BAB III

LANDASAN TEORI

3.1 *Internet of Things* (*IoT*)

Istilah "*Internet of Things"* diperkenalkan oleh Kevin Ashton pada sebuah presentasi di tahun 1999 kepada perusahaan P&G. Beliau adalah salah satu pendiri MIT dan seorang inovator yang memelopori penggunaan RFID yang diterapkan dalam detektor kode batang untuk manajemen rantai pasokan. Selain itu, Kevin Ashton juga mendirikan perusahaan bernama Zensi yang fokus pada teknologi pemantauan energi. Oleh karena itu, Kevin Ashton memiliki peran yang *sign*ifikan dalam pengembangan teknologi *IoT* dan telah memberikan kontribusi dalam bidang pengenalan dan penerapan konsep *IoT*.

IoT menghubungkan objek fisik melalui internet, memungkinkan komunikasi dan interaksi dalam kehidupan sehari-hari. Objek ini dilengkapi sensor, perangkat lunak, dan konektivitas jaringan untuk mengumpulkan dan bertukar data. Data ini digunakan untuk memonitor, mengontrol, dan mengotomatisasi proses. *IoT* berkembang pesat di berbagai bidang, termasuk:

1. *Smart home*: Meningkatkan kenyamanan penghuni
2. *Smart city*: Meningkatkan kualitas hidup penduduk
3. Industri: Meningkatkan efisiensi dan produktivitas produksi
4. Kesehatan: Mengumpulkan data pasien untuk diagnosis, pengobatan, dan perawatan
5. Logistik: Meningkatkan efisiensi pengiriman barang [7].

3.2 *Smart Home*

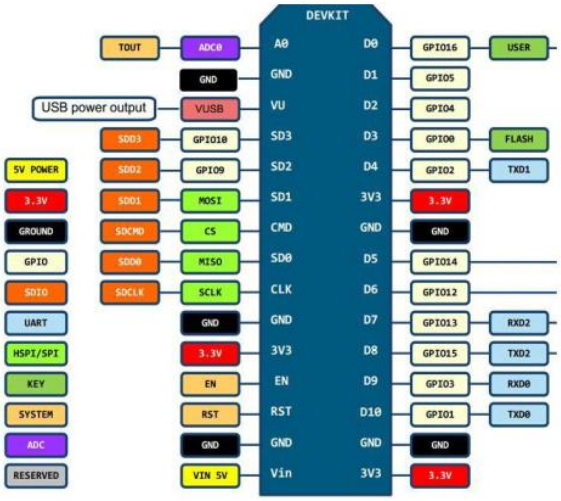
Smart home adalah istilah dalam pengimplementasian teknologi *IoT* pada rumah atau tempat tinggal, di mana perangkat yang digunakan terhubung dengan internet dan dapat beroperasi secara otomatis maupun dikontrol dari jarak yang jauh. Ini memungkinkan penghuni rumah dapat hidup dengan nyaman berkat bantuan perangkat *IoT* yang digunakan dalam rumah, hal ini dapat mencakup otomatisasi perangkat rumah dengan efisiensi, kenyamanan, keamanan, penghematan, dan hiburan pada tempat tinggal [8].

3.3 *Nodemcu* ESP8266

Nodemcu adalah salah satu piranti perangkat keras *opensource* untuk pengembangan *IoT*. Di dalam *Nodemcu* terdapat *System On Chip* (SoC) ESP8266 yang mendukung konektivitas internet melalui jaringan wifi. Awalnya Espressif System yang membuat SoC ESP8266, namun dalam penggunaannya yang sulit karena memerlukan beberapa komponen elektronik lain. *Nodemcu* mengemas *SoC* ESP8266 menjadi 1 papan modul yang memudahkan penggunanya menjadi *Nodemcu* ESP8288, yang dapat dianalogikan sebagai papan tambahan untuk menggunakan SoC ESP8266 agar lebih mudah. Dalam memprogram papan *Nodemcu* tersebut, dapat langsung dihubungkan menggunakan kabel *microUSB*. Lua adalah bahasa pemrograman yang digunakan. Papan ini bekerja layaknya *mikrontroler* yang dapat terhubung ke *wifi* [8].

Nodemcu dapat diprogram menggunakan *Arduino* IDE yang merupakan sebuah *kompiler* untuk *arduino*. Basis pengembangan kit ini menggunakan modul ESP8266, yang menggabungkan berbagai fitur seperti GPIO, PWM, IIC, 1-Wire,

dan ADC dalam satu papan. *Board* GPIO *Nodemcu* ESP8266 memiliki dimensi 4.83cm x 2.54cm dan berat 7 gram. Kit ini juga mendukung mode *deep sleep*, di mana semua komponen seperti CPU, *Clock*, periferal, *Wi-Fi*, dan radio dimatikan, kecuali blok RTC (*Real-Time Clock*). Fitur ini memungkinkan ESP8266 untuk beroperasi dalam jangka waktu yang lama. Fungsi bagian atau komponen GPIO *Nodemcu* ESP8266 v3, sebagai berikut:



Gambar 3.1 GPIO pada *Nodemcu* ESP8266 v3

Untuk mengaktifkan mode *deep sleep*, *pin* GPIO16 (D0) dan EXT_RSTB (RST) harus terhubung secara bersamaan [9],[10].

3.4 *Load cell*

Sensor *load cell* adalah sensor yang dapat mengukur berat suatu benda dengan cara mengubah gaya yang diterima, menjadi resistansi sinyal listrik pada *strain gauge* yang dapat diukur. Sensor berat yang berbentuk batang biasanya terdiri dari 4 kabel, 2 kabel sebagai sumber daya dan 2 kabel lainnya sebagai sinyal keluaran (output). Kabel dari *load cell* dihubungkan dengan modul HX711 sebagai

penguat sinyal keluaran dengan membaca perubahan resistansi menjadi tegangan yang dapat dibaca oleh *microcontroller* [11], [12].

3.5 Sensor ultrasonik HC-SR04

Sensor ultrasonik HC-SR04 adalah sensor yang dapat mengukur jarak 2 cm hingga 4 meter, dengan ketelitian 3 milimeter. Cara kerja sensor ini adalah menghitung selisih waktu kecepatan gelombang saat di pancarkan dan di terima dari objek suatu benda. Sensor ini memancarkan gelombang ultrasonik berfrekuensi 40 kilohertz [13].

3.6 *Buzzer*

Buzzer adalah komponen elektronik yang menghasilkan bunyi dari getaran yang dihasilkan dari tegangan listrik pada taraf tertentu, disesuaikan dengan spesifikasi *buzzer* itu sendiri. Karena penggunaannya yang mudah, biasanya *buzzer* digunakan sebagai alarm. Dengan memberikan tegangan *input*, maka *buzzer* dapat mengubah tegangan itu menjadi getaran dan menghasilkan bunyi yang dapat didengar manusia [14].

3.7 *Arduino* IDE

Arduino IDE adalah perangkat lunak *opensource* yang digunakan sebagai editor untuk menulis kode program, *mengkompilasi*, dan memasukkan (*upload*) kode program yang telah dibuat ke dalam papan *arduino*. *Arduino* sendiri merupakan papan rangkaian elektronik atau kit yang dilengkapi *chip microcontroller* dengan tujuan memudahkan penggunaan *arduino* pada berbagai bidang. Sedangkan kata IDE di belakang nama *software Arduino* IDE adalah, singkatan dari bahasa inggris yaitu *Integrated Development Environment*.

Lingkungan yang terintegrasi untuk melakukan pengembangan, pengembangan di sini ditujukan kepada pengembangan *board arduino*. Kode program yang ditulis disebut sebagai *sketch* dan berformat .ino yang nantinya dilakukan kompilasi sebelum di *upload* ke *board arduino* [15].

3.8 *React Native*

React Native, kerangka kerja yang dikembangkan oleh Meta (sebelumnya dikenal sebagai Facebook), memungkinkan pembuatan aplikasi *multi-platform* (iOS, *Android*, dan Windows) menggunakan *JavaScript*. Node.js digunakan untuk server pengembangan dan memfasilitasi penggunaan alat pengelola dependensi seperti *npm (Node Package Manager)* [16].

Javascript sendiri ialah bahasa pemrograman yang digunakan pada sisi pengguna atau *client*. *Javascript* bisa dikatakan bahasa pemrograman tingkat tinggi, karena *sintaks* yang mirip bahasa manusia sehingga memudahkan untuk dipelajari. Awalnya *javascript* dibuat dengan tujuan agar *website* lebih interaktif, seperti menampilkan dan menghilangkan objek pada *website* [17].

3.9 *Firebase*

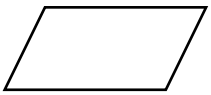
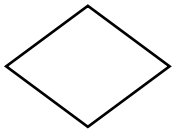
Firebase merupakan layanan yang disediakan oleh *Google*, ditujukan untuk memudahkan para pengembang aplikasi tanpa harus mengeluarkan usaha yang besar. Layanan *firebase* memiliki banyak fitur, di antaranya adalah *Firebase Realtime Database, Firebase Authentication* dan *Firebase Notification* atau yang sekarang menjadi *Firebase Cloud Messaging* [18]. *Firebase Realtime Database* merupakan layanan penyimpanan *database* berbasis awan (*Cloud*) yang berada di server *Firebase*. Menggunakan model basis data JSON, yang memudahkan setiap

client terhubung untuk menyimpan dan mengambil data secara *realtime*. Sedangkan *Firebase Cloud Messaging* atau disingkat *FCM*, adalah layanan pengiriman pesan ke *client* lintas platform yang dapat meningkatkan interaksi pengguna dengan mengirimkan notifikasi ke aplikasi yang dikembangkan [19].

3.10 *Flowchart* diagram

Flowchart adalah representasi grafis dari urutan langkah-langkah dalam suatu proses atau algoritma, menggunakan simbol-simbol standar untuk menggambarkan alur kerja atau logika program [20].

Tabel 3.1 Simbol *Flowchart*

Simbol	Keterangan
	***Terminator*** (oval): Menandakan awal atau akhir proses
	**Proses** (persegi panjang): Menunjukkan langkah atau operasi dalam proses
	***Input*/*Output*** (jajar genjang): Menunjukkan *input* data atau *output* hasil
	**Keputusan** (*diamond*): Menandakan titik pengambilan keputusan atau percabangan
	Arus/Aliran (panah): Menandakan arah aliran proses

3.11 Blok diagram

Blok diagram adalah representasi visual sederhana dari sistem atau proses yang menunjukkan komponen utama dan hubungan antar komponen menggunakan simbol, kotak dan garis [21].

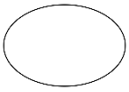
3.12 *Unified Modeling Language*

Unified Modeling Language (UML) adalah metode pemodelan yang ditujukan untuk memvisualisasikan perancangan sistem berorientasi objek. UML juga menetapkan standar untuk membuat *blueprint*/perancangan sistem, yang mencakup konsep proses bisnis, penulisan kelas dalam bahasa pemrograman tertentu, skema *database*, dan komponen yang dibutuhkan dalam sistem perangkat lunak [22]. Berikut ini adalah penjelasan 4 jenis diagram yang digunakan untuk memvisualisasikan model perancangan sistem dalam penelitian ini.

3.12.1 *Use Case* Diagram

Use Case Diagram digunakan untuk menggambarkan fungsionalitas sistem dan interaksi antara unit atau aktor [23].

Tabel 3.2 Simbol *Use Case* Diagram

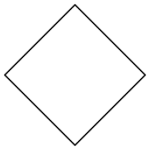
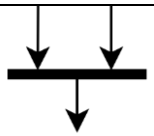
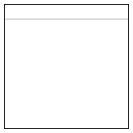
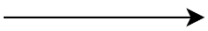
No.	Simbol	Keterangan
1.		*Actor*: Menggambarkan entitas eksternal yang berinteraksi dengan *Use Case*.
2.		*Use Case*: Merepresentasikan fungsionalitas sistem yang ditawarkan kepada aktor.
3.		*Association*: Menggambarkan koneksi antara aktor dan *Use Case* yang terlibat.

4.		System: Simbol kotak yang mewakili sistem.

3.12.2 *Activity* Diagram

Activity Diagram digunakan untuk merepresentasikan alur kerja suatu proses dan menggambarkan urutan aktivitas dalam proses tersebut. [23].

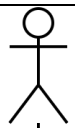
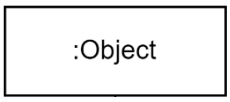
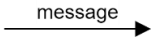
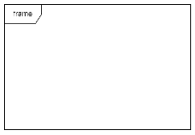
Tabel 3.3 Simbol *Activity* Diagram

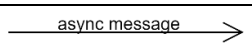
No.	Simbol	Keterangan
1.		Start *point* (status awal): Sebuah diagram aktivitas diawali dengan status awal.
2.		*Activity* (aktivitas): Aktivitas yang dilakukan sistem, biasanya diawali kata kerja
3.		*Decision* (percabangan): Digunakan jika terdapat pilihan aktivitas yang bercabang atau lebih dari satu.
4.		*Join* (penggabungan): Penggabungan dari aktivitas yang lebih dari satu.
5.		*End State* (status akhir): Sebuah diagram aktivitas diakhiri dengan status akhir.
6.		*Swimlane*: Memisahkan organisasi bisnis yang bertanggung jawan terhadap aktivitas yang terjadi.
7.		*Transition*: Menghubungkan ke aktivitas selanjutnya.

3.12.3 *Sequence* Diagram

Sequence Diagram digunakan untuk menggambarkan alur interaksi antar obyek dengan menekankan pada pengaturan waktu pesan-pesan yang dikirimkan [23].

Tabel 3.4 Simbol *Sequence* Diagram

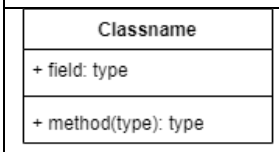
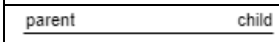
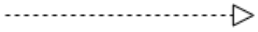
No.	Simbol	Keterangan
1.		*Actor*: *Actor* adalah entitas, baik orang atau sistem eksternal, yang mendapatkan manfaat atau menggunakan sistem.
2.		Objek: Merujuk pada objek yang terlibat dalam sistem.
3.		*Lifeline*: Garis yang menggambarkan masa hidup dari suatu objek dalam diagram.
4.		Activation bars: Ditempatkan pada *lifeline*, yang menggambarkan durasi terjadinya interaksi.
5.		*Synchronous message*: Informasi yang mengalir dari satu objek ke objek lainnya.
6.		*Frame*: Menggambarkan konteks diagram *Sequence*

7.		*Asynchronous message*: Digunakan ketika proses informasi tidak perlu menunggu waktu dari objek lain pada sistem.

3.12.4 *Class* Diagram

Class Diagram adalah representasi visual yang mendetail dari struktur suatu sistem, menggambarkan kelas-kelas, atribut, metode, dan hubungan antar kelas. Diagram ini berfungsi sebagai cetak biru yang memudahkan perencanaan, pengembangan, dan pemahaman sistem yang kompleks [24].

Tabel 3.5 Simbol *Class* Diagram

Simbol	Keterangan
	***Class*:** *Class* merupakan elemen dasar dalam pemrograman berorientasi objek, digambarkan sebagai kotak yang terdiri dari tiga bagian.
	***Association*:** Hubungan yang memperlihatkan interaksi antar *class*. Digambarkan dengan garis berujung panah terbuka yang menunjukkan aliran pesan satu arah.
	***Generalization*:** Hubungan yang menunjukkan relasi dari *class* yang lebih khusus menuju yang lebih umum.