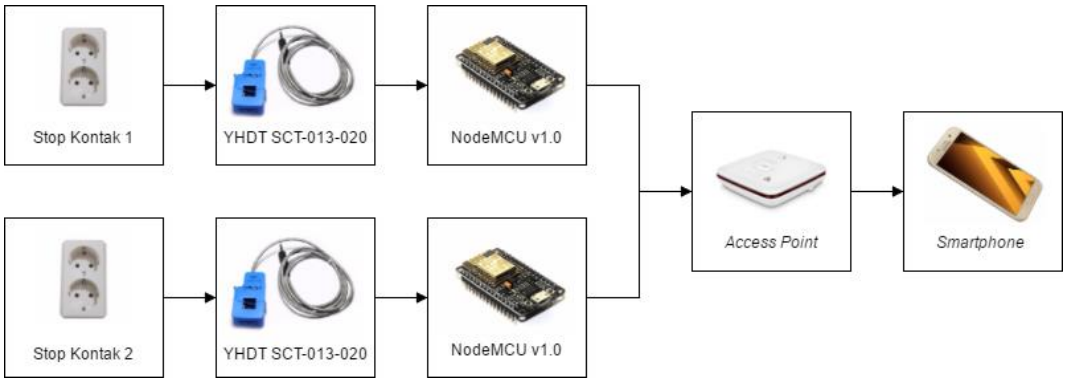
BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas tentang perancangan dan implementasi dari penelitian ini. Tahap perancangan sistem merupakan tahap pembuatan berdasarkan analisis yang telah dibuat dan dilakukan sebelumnya. Dengan perancangan sistem ini diharapkan pengguna bisa dapat lebih mengerti tentang kegunaan sistem.

5.1 Diagram Blok Sistem

Pada diagram blok sistem ini akan dijelaskan mengenai langkah-langkah yang dilakukan dalam proses pembuatan Implementasi *Pervasive Computing* Pada Sistem Monitoring Konsumsi Daya Listrik Stop Kontak Rumah.



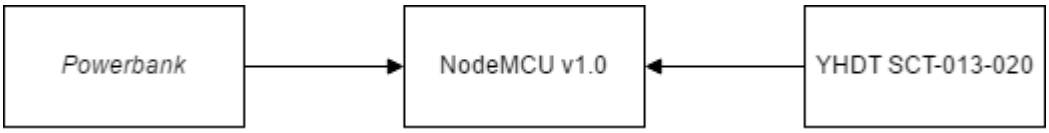
Gambar 5.1 Diagram blok sistem

Gambaran umum penelitian ini dapat dilihat pada gambar 5.1 Diagram Blok Sistem. Sistem ini menggunakan mikrokontroler *NodeMCU v1.0* yang akan mengatur keseluruhan sistem pada Implementasi *Pervasive Computing* Pada Sistem Monitoring Konsumsi Daya Listrik Stop Kontak Rumah. Mula-mula NodeMCU v1.0 dan *smartphone* harus terhubung dalam sebuah jaringan yang sama dengan media router yang telah disiapkan. Arus listrik yang mengalir pada stop kontak akan dideteksi oleh sensor arus YHDT SCT-013-020. Hasil data baca sensor arus akan diolah dan diproses oleh mikrokontroler *NodeMCU v1.0*. Aplikasi di *smartphone* akan melakukan *scanning subnet mask* IP untuk mendeteksi perangkat dengan sistem *pervasive* dalam satu jaringan yang sama. Ketika perangkat *pervasive* telah terdeteksi maka akan ditampilkan ke dalam bentuk daftar pada aplikasi *smartphone*.

5.2 Perancangan Sistem

Sub-bab ini menjelaskan tentang perancangan sistem pada bagian perangkat keras dan perangkat lunak dari Implementasi *Pervasive Computing* Pada Sistem Monitoring Konsumsi Daya Listrik Stop Kontak Rumah.

5.2.1 Perancangan Perangkat Keras

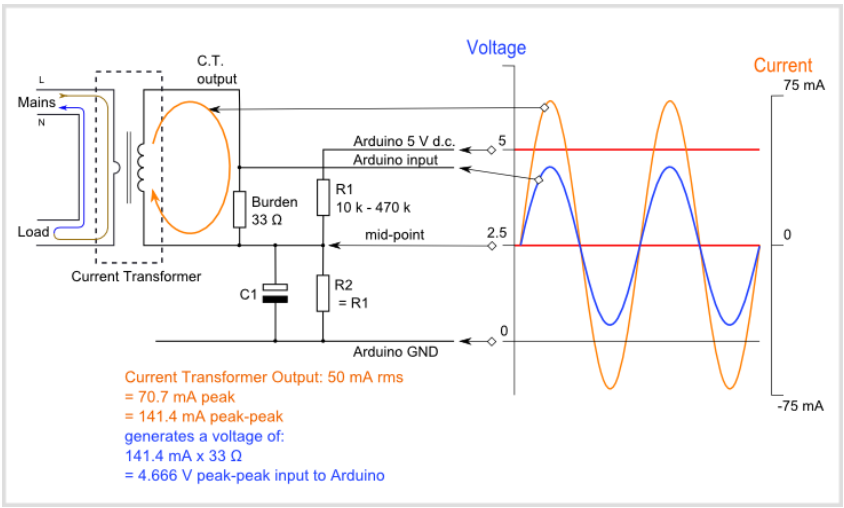


Gambar 5.2 Diagram blok perancangan perangkat keras

Pada Gambar 5.2 dijelaskan tentang perancangan perangkat keras pada penelitian ini. Mikrokontroler yang digunakan adalah NodeMCU v1.0 yang menggunakan powerbank sebagai sumber daya utama untuk menjalankan fungsi sistem. Daya yang dibutuhkan oleh nodeMCU v1.0 untuk dapat beroperasi adalah sebesar 3.3V. Ketika diaktifkan nodeMCU v1.0 akan mengambil data *output* dari sensor arus YHDT SCT-013-020. *Output* dari YHDC SCT-013-020 adalah output analog, oleh karena itu dihubungkan ke nodeMCU v1.0 melalui pin A0 yang merupakan pin ADC *(Analog to Digital Converter)* agar dapat diolah oleh nodeMCU v1.0 dalam bentuk *output* data digital. Sensor arus YHDT SCT-013-020 akan membaca *output* data baca berupa arus dari stop kontak yang diukur oleh sensor.

5.2.2 Penyambungan Sensor Arus YHDT SCT-013-020 ke NodeMCU v1.0

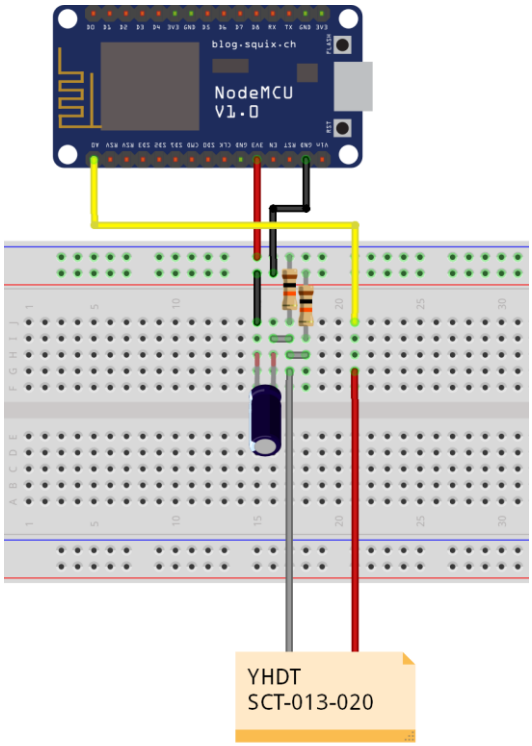
YHDC SCT-013-020 adalah sensor arus yang digunakan untuk mengukur arus pada listrik dengan output analog. Sensor arus YHDT SCT-013-020 memiliki dua kutub, yaitu positif dan negatif. YHDT SCT-013-020 memiliki rentang pengukuran arus dari 0-20 A. Untuk dapat menyambungkan sensor arus YHDT SCT-013-020 dengan nodeMCU v1.0, *output* analog dari YHDT SCT-013-020 harus dikondisikan terlebih dahulu agar kompatibel dengan syarat *input* analog nodeMCU v1.0.



Gambar 5.3 Penyambungan sensor arus YHDT SCT-013-020 ke nodemcu v1.0

Dapat dilihat pada Gambar 5.3 proses penyambungan sensor Arus YHDT SCT-013-020 ke NodeMCU v1.0, dibutuhkan kapasitor, pembagi tegangan dan *burden resistor*. Jika sensor yang digunakan adalah sensor dengan *output* berupa arus, sinyal arus perlu untuk dikonversi ke dalam bentuk sinyal tegangan dengan sebuah

burden resistor. Jika sensor yang digunakan adalah sensor dengan *output* berupa tegangan maka langkah konversi dapat dilewatkan dan burden resistor tidak dibutuhkan karena burden resistor telah terintegrasi di dalam sensor arus tersebut. Sensor arus yang digunakan disini adalah sensor arus YHDT SCT-013-020 dengan *output* berupa tegangan dan *input* berupa arus listrik sehingga tidak perlu dilakukan penghitungan *burden resistor*.



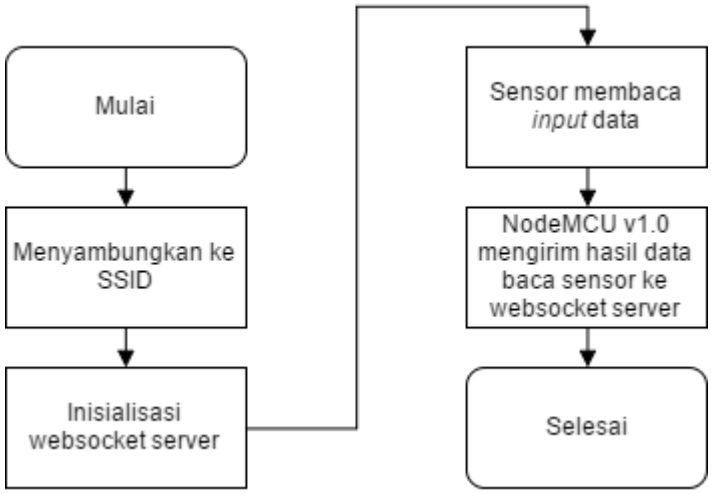
Gambar 5.4 Diagram skematik perancangan perangkat keras

Gambar 5.4 merupakan diagram skematik perancangan perangkat keras yang akan diimplementasikan. NodeMCU v1.0 memperoleh sumber daya utama dari *powerbank* dengan tegangan sebesar 3,3V.*Input* pin A0 pada nodeMCU v1.0 dihubungkan dengan kutub positif dari sensor arus YHDT SCT-013-020. Kapasitor, kutub negatif dan resistor akan dihubungkan ke bagian ground nodeMCU v1.0.

5.2.3 Perancangan Perangkat Lunak

Sub-bab ini menjelaskan perancangan perangkat lunak sistem pada perangkat keras yang digunakan dan perangkat lunak yang digunakan sebagai program utama pada sistem ini. Perancangan perangkat lunak memiliki beberapa proses yang dijabarkan dalam bentuk *flowchart*.

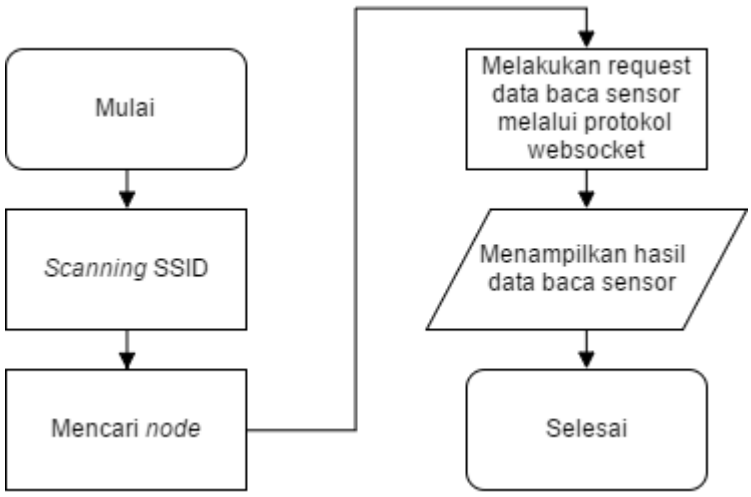
5.2.3.1 *Flowchart* Perancangan Perangkat Lunak Sistem Perangkat *Node*



Gambar 5.5 *Flowchart* perancangan perangkat lunak sistem perangkat keras

Pada Gambar 5.5 *Flowchart* Perancangan Perangkat Lunak *Node* menjelaskan tentang proses perancangan perangkat lunak pada sistem perangkat keras. Ketika perangkat keras diaktifkan diperlukan proses penyambungan ke SSID yang telah ditentukan dan melakukan inisialisasi websocket *server*. Kemudian sensor yang terdapat pada perangkat keras akan membaca *output* berupa arus listrik dari stop kontak yang terhubung dengan perangkat keras. Setelah hasil data baca sensor didapatkan, data tersebut akan dikirimkan oleh nodeMCU v1.0 ke websocket *server* melalui protokol websocket.

5.2.3.2 *Flowchart* Perancangan Perangkat Lunak Aplikasi Android

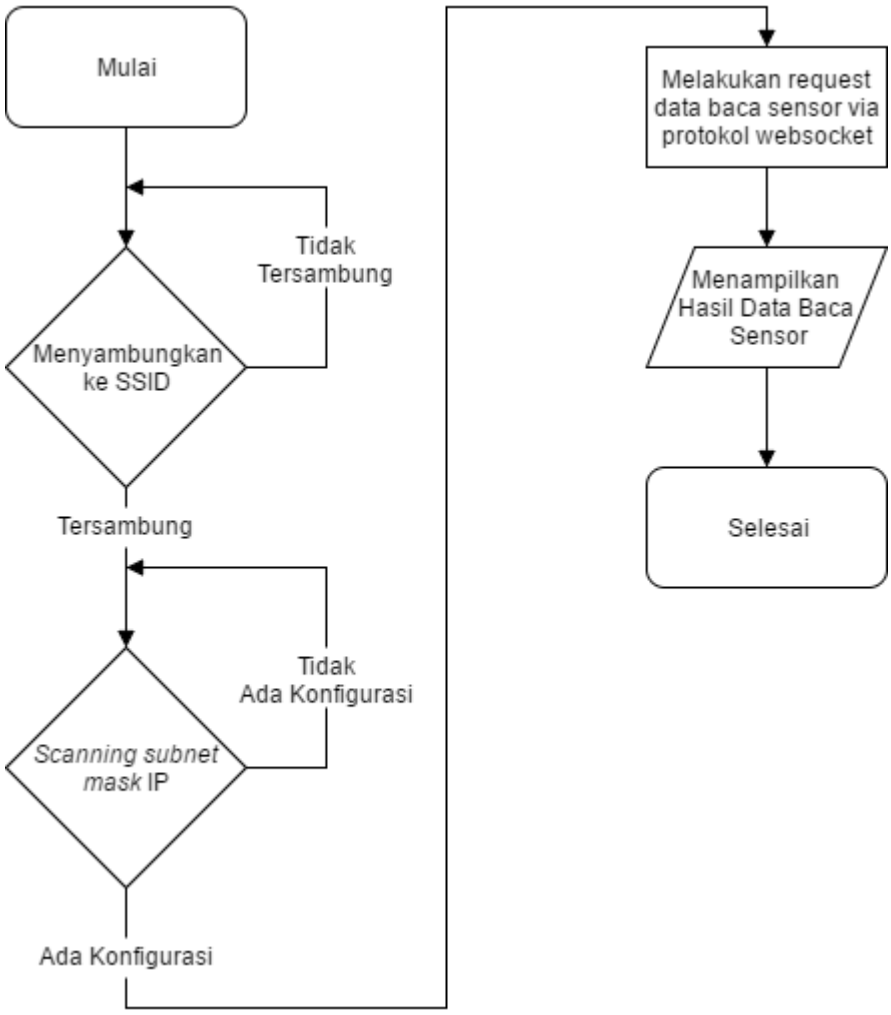


Gambar 5.6 *Flowchart* perancangan perangkat lunak aplikasi android

Pada Gambar 5.6 *Flowchart* Perancangan Perangkat Lunak Aplikasi menjelaskan tentang proses perancangan perangkat lunak pada aplikasi sistem. Aplikasi android berfungsi sebagai antarmuka pengguna untuk memantau sistem monitoring. Pada awal aplikasi dijalankan, diperlukan proses *scanning* SSID tertentu untuk dapat terhubung dengan sistem perangkat keras. Setelah

terhubung ke SSID, aplikasi akan mencari node yang terhubung melalui protokol websocket pada *port* 81. Ketika perangkat keras yang aktif terdeteksi oleh aplikasi, aplikasi akan melakukan *request* hasil data baca sensor pada node tersebut melalui protokol websocket. Setelah hasil data baca sensor diterima, aplikasi akan menampilkan hasil data baca sensor tersebut kepada pengguna.

5.2.3.3 *Flowchart* Perancangan Metode *Pervasive Computing*



Gambar 5.7 *Flowchart* perancangan metode *pervasive computing*

Pada Gambar 5.7 merupakan *flowchart* metode *pervasive computing* yang dirancang pada penelitian ini. Ketika aplikasi dijalankan, aplikasi akan menghubungkan *smartphone* ke SSID yang ditentukan. Ketika *smartphone* telah terhubung ke SSID yang telah ditentukan, aplikasi akan melakukan *scanning* pada *subnet mask* pada alamat IP mulai dari alamat IP ke 1 hingga 255. Apabila terdapat alamat IP yang memiliki konfigurasi pada port 81, maka aplikasi akan mendeteksi alamat tersebut sebagai alamat dari perangkat node yang terhubung pada jaringan tersebut.

5.3 Implementasi Sistem

Pada sub-bab ini akan dijelaskan tentang penerapan perancangan perangkat lunak dan perangkat keras.

5.3.1 Implementasi Perangkat Keras

Tahap ini menjelaskan proses pengimplementasian perangkat keras. Sistem ini menggunakan beberapa komponen perangkat keras yang memiliki fungsi masing-masing.

Gambar 5.8 Implementasi perangkat keras

Pada Gambar 5.8 menunjukkan hasil implementasi sensor YHDT-SCT-013-020 dengan nodeMCU v1.0, dimana terdapat komponen pendukung seperti 2 buah resistor 10kΩ sebagai pembagi tegangan dan kapasitor 10µF sebagai penyimpan daya sementara. NodeMCU v1.0 berfungsi sebagai mikrokontroler yang mengatur *input* dan *output* dari sistem perangkat keras. Sensor arus YHDT-SCT-013-020 digunakan untuk mendeteksi arus listrik yang mengalir pada stop kontak. Sistem ini menggunakan *powerbank* sebagai sumber daya listrik dengan tegangan sebesar 3,3V.

5.3.2 Implementasi Perangkat Lunak

Pada implementasi perangkat lunak akan dijelaskan proses mewujudkan sistem pada perangkat lunak yang telah dilakukan pada perancangan perangkat lunak. Dalam melakukan implementasi peragnkat lunak, sistem ini menggunakan Arduino IDE dan Android Studio untuk proses pemrograman perangkat lunak. Arduino IDE digunakan untuk memprogram sistem dari perangkat keras, sedangkan Android Studio digunakan untuk memprogram sistem dari aplikasi *smartphone* dengan sistem operasi android.

5.3.2.1 Implementasi Perangkat Lunak Pada Arduino IDE

Pada awal program dilakukan inisialisasi *library* dengan tujuan untuk mempermudah pemrograman beberapa fungsi tertentu. Pada tabel 5.1 ditunjukkan pengimplementasian *library* sistem monitoring konsumsi daya listrik

stop kontak rumah, diantaranya adalah *library* “ESP8266WiFi.h” untuk penggunaan .

Tabel 5.1 Kode pemrograman inisialisasi *library* sistem monitoring konsumsi daya listrik stop kontak rumah

Baris	Kode program
1	`#include <ESP8266WiFi.h>`
2	`#include <ESP8266WiFiMulti.h>`
3	`#include <ESP8266WebServer.h>`
4	`#include <WebSocketsServer.h>`
5	`#include <Hash.h>`
6	`#include <EmonLib.h>`

Setelah melakukan insialisasi *library*, sistem akan menjalankan program untuk menyambung ke sebuah SSID yang telah ditentukan. Kode pemrograman pada baris ke-2 digunakan untuk deklarasi nama SSID yang dituju, sedangkan pada baris ke-3 digunakan untuk deklarasi password dari SSID tersebut.

Tabel 5.2 Kode pemrograman menyambungkan ke SSID

Baris	Kode program
1	`ESP8266WiFiMulti WiFiMulti;`
2	`const char* ssid     = "VodafoneSharingDock_4B4E24";`
3	`const char* password = "password";`

Setelah tersambung dengan SSID, sistem akan menjalankan program untuk membuat HTTP *server* dan websocket *server*. Kode pemrograman pada baris pertama digunakan untuk deklarasi HTTP *server* pada port 19105. Kode pemrograman pada baris ke-2 digunakan untuk deklarasi websocket *server* pada port 81, sedangkan pada baris ke-3 merupakan deklarasi variabel node_2 sebagai stop kontak 2.

Tabel 5.3 Kode pemrograman HTTP *server* dan websocket *server*

Baris	Kode program
1	`ESP8266WebServer server(19105);`
2	`WebSocketsServer webSocket = WebSocketsServer(81);`
3	`const char* node_2 = "Stop Kontak 2";`

Ketika HTTP *server* dan websocket *server* telah dibuat, sistem akan menjalankan kode program untuk mengambil nilai hasil data baca sensor dan mengirim hasi data baca tersebut ke *client* melalui protokol websocket. Pada baris ke-4 sampai baris ke-7 menjelaskan apabila websocket tidak terhubung maka akan menampilkan status “Disconnected!”. Pada baris ke-7 sampai baris ke-11 menjelaskan apabila websocket terhubung maka akan menampilkan “Connected from (alamat IP yag terhubung)”. Sedangkan baris ke-14 sampai ke-18 digunakan untuk mengolah hasil output dari sensor arus YHDT SCT-013-020 ke dalam bentuk

tipe data string dengan satuan *ampere* kemudian mengirimkan data tersebut kepada *client* melalui protokol websocket.

Tabel 5.4 Kode pemrograman pembacaan dan pengiriman hasil data baca sensor kepada *client*

Baris	Kode program
1	`void webSocketEvent(uint8_t num, WStype_t type, uint8_t *`
2	`payload, size_t lenght) {`
3	`  switch (type) {`
4	`    case WStype_DISCONNECTED:`
5	`      Serial.printf("[%u] Disconnected!\n", num);`
6	`      break;`
7	`    case WStype_CONNECTED:`
8	`      {`
9	`        IPAddress ip = webSocket.remoteIP(num);`
10	`        Serial.printf("[%u] Connected from %d.%d.%d.%d`
11	`url: %s\n", num, ip[0], ip[1], ip[2], ip[3], payload);`
12	
13	`        // send message to client`
14	`        double Irms = emon1.calcIrms(1480); //Calculate`
15	`Irms only`
16	`        sens = String(Irms);`
17	`        webSocket.sendTXT(num, sens);`
18	`      }`

5.3.2.2 Implementasi Perangkat Lunak Pada Android Studio

Aplikasi pada sistem ini menggunakan Wi-Fi sebagai media untuk dapat berkomunikasi dengan perangkat *pervasive* melalui protokol websocket dan HTTP. Sebelum aplikasi dapat mengimplementasi layanan tersebut, diperlukan konfigurasi persyaratan izin mekanisme aplikasi. Aplikasi harus memiliki izin penggunaan Wi-Fi yang diletakkan pada AndroidManifest.xml.

Tabel 5.5 Konfigurasi izin pada AndroidManifest.xml

Baris	Kode program
1	`<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />`
2	`<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />`
3	`<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>`
4	`<uses-permission android:name="android.permission.INTERNET"/>`

Dengan izin ini aplikasi dapat mengakses perangkat keras yang terhubung ke *router* melalui jaringan Wi-Fi yang digunakan. Adapun penggunaan *library* dari luar yang harus dimasukkan pada build.gradle ditunjukkan pada tabel 5.6.

Tabel 5.6 Konfigurasi *library* pada build.gradle

Baris	Kode program
1	`compile 'com.android.volley:volley:1.0.0'`
2	`compile 'com.android.support:appcompat-v7:25.3.1'`
3	`compile 'com.larswerkman:HoloColorPicker:1.5'`
4	`compile 'tech.gusavila92:java-android-websocket-client:1.2.0'`

Setelah syarat konfigurasi telah selesai dikonfigurasikan, langkah selanjutnya adalah pengimplementasian aplikasi terhadap perangkat keras yang terhubung melalui *router*.

Tabel 5.7 Metode *pervasive computing*

Baris	Kode program
1	`private void scanHost() {`
2	`    WiFiManager mWiFiManager = (WiFiManager)`
3	`getSystemService(Context.WIFI_SERVICE);`
4	`    WiFiInfo  mWiFiInfo =`
5	`mWiFiManager.getConnectionInfo();`
6	`    String subnet =`
7	`getSubnetAddress(mWiFiManager.getDhcpInfo().gateway);`
8	`    Log.d("TEST",subnet);`
9	`    checkHosts(subnet);`
10	`}`
11	
12	`private String getSubnetAddress(int address) {`
13	`    String ipString = String.format(`
14	`            "%d.%d.%d",`
15	`            (address & 0xff),`
16	`            (address >> 8 & 0xff),`
17	`            (address >> 16 & 0xff));`
18	
19	`    return ipString;`
20	`}`
21	
22	`private void checkHosts(String subnet) {`
23	`    adapter.clearData();`
24	`    adapter.notifyDataSetChanged();`
25	`    for (int i = 1; i < 255; i++) {`
26	`        final String host = subnet + "." + i;`
27	`        new Thread(new Runnable() {`
28	`            public void run() {`
29	`                try {`
30	`                    Socket socket = new Socket();`
31	`                    socket.connect(new`
32	`InetSocketAddress(host, 19105), timeout);`
33	`                    socket.close();`
34	`                    Log.d("TEST", "checkHosts() :: " +`
35	`host + " is reachable");`
36	`                    String url = "http://" + host +`
37	`":19105/";`

```
                    StringRequest stringRequest = new
StringRequest(Request.Method.GET, url,
                            new
Response.Listener<String>() {
                                public void
onResponse(String response) {
                                    try {
                                        SmartCurrent
newBulb = new SmartCurrent(response, host);

adapter.addData(newBulb);

adapter.notifyDataSetChanged();
                                    } catch (Exception ex)
{

ex.printStackTrace();
                                    }
                                }
                            }, new
Response.ErrorListener() {
                        @Override
                        public void
onErrorResponse(VolleyError error) {
                            Log.d("TEST", "Fail to Get
Name ::" + host);
                        }
                    });
                    queue.add(stringRequest);
                } catch (ConnectException ce) {
                    Log.i("TEST", "Fail ::" + host);
                } catch (Exception ex) {
                    //ex.printStackTrace();
                }
            }
        }).start();
    }
}

@Override
public void onRequestPermissionsResult(int requestCode,
String[] permissions, int[] grantResults) {
    if (requestCode == 1 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

    }
    if (requestCode == 2 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
        scanHost();
    }

}
```

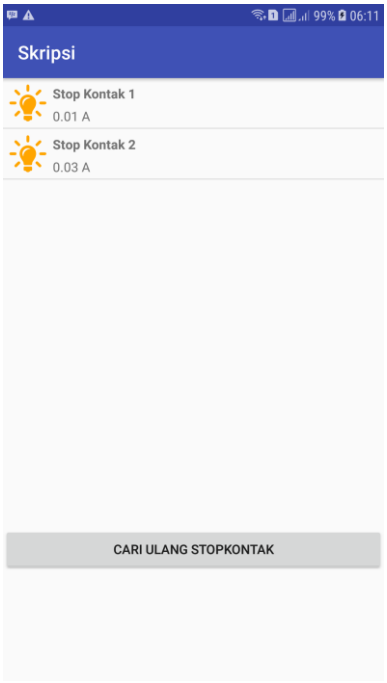
Dengan kode program pada tabel 5.7 inilah yang melakukan implementasi metode *pervasive computing*. Aplikasi mengimplementasikan metode *pervasive computing* dengan cara melakukan *scanning* pada *subnet mask* di jaringan tersebut secara manual mulai dari 1 hingga 255 pada HTTP *port* 19105.

Tabel 5.8 Fungsi *class* pada *Splash Activity*

Baris	Kode program
1	`public class Splash extends Activity {`
2	`    @Override`
3	`    public void onCreate(Bundle savedInstanceState) {`
4	`        super.onCreate(savedInstanceState);`
5	`        setContentView(R.layout.activity_splash);`
6	
7	`        int secondsDelayed = 2;`
8	`        new Handler().postDelayed(new Runnable() {`
9	`            public void run() {`
10	`                startActivity(new Intent(Splash.this,`
11	`MainActivity.class));`
12	`                finish();`
13	`            }`
14	`        }, secondsDelayed * 1000);`
15	`    }`
16	`}`

Tabel 5.9 Tambahan baris kode untuk AndroidManifest.xml

Baris	Kode program
1	`<activity android:name=".Splash"`
2	
3	`android:theme="@android:style/Theme.NoTitleBar.Fullscreen"`
4	`>`
5	`    <intent-filter>`
6	`        <action android:name="android.intent.action.MAIN"`
7	`/>`
8	`        <category`
9	`android:name="android.intent.category.LAUNCHER" />`
10	`    </intent-filter>`
11	`</activity>`



Gambar 5.9 Tampilan awal aplikasi

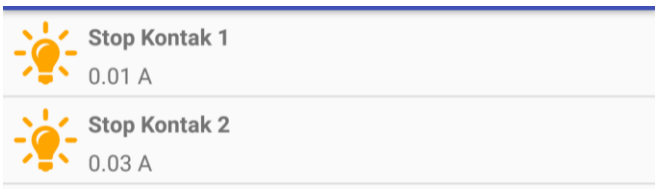
Pada dua tabel di atas, terdapat fungsi *class* pada *Splash Activity* yang digunakan untuk mendesain dan mengatur tampilan awal pada saat aplikasi dinyalakan.

Tabel 5.10 Inisialisasi pada komponen *listview* tampilan awal

Baris	Kode program
1	`protected void onCreate(Bundle savedInstanceState) {`
2	`super.onCreate(savedInstanceState);`
3	`setContentView(R.layout.activity_main);`
4	`queue = Volley.newRequestQueue(this);`
5	
6	`final ListView listView = (ListView)`
7	`findViewById(R.id.stopkontak_listview);`
8	`adapter = new SmartCurrentAdapter(this);`
9	`listView.setAdapter(adapter);`
10	`listView.setOnItemClickListener(new`
11	`AdapterView.OnItemClickListener() {`
12	`@Override`
13	`public void onItemClick(AdapterView<?>`
14	`adapterView, View view, int i, long l) {`
15	`Intent itemIntent = new`
16	`Intent(MainActivity.this, Monitoring.class);`
17	`startActivity(itemIntent);`
18	`}`
19	`});`
20	
21	`});`
22	`}`
23	`} catch (InterruptedException e) {`
24	`}`
25	`}`
26	`};`

```
27
28              t.start();
29
30          }
```

Pada kode diatas, terdapat fungsi untuk menampilkan perangkat keras yang terdeteksi oleh sistem. Perangkat keras yang terdeteksi akan diurutkan dalam sebuah daftar menurun.



Gambar 5.10 *ListView* dengan informasi perangkat keras yang terhubung

Contoh *ListView* dapat dilihat pada gambar 5.8. Daftar aplikasi ditampilkan berdasarkan perangkat node yang telah terdeteksi. Data yang ditampilkan berupa nama daftar nama perangkat node dan nilai dari arus yang diukur.