# BAB II

# TINJAUAN PUSTAKA

## 2.1 Smart Dog Feeder

*Smart Dog Feeder* atau pemberi makan anjing otomatis merupakan jawaban bagi pemilik anjing yang sibuk. Alat ini memberikan fitur pemberian makan otomatis sesuai dengan yang diinginkan oleh penggunanya. Saat ini terdapat beberapa model pemberi makan anjing yang dapat dibeli di pasaran. Berbagai merek dan fitur ditawarkan sebagai daya saing antara produk satu dengan yang lainnya.

PetSafe merupakan salah satu perusahaan yang menjual pemberi makan anjing otomatis. Salah satu jenis yang dipasarkan adalah Healthy Pet Simply Feed$^{TM}$ 12-Meal Automatic Pet Feeder [3]. Alat ini dapat diprogram untuk memberi makan sampai dengan dua belas kali per hari. Pengguna Healthy Pet Simply Feed$^{TM}$ 12-Meal Automatic Pet Feeder juga dapat mengatur porsi per sajian dari 1/8 *cup* sampai dengen 4 *cups*. Pemberi makan ini dapat menyimpan sebanyak 24 *cups* makanan anjing yang berjenis kering. Pengguna dapat memilih untuk menggunakan baterai atau adaptor sebagai sumber daya dari alat tersebut. Healthy Pet Simply Feed$^{TM}$ 12-Meal Automatic Pet Feeder diperuntukkan bagi anjing berukuran kecil sampai sedang. Cara kerja pemberi makan otomatis dari PetSafe tersebut adalah dengan mengatur jadwal dan jumlah porsi dengan menekan tombol yang disediakan. Pada jadwal yang telah diatur, alat tersebut akan mengeluarkan makanan untuk kemudian dimakan oleh anjing peliharaan.

Perusahaan lain yang memproduksi pemberi makan anjing otomatis adalah Petmate. Dengan merek Aspen Pet Lebistro Programmable Food Dispenser, Petmate mendesain alat tersebut khusus untuk pemilik anjing yang sibuk [2]. Namun, alat ini hanya memberi makan sampai dengan tiga kali sehari. Hal tersebut disebabkan Petmate berpegang bahwa anjing yang sehat dapat diraih dengan pemberian porsi makan yang terkontrol.

Aspen Pet Lebistro Programmable Food Dispenser dioperasikan hanya dengan sumber daya baterai. Cara penggunaan alat ini mirip dengan alat sebelumnya, yakni mengatur jam dan besar porsi makan dengan menekan tombol yang disediakan. Pemberi makan otomatis dari Petmate dapat menyimpan makanan sebanyak 5 lb atau kurang lebih 9,5 *cups*.

*Automatic Feeder* lainnya dipasarkan oleh Arf Pets yang memiliki reputasi sebagai *best seller* di situs jual beli internasional Amazon [1]. Alat ini memiliki fitur yang sama dengan kedua alat sebelumnya. Pemberi makan otomatis ini beroperasi dengan mengandalkan sistem alarm yang diatur sebelumnya sesuai dengan pilihan pengguna. Pengguna dapat mengatur jadwal sampai dengan empat kali sehari dan dengan sepuluh jenis porsi per sajian. Arf Pets beroperasi dengan sumber daya dari baterai atau dengan adaptor. Keunikan alat ini adalah pengguna dapat merekam suara sebagai sarana untuk memanggil anjing pengguna. Rekaman dapat diambil sampai dengan 10 detik. Makanan yang dapat digunakan pada alat tersebut adalah jenis makanan kering.

Ketiga pemberi makan anjing otomatis diatas memiliki kelebihan dan kekurangan pada masing-masing produk. Ketiganya memiliki kekurangan pada bagian konfigurasi jadwal bila dibandingkan dengan fitur yang ditawarkan *Smart*

*Dog Feeder* yang dapat melakukan konfigurasi secara *remote,* tidak dapat melakukan *remote monitoring* dan tidak adanya fitur tambahan otentikasi. Pengguna harus melakukan pengaturan manual dengan menekan tombol-tombol yang disediakan pada ketiga produk tersebut. Sedangkan pada *Smart Dog Feeder,* pengguna dapat melakukan konfigurasi hanya dengan membuka aplikasi Appliance Hub pada *smartphone*-nya. Selain itu, pengguna dapat mengawasi kegiatan pemberian makan pada fungsi *log* pada aplikasi tersebut. Penulis melihat adanya perkembangan penggunaan *smartphone* yang terus meningkat tiap tahun. Di Indonesia, pengguna *smartphone* yang aktif pada tahun 2016 mencapai 69.4 juta [5]. *Smartphone* merupakan salah satu perangkat yang telah menjadi bagian dalam hidup sehari-hari. Oleh karena itu, konfigurasi dan pengawasan paling mudah bila dilakukan pada *smartphone* pengguna di tengah kesibukkannya. Hal ini ditunjukkan dengan munculnya berbagai studi mengenai kegunaan *smartphone* untuk membantu kegiatan pengawasan dan pengaturan [6] [7] [8].

Smart Dog Feeder dirancang untuk mempermudah pengguna untuk mengatur jadwal makan anjing mereka. Alat ini dilengkapi fitur otentikasi, penyimpanan data di *database,* dan memiliki aplikasi berbasis Android. Otentikasi digunakan untuk melakukan verifikasi bahwa anjing yang berhak menggunakan alat tersebut hanya anjing milik pengguna. Proses ini melibatkan RFID yang akan dikenakan pada kalung anjing pengguna. Penjadwalan dan penentuan porsi dilakukan dengan menggunakan perangkat pintar Android yang telah terlebih dulu melakukan instalasi aplikasi Appliance Hub. Setiap perubahan yang dibuat pengguna akan diperiksa dan disimpan dalam database server. Informasi pribadi berupa password akan dilindungi dengan menggunakan

SHA2-256 dan komunikasi menggunakan SSL sehingga seluruh data terenkripsi dan tidak mudah dibaca oleh pihak yang tidak bertanggung jawab.

Pada waktu yang dijadwalkan, Smart Dog Feeder akan mengeluarkan suara untuk menarik perhatian anjing. Kemudian anjing pemilik datang dan otomatis diotentikasi dari kalung RFID yang digunakan. Setelah proses otentikasi selesai, alat akan mengeluarkan makanan sejumlah porsi yang telah diatur pengguna. Saat makanan telah dihidangkan, alat akan mengirimkan laporan ke server untuk didata. Apabila telah lewat setengah jam dari waktu penyajian makanan, alat akan mengecek status sajian. Baik makanan sudah habis atau tersisa, Smart Dog Feeder akan melaporkan ke server untuk dicatat.

Porsi yang ditetapkan pada *Smart Dog Feeder* diambil dari banyaknya porsi makan yang dianjurkan pada salah satu merek makanan anjing, Pedigree. Penulis mengambil contoh dari salah satu produk yang digunakan untuk anjing yang masih kecil. Hal ini dilakukan dengan alasan *Smart Dog Feeder* memang diperuntukkan bagi anjing yang masih kecil sampai dengan sedang. Pada produk tersebut tertera beberapa porsi makan bagi anjing sesuai dengan beratnya. Penulis mengambil jenis pertama yang paling cocok digunakan pada *Smart Dog Feeder* yang dapat dilihat pada Tabel 2.1.

***Tabel 2. 1 Pembagian porsi dari Pedigree* [9]**

| **WEIGHT OF PUPPY** | **CUPS PER DAY** | | |
|---|---|---|---|
| | < 3 Months | 3-6 Months | 6-9 Months |
| **Up to 5 lbs.** | Up to 1 1/4 | 2/3 to 1 | 1/2 to 1 |
| **5 to 10 lbs.** | 1 to 2 | 1 to 1 3/4 | 3/4 to 1 ½ |

**1 cup = 8 oz. Measuring cup, 1 can = 13.2 oz.*

## 2.2 Radio Frequency Identification (RFID)

Radio *frequency identification* atau yang dikenal dengan RFID adalah sebutan umum untuk sistem yang mengirimkan identitas berupa angka serial yang unik dari sebuah objek, benda ataupun manusia, dengan menggunakan gelombang radio [10]. RFID merupakan salah satu jenis dalam kategori teknologi *automatic identification* (Auto-ID). Dalam *automatic identification*, hal yang menjadi fokus utama ialah mengurangi pendeteksian sesuatu secara manual yang mengharuskan seseorang untuk melakukan pengecekan satu per satu. Dengan teknologi identifikasi otomatis ini, tingkat ketelitian pengecekan dapat dinaikkan dan mengurangi waktu yang dibutuhkan.

Selain RFID, dalam teknologi Auto-ID terdapat pula yang dinamakan dengan sistem *bar code*. Namun, sistem *bar code* ini tidak dapat digunakan secara efisien seperti RFID. RFID menghadirkan identifikasi jarak jauh dan menyediakan ID unik yang lebih banyak dibandingkan *bar code*. RFID *tag* dapat menyimpan data sebesar 2 sampai dengan 8 *kilobytes* [11] (512 *bits* menyimpan kurang lebih 64 karakter [12]) dan *bar code* rata-rata kurang dari 2000 karakter [13]. Dengan RFID, pengguna dapat membedakan sebuah barang berdasarkan tipe, lokasi, atau data pembuat atau pemilik tanpa bantuan manusia.

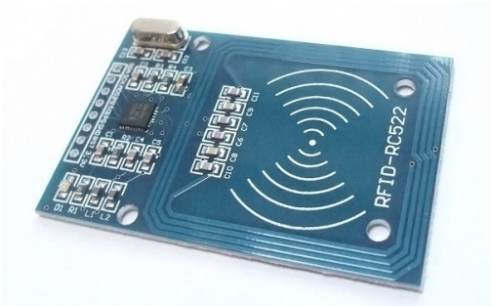
Sistem RFID terdiri dari 3 komponen, yakni antena, *tranceiver,* dan *transponder* [14]. Transponder merupakan *microchip* dengan dilengkapi antena yang akan disematkan pada objek yang ingin diidentifikasi dan transponder, atau yang disebut RFID *tag*, tersebut akan dibaca oleh sebuah *device* dengan menggunakan gelombang radio. Sebagian besar RFID *tags* menggunakan *microchip* berbahan silikon yang dapat menyimpan nomor serial unik dan

beberapa informasi tambahan. Cip tersebut dapat menyimpan data sebanyak 2 *kilobytes*.

Secara garis besar, RFID dapat dibedakan menjadi dua kategori, yakni pasif dan aktif. Sistem RFID aktif biasa digunakan pada objek yang luas atau besar dan mencakup pada area yang luas. RFID jenis ini beroperasi pada frekuensi 455 MHz, 2.45 GHz, atau 5.8 GHz. Dengan frekuensi tersebut, RFID ini dapat membaca dengan jarak dari 20 meter sampai dengan 100 meter. Sedangkan sistem RFID pasif tidak memiliki sumber daya sendiri dan tidak memiliki *transmitter*. RFID pasif memiliki jarak baca lebih pendek dibandingkan RFID aktif, yakni hanya sampai jarak 9 meter. Transponder RFID pasif terdiri dari *microchip* yang dihubungkan dengan antena dan dapat bekerja pada frekuensi rendah, tinggi, dan *ultra-high frequency*. Pada frekuensi rendah, RFID biasa beroperasi pada 124 kHz, 125 kHz atau 135 kHz. Untuk frekuensi tinggi, sistem menggunakan 13.56 MHz dan pada *ultra-high frequency* memakai frekuensi di 860 MHz sampai 960 MHz.

Pada penelitian ini, penulis memakai MFRC522 RFID *reader* untuk membaca *tag* pada kalung anjing. MFRC522 dapat dilihat pada Gambar 2.1 dan memiliki spesifikasi sebagai berikut [15].

1) Beroperasi pada frekuensi 13.56MHz;
2) Tegangan operasi 3.3V;
3) Jarak baca sampai dengan 3 cm;
4) Komunikasi dengan komunikasi SPI.




***Gambar 2. 1MFRC522 RFID Reader*** **[15]**

## 2.3 Android

Android merupakan sistem operasi perangkat *mobile* yang berbasis kernel Linux dan saat ini dikembangkan oleh Google. Sistem operasi ini sedang banyak digunakan oleh banyak pengguna perangkat *mobile* karena kemudahan dan berbasis *open-source* sehingga pengguna dapat melakukan variasi pengaturan pada perangkatnya masing-masing [16].

Pengembangan aplikasi Android dilakukan dengan bahasa pemrograman Java dan dijalankan pada *Dalvik Virtual Machine*. *Dalvik Virtual Machine* merupakan *virtual machine* yang menjalankan aplikasi dan kode yang dituliskan dalam bahasa pemrograman Java. Pada saat mengeksekusi kode dalam Java, kode tersebut akan diubah menjadi *Bytecode* kemudian diubah kembali menjadi *.dex file* sehingga *Dalvik VM* dapat membaca dan menggunakannya. *Dalvik executable files* (.dex) digunakan dengan tujuan untuk mengoptimalkan menggunaan memori dengan *threading* dan *low-level memory management* yang telah dioptimalkan. Hal ini membuat perangkat Android dapat menjalankan berbagai kondisi dengan lebih efisien [16].

Untuk membangun sebuah aplikasi Android, dibutuhkan enam komponen sebagai berikut.

1) *Activities*, yang merepresentasikan sebuah halaman antarmuka yang dilihat oleh pengguna. *Activity* dibangun dari *Views* ataupun *ViewGroups* yang secara langsung bertanggung jawab untuk menampilkan keseluruhan tampilan UI pada layar dan merespon gerakan pengguna.
2) *Services,* merupakan komponen yang bekerja dibelakang layar untuk merespon permintaan data dan menggerakkan notifikasi.
3) *Content Providers,* yang bertugas dalam pengelolaan dan mendistribusikan data pada aplikasi.
4) *Intents,* merupakan *message-passing framework* yang dapat mengirimkan pesan antar *Activity* sehingga dapat dipakai untuk kebutuhan *Activity*.
5) *Broadcast Receivers,* merupakan bagian yang bertugas untuk mendeteksi adanya *Intent* dan memastikan aplikasi mendapatkan *Intent* yang sesuai.
6) *Notifications*, merupakan kerangka notifikasi yang bertugas untuk mengirimkan sinyal kepada pengguna bila mendapatkan suatu data dari *Services* atau *Broadcast Receiver*.

Keseluruhan komponen yang membangun aplikasi akan dinyatakan pada Android Manifest. Android Manifest akan mendata komponen aplikasi, pengaturan keamanan, dan *classes*. Seluruh data pada Android Manifest akan berupa XML yang memiliki *tag* utama, yaitu *application, uses-permission, permission,* dan *instrumentation*. *Tag application* akan menyatakan *activity* apa saja yang terdapat pada aplikasi, *service* yang digunakan, *provider* untuk *Content Providers* yang dipakai, dan *receiver* untuk mendaftarkan *Broadcast Receiver*. *Tag uses-permission* mengatur izin yang akan diberikan pada aplikasi seperti izin mengakses lokasi yang mengaktifkan GPS. Sedangkan *tag permission* digunakan untuk membatasi akses aplikasi. *Tag instrimentation*

menyediakan kerangka *running tests* yang digunakan saat menjalankan aplikasi sehingga dapat melihat jalannya interaksi antara aplikasi dengan sistem.

## 2.4 Arduino UNO

Arduino UNO merupakan mikrokontroler yang berbasis Atmega328P yang memiliki 14 input/output digital (6 diantaranya dapat digunakan sebagai keluaran PWM), 6 input analog, sebuah 16 MHz ceramic resonator, sebuah konektor USB, sebuah *power jack*, sebuah ICSP header, dan sebuah tombol *reset* [17].

### 2.4.1 Spesifikasi

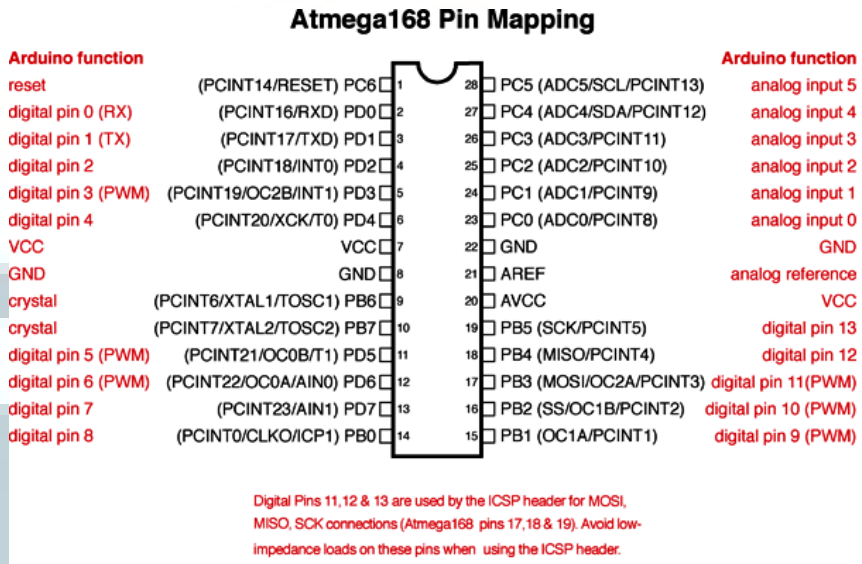
Berikut adalah spesifikasi teknik dari Arduino UNO yang dapat dilihat pada Gambar 2.2.

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P)<br>of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

***Gambar 2. 2 Technical Specs dari Arduino* UNO [17]**

### 2.4.2 Pin Mapping

Berikut adalah *pin mapping* dari Arduino UNO yang dapat dilihat pada Gambar 2.3.



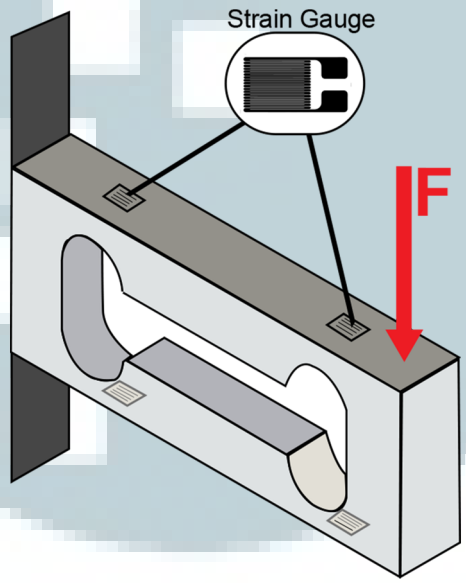

***Gambar 2. 3 Pin Mapping Arduino UNO* [18]**

## 2.5 Load Cell

Load Cell merupakan *transducer* yang digunakan untuk membuat sinyal listrik yang berbanding lurus dengan tekanan yang diukur. Sinyal yang dihasilkan ini dapat berupa perubahan tegangan, arus atau frekuensi. Load Cell memiliki beberapa jenis seperti *hydraulic load cells, pneumatic load cells,* dan *strain gauge load cells.* Dari beberapa jenis tersebut, *load cell* yang umum dipakai adalah *strain gauge load cells.*

*Strain Gauge* mengukur tekanan sebagai perubahan resistansi listrik yang merupakan hasil pengukuran dari tekanan yang diberikan seperti yang dicontohkan pada Gambar 2.4. Sebuah *load cell* umumnya terdiri dari empat *strain gauge* di dalam sebuah *Wheatstone bridge*. *Strain Gauge* merupakan

konduktor yang diatur dalam pola zigzag pada permukaan sebuah membran. Ketika membran tersebut meregang, resistansinya akan meningkat. Dalam banyak kasus, *strain gauge* yang digunakan berjumlah 4 buah untuk mencapai tingkat sensitifitas yang maksimum. Dua buah *gauge* bertugas pada tegangan dan dua lainnya untuk tekanan. Umumnya *strain gauge* akan dipasang pada daerah yang akan mengalami perbedaan tekanan untuk akurasi. Saat beban diberikan, *strain* akan mengubah resistansi listrik menjadi sebanding dengan tegangan yang diberikan pada *load cell*.




***Gambar 2. 4 Cara Kerja Load Cell* [19]**

## 2.6 Paho Android Service

*Paho Android Service* merupakan antarmuka dari *Paho Java MQTT Client* untuk *platform* Android [20]. Dengan MQTTAndroidClient, komunikasi MQTT akan ditangani oleh *Android-Service* pada latar belakang aplikasi.

Untuk melakukan instalasi, pada *Gradle* aplikasi ditambahkan *Paho Android Service* agar aplikasi dapat menggunakan komunikasi MQTT dari Paho seperti pada Gambar 2.5.

```
repositories {
    maven {
        url "https://repo.eclipse.org/content/repositories/paho-releases/"
    }
}


dependencies {
    compile('org.eclipse.paho:org.eclipse.paho.android.service:1.0.2') {
        exclude module: 'support-v4'
    }
}
```

***Gambar 2. 5 Bagian Gradle dengan Paho Android Service*** **[20]**

Penggunaan *service* ini membutuhkan beberapa *uses-permission* dan *service* yang harus ditambahkan pada bagian AndroidManifest.xml. Penambahan pada file manifestasi dapat dilihat pada Gambar 2.6.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.evw.appliancehub">

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ApplianceHub"
        android:theme="@style/AppTheme">
        <service android:name="org.eclipse.paho.android.service.MqttService" />

        <activity
            android:name=".MainActivity"
            android:theme="@style/FullScreenTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

***Gambar 2. 6 AndroidManifest.xml dari Aplikasi Appliance Hub***

Setelah AndroidManifest.xml ditambahkan segala komponen pendukung Paho Android Service, aplikasi dapat menjalankan komunikasi MQTT.

Komunikasi MQTT pada aplikasi tidak bisa berjalan dengan sendirinya tanpa membuat antarmuka dengan *service*. Dengan membuat MQTTAndroidClient, komunikasi MQTT akan terbangun pada sebuah aplikasi. Format pembuatan MQTTAndroidClient dapat dilihat pada Gambar 2.7. Pada bagian URL merupakan alamat dari server MQTT yang diinginkan. Fungsi *MqttClient.generateClientId()* digunakan untuk membuat *clientID* yang dipakai server untuk membedakan *client*. *MemoryPersistence* merupakan objek dibutuhkan untuk menyimpan data selama komunikasi berlangsung.

```
MqttAndroidClient aClient;

protected MqttAndroidClient onCreate(Context context){

    aClient = new MqttAndroidClient(
            context,
            "tcp://192.168.3.19:1883",//"ssl://uskk6067f009.skumniotprojects.koding.io:3695",
            MqttClient.generateClientId(),
            new MemoryPersistence()
    );
```

***Gambar 2. 7 Fungsi Pembuatan MQTTAndroidClient pada Aplikasi Appliance Hub***

Setelah aplikasi terhubung dengan *service* yang dibutuhkan, aplikasi dapat melakukan koneksi ke server dengan menggunakan fungsi *connect*. Pada fungsi tersebut, koneksi dapat dilengkapi dengan pilihan keamanan seperti *username, password,* ataupun menyertakan sertifikat untuk mendukung komunikasi dengan SSL. Pilihan tambahan tersebut dinyatakan pada *MqttConnectionOptions* seperti yang dapat dilihat pada fungsi *connect* di Gambar 2.8.

Fungsi *getSSLSocketFactory* digunakan untuk menyertakan sertifikat pada saat memulai koneksi. *File* .bks yang merupakan format sertifikat pada Android saat melakukan koneksi SSL didapat dari *ca.crt* yang didapatkan dari

server. Format sertifikat server diubah dengan *keytool* yang dilakukan pada *terminal* sehingga menghasilkan *BouncyCastle keystore* (BKS). Perintah yang dituliskan pada *terminal* dapat dilihat pada Gambar 2.9. Pada pilihan *–file* yang tertulis *iot.eclipse.org.crt* diubah menjadi *ca.crt* dan *iot.eclipse.org.bks* merupakan *file* BKS yang dihasilkan sehingga dapat diubah juga menjadi *client-crt.bks* pada aplikasi Appliance Hub. Sedangkan *file bcprov-ext-jdk14-1.53.jar* merupakan *file* yang dibutuhkan untuk menghasilkan BKS yang dapat diunduh pada *repo2.maven.org* [21].

```
protected boolean connect(final Context context, String username, String password)
        throws IOException, MqttSecurityException, InterruptedException {
    MqttConnectOptions options = new MqttConnectOptions();

    InputStream input =
        context.getApplicationContext().getAssets().open("client-crt.bks");

    options.setSocketFactory(aClient.getSSLSocketFactory(input, "moscak3695"));
    options.setConnectionTimeout(30);
    options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1_1);
    options.setCleanSession(true);
    options.setUserName(username);
    options.setPassword(password.toCharArray());
    Log.d("username", username);
    Log.d("pass",password);

    try {

        IMqttToken token = aClient.connect(options);
        token.setActionCallback(new IMqttActionListener() {
            @Override
            public void onSuccess(IMqttToken asyncActionToken) {
                Log.d("report", "SUCCESS");
                stat = 1;
            }

            @Override
            public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
                Log.d("report", "FAILURE");
                stat =0;
            }
        });
    }
    catch (MqttException e) {
        e.printStackTrace();
    }
    Thread.sleep(1000);

    if (stat==1){
        return true;
    }
    else{
        return false;
    }
}
```

***Gambar 2. 8 Fungsi Connect pada Aplikasi Appliance Hub***

```
keytool -import -alias eclipse.broker -file iot.eclipse.org.crt -keypass eclipse-password -keystore iot.eclipse.org.bks  -storetype BKS -storepass eclipse-password -providerClass org.bouncycastle.jce.provider.BouncyCastleProvider -providerpath bcprov-ext-jdk14-1.53.jar
```

***Gambar 2. 9 Perintah Pembuatan File BouncyCastle keystore (BKS)*** **[20]**

Komunikasi antara *client* dan *server* pada MQTT berdasarkan pada konsep *publish-subscribe* pada sebuah topik yang ditentukan. *Client* yang melakukan *subscribe* pada topik tertentu akan menerima semua pesan yang di-*publish* ke topik tersebut. *Server* bertugas untuk mengelola proses *publish-subscribe* tersebut. *Server* akan melakukan *broadcast* ke seluruh *client* sehingga masing-masing *client* mendapatkan pesan yang ter-*publish.* Proses *subscribe* dapat dilakukan dengan cara seperti pada Gambar 2.10. Sedangkan proses *publish* dapat dilakukan seperti pada Gambar 2.11.

```
protected void subscribe(final Context context, String topic, int QOS) {
    try {
        IMqttToken token = aClient.subscribe(topic, QOS, null, new IMqttActionListener() {
            @Override
            public void onSuccess(IMqttToken iMqttToken) {
                Toast.makeText(context, "subscription successful", Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onFailure(IMqttToken iMqttToken, Throwable throwable) {
                Toast.makeText(context, "subscription failed: " + throwable, Toast.LENGTH_SHORT).show();
            }
        });

    } catch (MqttException e) {
        Toast.makeText(context, "could not subscribe", Toast.LENGTH_SHORT).show();
    }
}
```

***Gambar 2. 10 Fungsi Subscribe pada MQTTAndroidClient***

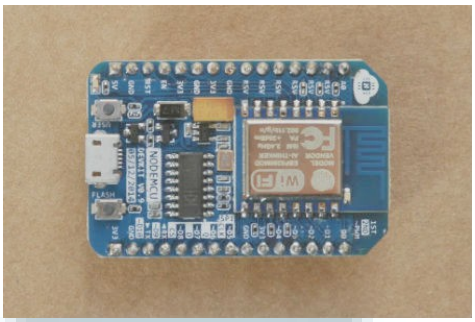
```
protected boolean send(final Context context, int QOS, String JSONStr, String MQTT_TOPIC) {
    if (aClient == null || !aClient.isConnected()) {
        Toast.makeText(context, "sorry, currently not connected...", Toast.LENGTH_SHORT).show();
        return false;
    }

    MqttMessage message = new MqttMessage();

    message.setPayload(JSONStr.getBytes());
    message.setQos(QOS);
    try {
        aClient.publish(MQTT_TOPIC, message, null, new IMqttActionListener() {
            @Override
            public void onSuccess(IMqttToken iMqttToken) {
                Toast.makeText(context, "message sent", Toast.LENGTH_SHORT).show();
                stat = 1;
            }

            @Override
            public void onFailure(IMqttToken iMqttToken, Throwable throwable) {
                Toast.makeText(context, "failed to send message: " + throwable.getMessage(), Toast.LENGTH_SHORT).show();
                stat = 0;
            }
        });
    } catch (MqttException e) {
        Toast.makeText(context, "could not send message to MQTT broker", Toast.LENGTH_SHORT).show();
    }
    if(stat == 1){
        return true;
    }
    else{
        return false;
    }
}
```

*Gambar 2. 11 Fungsi Publish pada MQTTAndroidClient*

## 2.7 Modul WiFi ESP8266

ESP8266 merupakan *integrated chip* yang digunakan sebagai modul WiFi yang dapat menghubungkan suatu perangkat ke internet. Modul ini memiliki *on-board processing* dan kemampuan penyimpanan yang dapat dimanfaatkan untuk beroperasi bersama sensor dan aplikasi lainnya. ESP8266 dioperasikan dengan AT *commands*.

Saat ini, ESP8266 mengalami perkembangan yang cukup pesat. Banyak *development kit* yang dikembangkan untuk digunakan bersama ESP8266. Salah satunya adalah NodeMCU *Development Kit*. Dengan pemrograman LUA, NodeMCU *Development Kit* menggunakan *event-driven API* sehingga mempercepat proses pada aplikasi IoT [22].



*Gambar 2. 12 NodeMCU Development Kit dengan ESP8266* **[22]**

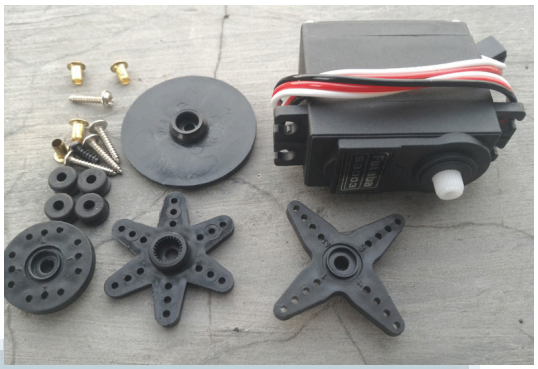
Gambar 2.12 merupakan bentuk NodeMCU *Development Kit* yang berukuran 32 mm x 25 mm. Dengan ukuran yang tidak terlalu besar dan harga yang terjangkau, *board* ini banyak dipakai dalam berbagai projek IoT.

## 2.8 Servo

*Servo* merupakan alat yang menghasilkan keluaran berupa gerakan yang memiliki poros sehingga dapat diatur dengan mengirimkan sinyal [23]. Selama *servo* mendapatkan sinyal tersebut, *servo* akan berputar sampai yang diperintahkan. Pada umumnya *servo* dapat dikontrol dari $0^o$ sampai $180^o$. Namun, beberapa pabrik membuat *servo* yang dapat memutar $210^o$ atupun $360^o$ secara kontinu. Selama *servo* mendapatkan sinyal tersebut, *servo* akan berputar sampai yang diperintahkan.

Pada penelitian ini, *servo* yang dipakai adalah DF15RSMG $360^o$ *Servo Motor* dari DFRobot yang dapat dilihat pada Gambar 2.13. *Servo* ini memiliki spesifikasi sebagai berikut.

1) Ukuran 40 x 40 x 20 mm;
2) Memiliki torsi 19.3 kg/cm di saat berhenti dan torsi 15.1kg/cm di saat bergerak pada 7.4V;
3) Beroperasi pada tegangan 5-7.4V.

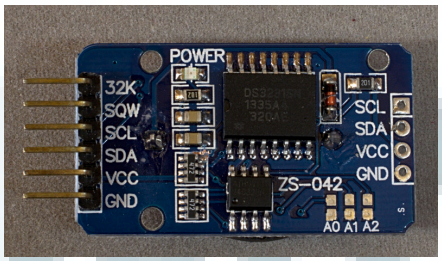


*Gambar 2. 13 DF15RSMG Motor Servo* [23]

## 2.9 Real Time Clock

*Real Time Clock* (RTC) merupakan sebuah jam yang biasanya berbentuk sirkuit terintegrasi yang menggunakan baterai sebagai sumber daya dan berfungsi menyimpan waktu [24]. RTC digunakan untuk memastikan informasi mengenai waktu tidak hilang saat mikrokontroler kehilangan sumber dayanya. Pada *Smart Dog Feeder*, RTC digunakan untuk menyimpan waktu dan sebagai penyedia *interrupt* berupa alarm.

Gambar 2.14 merupakan RTC yang dipakai penulis pada penelitian ini. DS3231 dapat memberikan informasi mengenai detik, menit, jam, hari, tanggal, bulan, dan tahun [25].



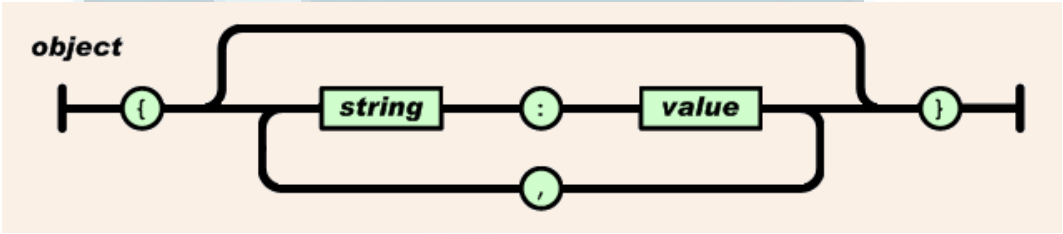

*Gambar 2. 14 DS3231 RTC* [26]

Penulis memakai *library* untuk Arduino untuk memanfaatkan fungsi dari *real time clock* yaitu DS3232RTC.h [27].

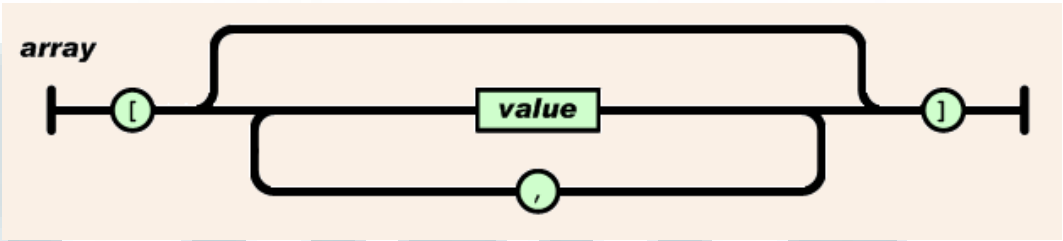
## 2.10 JSON

JSON (*JavaScript Object Notation)* merupakan format pertukaran data yang dapat dengan mudah dimengerti oleh manusia dan ringan [28]. JSON dapat digunakan pada berbagai bahasa pemrograman seperti bahasa C, C++, C#, Java, dan JavaScript. Struktur data JSON dapat terdiri dari kumpulan nama dengan nilai seperti *struct* pada bahasa C atau kumpulan dari daftar nilai seperti *array* pada bahasa pemrograman lain.

Pada penelitian ini JSON dipakai sebagai format pertukaran data dari perangkat ke server, server ke Android, dan Android ke server. Format JSON yang dipakai adalah JSON *object* dan JSON *array*. Pada Gambar 2.15 merupakan format dari JSON *object* dan pada Gambar 2.16 merupakan format dari JSON *array*.




***Gambar 2. 15 Format dari JSON Object*** **[28]**




***Gambar 2. 16 Format dari JSON Array*** **[28]**

## 2.11 Server

*Server* merupakan suatu perangkat atau sebuah program komputer yang memiliki fungsi menyediakan jasa kepada program komputer lainnya yang disebut *client*. Server biasanya mengelola segala kebutuhan *client* yang bersifat spesifik, seperti menyediakan data bagi *client*. Oleh karena itu, server memiliki peran yang sangat penting pada jaringan yang membutuhkan *shared resources* [29].

Pada sistem Appliance Hub, kehadiran *server* dibutuhkan untuk proses pertukaran data maupun penyimpanan data. *Server* yang dipakai berbasis protokol MQTT karena pada sistem ini dibutuhkan waktu yang *real-time* dan ukuran *bandwidth* jaringan yang minimal. MQTT memang sering digunakan dalam komunikasi *machine-to-machine* (M2M) atau *Internet of Things* dan memakai aplikasi *mobile* karena memenuhi kriteria *bandwidth* dan kapasitas baterai [30].

Protokol MQTT yang dipakai oleh *server* memiliki lapisan keamanan untuk memastikan komunikasi perangkat ke server dan aplikasi Android ke server atau sebaliknya tidak dapat dimengerti pihak luar yang tidak berkepentingan. Seluruh komunikasi dari dan menuju ke *server* menggunakan enkripsi data dengan TLS/SSL. Untuk mendapatkan sertifikat yang dibutuhkan dalam komunikasi dengan SSL, *server* menggunakan OpenSSL untuk membuat *self-certificate*.

Pada bagian aplikasi dari *server* menggunakan Node.js. Node.js dipakai karena mampu menangani banyak koneksi yang serentak [31] dan optimal dalam melayani aplikasi web yang *real-time* karena bersifat *event-driven* [30]. *Server* akan mencocokkan pesan yang diterima setiap terdapat *event onPublished* yang

telah dideklarasi kemudian menjalankan penyimpanan atau pembaruan data. Selain berbasis Node.js, *server* pada penelitian ini juga menggunakan modul mosca yang bertugas sebagai *broker* koneksi MQTT. Sedangkan proses penyimpanan atau pembaruan data dilakukan dengan menambahkan modul MySQL sebagai basis data. [30]