submenu yaitu tambah pengguna dan daftar pengguna, jika pengguna mengklik daftar pengguna maka pengguna (root / koordinator keluarga) dapat mengedit atau menghapus pengguna tersebut.

1. **Implementasi *Hardware***

Pada tahapan ini pengujian sistem yang dibahas berdasarkan objek yang akan diuji berupa *hardware* sebagai berikut:



**Gambar 4.19 Tampilan Pengkabelan Pada Maket**



**Gambar 4.20 Tampilan Maket Tampak Atas**

1. **Pengujian Sistem**

Pengujian sistem adalah proses mengeksekusi perangkat lunak untuk menentukan apakah sistem tersebut memenuhi spesifikasi sistem dan berjalan sesuai dengan yang diharapkan.

* + 1. ***Black Box Testing***

Pada tahap ini pengujian sistem yang dibahas yaitu objek-objek pada antarmuka yang berupa *Button, TextView* dan koneksi antar perangkat.

**Tabel 4.3 Pengujian *Black Box Testing***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Fungsi yang diuji | Cara Menguji | Hasil yang diharapkan | Hasil pengujian |
| 1. | Login | Pengguna memasukan *username* dan *pasword* kemudian menekan tombol login | Sistem akan mengarahkan halaman pada tampilan menu utama sesuai dengan tingkatan user | Sistem menampilkan halaman utama dengan alert masuk sesuai dengan tikatan user. |
| 2. | Daftar | Pengguna memasukan username, pasword, kemudian menekan tombol daftar | Jika pendaftaran berhasil sistem akan mengarahkan pengguna pada halaman login, dan jika gagal maka sistem akan mengeluarkan peringatan | Pendaftaran berhasil data berhasil disimpan dan redirect ke halaman login, data gagal disimpan sistem menampilkan peringatan |
| 3. | Monitoring | Pengguna mengirim perintah dengan menekan menu monitoring. | Sistem akan mengirim *request* data pada *server* untuk memunculkan data dari sensor. | Sistem menampilkan status sensor terkini. |
| 4. | Kelola Profil | Pengguna mengirim perintah dengan menekan menu kelola profil, kemudian merubah profil dirinya sendiri. | Sistem menampilkan data pengguna yang sedang sedang *login* dan dapat dirubah oleh pengguna dengan menekan tombol *edit*. | Sistem menampilkan profil pengguna dan ketika diedit data berhasil disimpan. |
| 5. | Kelola *Password* | Pengguna mengirim perintah dengan menekan menu kelola *password*, kemudian merubah *password*. | Sistem menampilkan form ubah password, jika data sesuai maka pass baru disimpan jika tidak maka sistem mengeluarkan alert. | Sistem menampilkan *form* ubah pass dan ketika data sesuai *password* berhasil disimpan ketika data tidak sesuai sistem mengeluarkan peringatan. |
| 6. | Kelola Pengguna | Pengguna mengirim perintah dengan menekan menu kelola pengguna, kemudian melakukan tambah, edit, atau hapus pengguna. | 1. Sistem menampilkan form tambah pengguna jika pengguna memilih submenu tersebut, jika data pengguna baru sesuai maka data akan disimpan, jika tidak maka sistem menampilkan peringatan  2. Jika pengguna (koordinator keluarga / root) mengedit data pengguna maka sistem akan menampilkan data pengguna yang akan diedit, jika data sesuai maka data akan disimpan, jika tidak maka sistem menampilkan alert.  3. Jika pengguna (koordinator keluarga / root) menghapus data pengguna maka sistem menampilkan alert apakah yakin akan dihapus atau tidak. | 1. Sistem menampilkan form tambah pengguna dan ketika data pengguna baru sesuai berhasil disimpan ketika data tidak sesuai sistem mengeluarkan peringatan.  2. Sistem menampilkan form edit pengguna dan ketika data sesuai berhasil disimpan ketika data tidak sesuai sistem mengeluarkan peringatan.  3. Sistem menampilkan alert ketika pengguna (koordinator keluarga / root) akan menghapus data pengguna |
| 7. | Kelola Sistem | Pengguna mengirim perintah dengan menekan tombol matikan / nyalakan sistem | Sistem mengirim perintah ke arduino untuk mematikan / menyalakan sistem, status sistem berubah menjadi nyala / mati | Sistem menampilkan status sistem menjadi nyala / mati |
| 8. | *Log* | Pengguna mengirim perintah dengan menekan menu Log. | Sistem akan mengirim *request* data pada server untuk memunculkan data dari sensor yang mendeteksi. | Sistem menampilkan data sensor yang mendeteksi pada waktu tertentu. |
| 9. | Notifikasi | Melakukan monitoring sampai state dari setiap sensor berubah menjadi Waspada atau Awas. | 1. Sensor PIR 1 melakukan pengukran 3-5 kali untuk menampilkan notifikai waspada. Sensor melakukan pengukran > 5 kali untuk menampilkan notifikai awas.  2. Sensor PIR 2 melakukan pengukran 3-5 kali untuk menampilkan notifikai waspada. Sensor melakukan pengukran > 5 kali untuk menampilkan notifikai awas.  3. Magnetic switch melakukan pengukran 3-5 kali untuk menampilkan notifikai waspada. Sensor melakukan pengukran > 5 kali untuk menampilkan notifikai awas. | Notifikasi muncul |

1. ***White Box Testing***

Pengujian *white box* berfokus pada struktural kontrol program. Adapun metode yang digunakan dalam pengujian *white box* ini adalah metode basis Path untuk memastikan bahwa semua statement pada program telah dieksekusi paling tidak satu kali selama pengujian dan bahwa kondisi logis yang telah diuji. Berikut adalah struktur program yang akan diuji:

**Tabel 4.4 *Node Flow Graph Nation***

|  |  |
| --- | --- |
| *Line*/*Node* | *Coding* |
|  | #include <Ethernet.h>  #include <SPI.h>  #define pin\_outdoor\_pir 2  #define pin\_indoor\_pir 3  #define pin\_echo\_ussrf 5  #define pin\_trigger\_ussrf 4  #define pin\_buzz 6  #define pin\_magnetic A8  #define STATUS\_CONNECTED 1  #define STATUS\_DISCONNECTED 0  #define STATUS\_NO\_HUMAN\_DETECTED "0"  #define STATUS\_HUMAN\_DETECTED "1"  #define STATE\_NULL "NULL"  #define STATE\_DETECTED "DETECTED"  #define DOOR\_OPEN 1  #define DOOR\_CLOSE 0  #define API\_KEY "2cecb56a247cd21bc001c422465fe7ea"  char namaServer[] = "169.254.2.183";  char inString[1024];  char charFromWeb[9];  byte IP\_eth[] = {169,254,2,184};  byte MAC\_eth[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  int calibrationTime = 30;  int iterasi = 0;  int counter\_outdoor\_detected = 0;  int counter\_outdoor\_no\_detect = 0;  int counter\_indoor\_detected = 0;  int counter\_indoor\_no\_detect = 0;  int counter\_door\_open = 0;  int counter\_door\_close = 0;  boolean lockLow = true;  boolean lockLow2 = true;  boolean takeLowTime;  boolean takeLowTime2;  boolean outdoor\_pin\_status = false;  boolean indoor\_pin\_status = false;  boolean startRead = false;  long unsigned int lowIn;  long unsigned int lowIn2;  long unsigned int pause = 100;  String password;  String secure\_key;  String monitoring = "KONDISI ON";  EthernetClient myEthernet; |
|  | void setup() {  Serial.begin(9600);  Serial.println("--------------------------------------------------");  Serial.println("Setting Perangkat");  Serial.println("Setting PIR dan teman-teman");  pinMode(pin\_trigger\_ussrf, OUTPUT);  pinMode(pin\_echo\_ussrf, INPUT);  pinMode(pin\_outdoor\_pir, INPUT);  digitalWrite(pin\_outdoor\_pir, LOW);  pinMode(pin\_indoor\_pir, INPUT);  digitalWrite(pin\_indoor\_pir, LOW);  //--  pinMode(pin\_buzz,OUTPUT);  pinMode(pin\_magnetic,INPUT);  //--  Serial.print("calibrating sensor ");  for(int i = 0; i < calibrationTime; i++){  Serial.print(".");  delay(100);  }  Serial.println(" done");  Serial.println("SENSOR ACTIVE");  delay(50);  Serial.println("Mohon menunggu . . . ");  Serial.println("Setting Ethernet MAC Address dan IP Address");  Serial.println("Mohon menunggu . . . ");  if (Ethernet.begin(MAC\_eth) == 0) {  Serial.println("Failed to configure Ethernet using DHCP");  Ethernet.begin(MAC\_eth,IP\_eth);  }  // Ethernet.begin(MAC\_eth,IP\_eth);  delay(1000);  Serial.println("Setting Perangkat selesai!");  Serial.println("--------------------------------------------------");  } |
|  | void loop() {  Serial.println("--------------------------------------------------------------");  iterasi++;  Serial.print("Iterasi ke : ");  Serial.println(iterasi);  int resultBukaKoneksi = bukaKoneksi();  String data = collecting\_sensor(outdoor\_pir(),indoor\_pir(),ussrf(),magnetic());  if(resultBukaKoneksi==1){  kirim\_data(data);  Serial.println();  }  Serial.print("Monitoring : ");  Serial.println(monitoring);  Serial.println("--------------------------------------------------------------\n");  delay(3000);  } |
|  | boolean indoor\_pir(){  if (digitalRead(pin\_indoor\_pir) == HIGH){  if (lockLow2) {  lockLow2 = false;  indoor\_pin\_status = true;  delay(50);  }  takeLowTime2 = true;  }  if (digitalRead(pin\_indoor\_pir) == LOW) {  if (takeLowTime2) {  lowIn2 = millis(); //save the time of the transition from high to LOW  takeLowTime2 = false; //make sure this is only done at the start of a LOW phase  }  if (!lockLow2 && millis() - lowIn2 > pause) {  lockLow2 = true;  indoor\_pin\_status = false;  digitalWrite(pin\_buzz,LOW);  delay(50);  }  }  return indoor\_pin\_status;  }  int ussrf(){  long duration, distance;  digitalWrite(pin\_trigger\_ussrf, LOW); // Added this line  delayMicroseconds(2); // Added this line  digitalWrite(pin\_trigger\_ussrf, HIGH); // delayMicroseconds(1000); - Removed this line  delayMicroseconds(10); // Added this line  digitalWrite(pin\_trigger\_ussrf, LOW);  duration = pulseIn(pin\_echo\_ussrf, HIGH);  distance = (duration/2) / 29.1;  delay(500);  return distance;  }  int magnetic(){  int magnet;  int baca\_switch = analogRead(pin\_magnetic);  Serial.print(" > > > > ");  Serial.println(baca\_switch);  if(baca\_switch < 1020 && monitoring.equals("KONDISI ON")){  digitalWrite(pin\_buzz,HIGH);  delay(500);  digitalWrite(pin\_buzz,LOW);  delay(100);  magnet = DOOR\_OPEN;  }else{  digitalWrite(pin\_buzz,LOW);  magnet = DOOR\_CLOSE;  }  return magnet;  } |
|  | String collecting\_sensor(boolean outdoor\_pir,boolean indoor\_pir,int ussrf,int magnetic\_sw){  String data,outdoor,indoor,state,state\_indoor,state\_outdoor,state\_magnetic;  if(outdoor\_pir){  outdoor = STATUS\_HUMAN\_DETECTED;  counter\_outdoor\_detected ++;  counter\_outdoor\_no\_detect = 0;  }else{  outdoor = STATUS\_NO\_HUMAN\_DETECTED;  counter\_outdoor\_detected = 0;  counter\_outdoor\_no\_detect ++;  }  if(indoor\_pir){  indoor = STATUS\_HUMAN\_DETECTED;  if(monitoring.equals("KONDISI ON")){  digitalWrite(pin\_buzz,HIGH);  Serial.println("TRUE");  }else{  digitalWrite(pin\_buzz,LOW);  Serial.println("FALSE");  }  counter\_indoor\_detected ++;  counter\_indoor\_no\_detect = 0;  }else{  indoor = STATUS\_NO\_HUMAN\_DETECTED;  counter\_indoor\_detected = 0;  counter\_indoor\_no\_detect ++;  }  if(magnetic\_sw == DOOR\_OPEN){  counter\_door\_open ++;  counter\_door\_close = 0;  }else{  counter\_door\_open = 0;  counter\_door\_close ++;  }  if(counter\_outdoor\_detected < 3){  state\_outdoor = "NORMAL";  }else  if(counter\_outdoor\_detected >= 3 && counter\_outdoor\_detected <= 5 ){  state\_outdoor = "SIAGA";  }else  if(counter\_outdoor\_detected > 5){  state\_outdoor = "AWAS";  }    if(counter\_indoor\_detected < 3){  state\_indoor = "NORMAL";  }else  if(counter\_indoor\_detected >= 3 && counter\_indoor\_detected <= 5 ){  state\_indoor = "SIAGA";  }else  if(counter\_indoor\_detected > 5){  state\_indoor = "AWAS";  }  if(counter\_door\_open < 3){  state\_magnetic = "NORMAL";  }else  if(counter\_door\_open >= 3 && counter\_door\_open <= 5 ){  state\_magnetic = "SIAGA";  }else  if(counter\_door\_open > 5){  state\_magnetic = "AWAS";  }  state = state\_outdoor + "\_" + state\_indoor + "\_" + state\_magnetic ;    data = state + "/" + outdoor + "/" + indoor + "/" + ussrf + "/" + magnetic\_sw + "/" + API\_KEY + "/";  return data;  } |
|  | int bukaKoneksi(){  Serial.print("Mencoba sambungan ke server http://");  Serial.println(namaServer);  Serial.println("Mohon menunggu . . . ");  if(myEthernet.connect(namaServer,80)){  Serial.println("Sambungan ke server berhasil!");  return STATUS\_CONNECTED;  }else{  Serial.print("Sambungan ke server gagal!");  Serial.println();  return STATUS\_DISCONNECTED;  }  } |
|  | void kirim\_data(String data){  Serial.println("Menjalankan perintah kirim data");  int ln = data.length();  String uri\_segment;  uri\_segment = "/keamananrumah/index.php/api/post\_sensor\_data/" + data;  myEthernet.print("GET ");  myEthernet.print(uri\_segment);  Serial.print("Data yang dikirim di ke server : ");  Serial.println(data);  myEthernet.println(" HTTP/1.1");  myEthernet.print( "Host: " );  myEthernet.println(" 169.254.2.183 \r\n");  Serial.println("Host OK");  myEthernet.println( "Content-Type: application/x-www-form-urlencoded \r\n" );  Serial.println("Content type OK");  myEthernet.print( "Content-Length: " );  myEthernet.print(ln);  myEthernet.print(" \r\n");  myEthernet.println( "Connection: close" );  myEthernet.println();  String res;  res = baca\_response\_web();  if(res.equals("")==false){  Serial.print("Response server : ");  Serial.println(res);  String raw\_secure\_key = string\_spliter(res, '-', 1);  secure\_key = string\_spliter(raw\_secure\_key, '#', 0);  String raw\_monitoring = string\_spliter(res, '-', 2);  monitoring = string\_spliter(raw\_monitoring, '#', 0);  Serial.print("Secure key : ");  Serial.println(secure\_key);  }  } |
|  | String baca\_response\_web(){  unsigned int time;  Serial.println("Baca respon dari server . . . ");  Serial.println("Mohon menunggu . . . ");  time = millis();  Serial.print("Timer Millis () : ");  Serial.println(time);  int stringPos = 0;  memset( &inString, 0, 1024 );  int unvailable\_ctr = 0;  while(true){  if (myEthernet.available()) {  char c = myEthernet.read();  Serial.print(c);  if (c == '#' ) {  Serial.print("Menemukan start key # dengan isi : ");  startRead = true;  }else if(startRead){  if(c != '^'){  inString[stringPos] = c;  stringPos ++;  }else{  startRead = false;  Serial.println();  Serial.println("Baca respon dari server selesai!");  myEthernet.stop();  myEthernet.flush();  Serial.println("Sambungan diputuskan . . . ");  return inString;  }  }  }else{  delay(50);  unvailable\_ctr++;  if(unvailable\_ctr == 25){  myEthernet.stop();  myEthernet.flush();  Serial.println("Koneksi mengalami time out");  Serial.println("Sambungan diputuskan . . . ");  Serial.println("Reset...");  return inString;  }  }  }  } |

Keterangan:

1. Node 1 merupakan Deklarasi Librasi
2. Node 2 Inisialissi
3. Node 3 *Looping* untuk mengulang seluruh proses pengiriman data
4. Node 4 merupakan kondisi membaca pergerakan manusia pada sensor PIR di luar rumah, PIR di dalam rumah, membaca jarak dari sensor ultrasonic dan, keadaan pintu terbuka atau tertutup dari *magnetic switch*.
5. Node 5 Proses *Fuzzy* *Logic*.
6. Node 6 merupakan kondisi untuk membuka koneksi agar data dapat dikirim ke server
7. Node 7 merupakan fungsi untuk mengirim data.
8. Node 8 berfungsi membaca respon dari web server.

Berikut ini adalah gambar *Flow Graph Notation* dalam melakukan *white box testing*:



**Gambar 4.12 *Flow Graph Notation***

1. Keterangan Gambar:

Node (N) : 8

Edge (E) : 8

1. Perhitungan
2. V(G) = E – N + 2

Dimana: E = 10

V(G) = 8 – 8 + 2

= 2

Jadi hasil perhitungan *Cyclomatic Complexity* pada gambar 4.12 adalah 2. Berdasarkan *Cyclomatic Complexity* tersebut, maka terdapat 2 path atau jalur yang terdiri dari :

Path 1 : 1,2,3,4,5,6,7,8

Path 2 : 1,2,3,4,5,6,7,3

Jadi setelah didapatkan path nya, berikut ini adalah cara pengujian nya:

Berikut ini pengujian dengan *graph matriks*. *Graph Matriks* merupakan matrik empat persegi yang mempunyai ukuran yang sama dengan jumlah node pada *flow graph*.

**Tabel 4.5 *Graph Matriks***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Jumlah** | | | | | | | | | 1 |

V(G) = Jumlah *Graph Matriks* + 1

V(G) = 1 + 1

V(G) = 2

Sehingga, nilai pafa flowgraph metriks berjumlah 2.