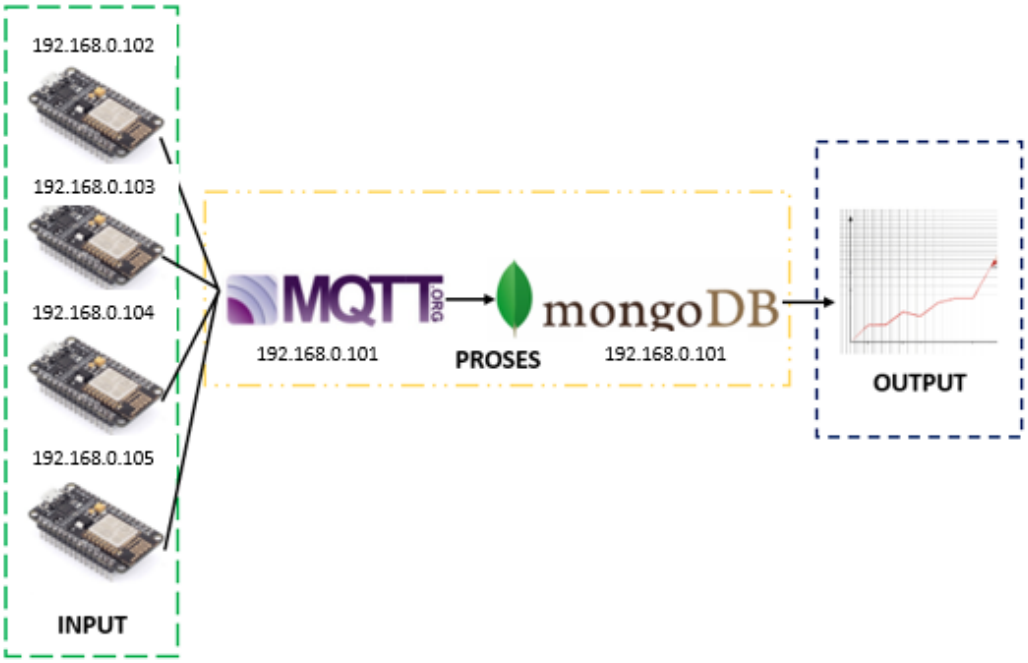
BAB 5 PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini akan menjelaskan tentang perancangan dan implementasi dari sistem pengiriman data yang meliputi tentang perangkat lunak dan perangkat keras yang digunakan oleh sistem.

5.1 Perancangan Sistem

Pada bagian ini dijelaskan, proses perancangan sistem dimulai dari perancangan perangkat keras, perangkat lunak, dan format pesan protokol MQTT.



Gambar 5. 1. Topologi Sistem (diterapkan pada jaringan lokal)

Pada Gambar 5.1 yang merupakan topologi jaringan dari sistem. Kita dapat melihat bahwa data sensor yang berupa *input* akan dikirim melalui NodeMCU. Data yang dikirim akan dimasukkan dan diproses kedalam MQTT *broker* untuk selanjutnya dikirim kedalam basis data MongoDB. Hasil atau *output* akan ditampilkan dalam tampilan grafik untuk memudahkan penggambaran hasil.

5.1.1 Perancangan Perangkat Keras

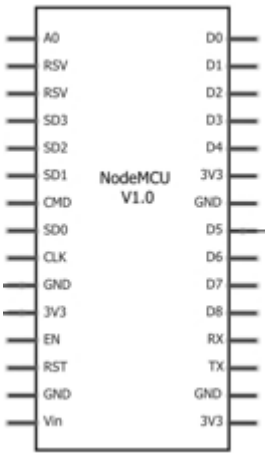
Pada tahap ini dijelaskan proses perancangan perangkat keras. Pada bab sebelumnya dijelaskan bahwa perangkat keras sistem menggunakan mikrokontroler NodeMCU.

5.1.1.1 Perancangan NodeMCU sebagai Client

NodeMCU berperan sebagai perangkat pembuat data sensor, mikrokontroler ini juga mempunyai modul *wireless* yang bertindak sebagai media transmisi data. Tabel 5.1 adalah tabel spesifikasi mikrokontroler NodeMCU, sedangkan gambar 5.2 adalah gambar skematik *board* NodeMCU.

Tabel 5. 1. Spesifikasi NodeMCU ESP8266 12E

Wifi Protocol	802.11 b/g/n
Power Supply	5 V
Security	WPA/WPA2
Frequency Range	2.4G-2.5G (2400M-2483.5M)
Network Protocols	IPv4, TCP/UDP/HTTP/FTP



Gambar 5. 2. Skematik NodeMCU

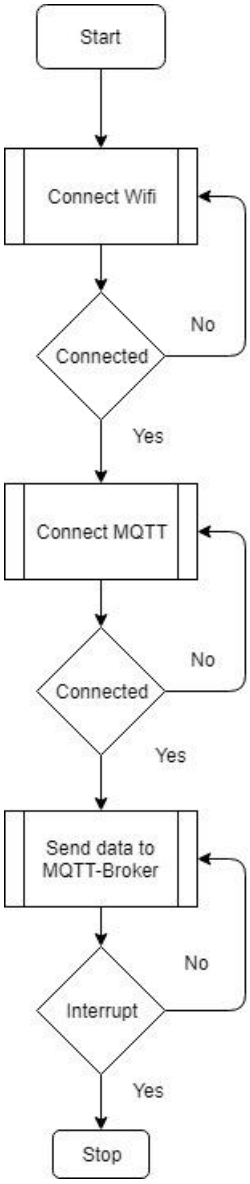
NodeMCU akan dilakukan *upload* program dari Arduino IDE sebagai pengirim data sensor kedalam *broker* MQTT. Programnya berisi tentang nilai suhu dalam satuan Celcius dan akan *generate* secara acak oleh NodeMCU sebagai data pengiriman.

5.1.2 Perancangan Perangkat Lunak

Pada tahap ini dijelaskan proses perancangan perangkat lunak. Pada bab sebelumnya dijelaskan bahwa perangkat lunak sistem terdiri dari arduino IDE untuk kompilasi kode program mikrokontroler, untuk simulasi MQTT menggunakan aplikasi *broker* MQTT, dan untuk sistem basis data menggunakan MongoDB.

5.1.2.1 Perancangan MQTT Client

Perancangan perangkat lunak MQTT *client* mencakup pada fungsionalitas kebutuhan perangkat lunak yang digunakan untuk menulis dan kompilasi program. Pemrograman untuk perangkat MQTT *Publisher* menggunakan bahasa C Arduino dengan software Arduino IDE untuk kompilasinya. *Library* yang digunakan antara lain; *library* MQTT digunakan untuk mengaktifkan fungsi MQTT untuk NodeMCU. *Library* Arduino JSON digunakan untuk mengaktifkan fungsi pengiriman data bertipe JSON untuk dikirim kedalam *broker*.

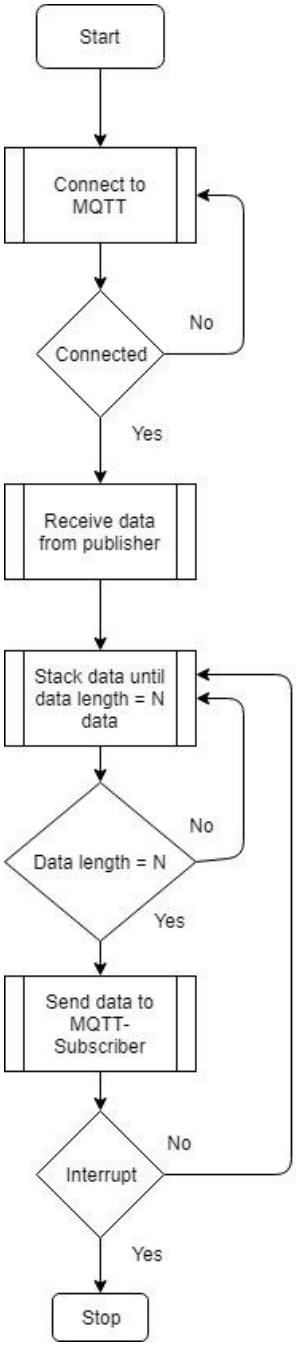


Gambar 5. 3. Flowchart MQTT-Client

Program Arduino pertama kali melakukan koneksi kedalam *wifi*, setelah terkoneksi, maka program akan melakukan koneksi kedalam MQTT-Broker, setelah dapat terkoneksi kedalam MQTT-Broker, program akan melakukan *generate* data sensor supaya nantinya akan diteruskan kedalam MQTT-Broker.

5.1.2.2 Perancangan MQTT Broker

Perancangan perangkat lunak MQTT *Broker* mencakup pada fungsionalitas kebutuhan perangkat lunak yang digunakan untuk menulis dan kompilasi program *broker*. Pemrograman untuk perangkat MQTT *broker* menggunakan bahasa Python dengan software Python untuk kompilasinya. *Library* yang digunakan antara lain; *Library* PyMongo digunakan untuk mengaktifkan fungsi komunikasi Python dan MongoDB.



Gambar 5. 4. Flowchart MQTT-Broker

Program MQTT-Broker bermula dari awal program melakukan koneksi ke MQTT yang berada pada komputer lokal. Jika program sudah terkoneksi ke MQTT, maka program akan menerima data dari *publisher* dan melakukan *stack* data sampai data ke-N, tergantung kebutuhan programnya. Jika Panjang data sudah sampai N-data, maka data *stack* akan dikirimkan kedalam MQTT-Subscriber.

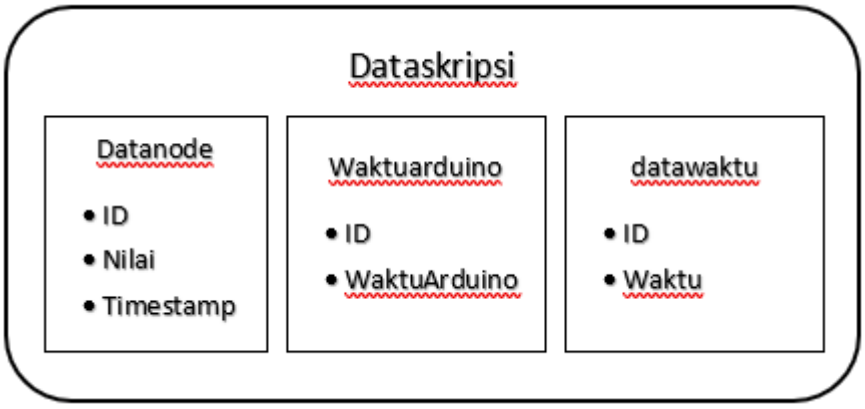
5.1.2.3 Perancangan Database MongoDB

Sistem basis data MongoDB menggunakan versi 3.4 yang di-install didalam *operating system* Windows. Sistem basis data MongoDB akan menerima data yang berisi nilai pembacaan data sensor dan *timestamp* dari MQTT-Broker, dan MongoDB dapat membuat *unique ID* secara otomatis.

RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Index	→	Index
Row	→	Document
Column	→	Field
Join	→	Embedding & Linking

Gambar 5. 5. Perbedaan struktur data MongoDB dan RDBMS

Didalam MongoDB terdapat perbedaan nama dengan sistem basis data SQL. Tabel didalam MongoDB dinamakan *Collection*, baris didalam MongoDB dinamakan *Document*, kolom didalam MongoDB dinamakan *Field*, dan gabungan didalam MongoDB dinamakan *Embedding & Linking*.

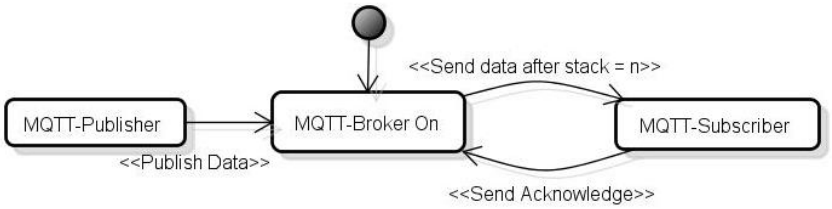


Gambar 5. 6. ERD "Dataskripsi"

Diagram ERD MongoDB untuk perancangan basis data pada gambar 5.6 terdapat satu database yang dinamakan "Dataskripsi" mempunyai collection "Datanode", "WaktuArduino", dan "datawaktu". "Datanode" berisi informasi tentang data pembacaan sensor. "WaktuArduino" mempunyai informasi tentang pencatatan waktu ketika program stack didalam broker berjalan. Dan "datawaktu" mempunyai informasi tentang pencatatan waktu ketika MQTT-Broker mengirim data yang sudah ditumpuk kedalam database.

5.1.2.4 Perancangan Simulasi MQTT

Simulasi MQTT terdiri dari MQTT *publisher*, *broker* dan MQTT *subscriber*. Simulasi dilakukan pada PC/laptop dengan alamat jaringan lokal. Untuk menjalankan simulasi MQTT *publisher*, port serial yang sedang digunakan untuk simulasi tidak dapat digunakan untuk menjalankan aplikasi lain. Sehingga sebelum memulai simulasi pastikan agar port tidak sedang digunakan untuk program lain.



Gambar 5. 7. Block Diagram MQTT

Diagram Block MQTT terdiri dari MQTT *publisher*, *broker* dan *subscriber*. Flowchart ini menjelaskan bagaimana tiap-tiap komponen MQTT berkerja. Pertama program *broker* dinyalakan sebagai penengah antara *publisher* dan *subscriber*, ketika *broker* sudah dinyalakan lalu dijalankan *publisher* dan *subscriber* untuk transmisi data. MQTT-Publisher mem-publish data kedalam broker tetapi *publisher* tidak menyimpan data apa-apa. Apabila data sudah dikirim, data akan ditampung di MQTT-Broker, jika panjang data yang sudah ditampung sudah mencapai N data, maka data akan dikirim ke MQTT-Subscriber dan MQTT-Broker akan mendapat pemberitahuan tentang data yang sudah dikirim. Ketika data sudah dikirim maka data seharusnya sudah diterima oleh MQTT-Subscriber yang menjadi tujuan akhir pengiriman data.

5.2 Implementasi Sistem

Pada bagian ini dijelaskan, proses implementasi sistem dimulai dari perancangan perangkat keras, perangkat lunak, dan protokol MQTT.

5.2.1 Implementasi NodeMCU sebagai Client

Sesuai dengan perancangan perangkat keras MQTT *client*, implementasi MQTT *client* terdiri dari input sensor *module* yaitu mikrokontroler NodeMCU yang terhubung pada komputer. Seperti pada gambar 5.8 yang merupakan 4 buah NodeMCU sebagai MQTT *client*.

Gambar 5. 8. NodeMCU sebagai MQTT-Client

NodeMCU dikoneksikan pada komputer menggunakan kabel USB, kabel USB digunakan untuk menyalakan NodeMCU dan melakukan penggunggahan program kedalamnya. 4 NodeMCU ini berada di *port* yang berbeda pada *slot* komputer supaya dapat digunakan semuanya.

5.2.2 Implementasi MQTT Client

Setelah NodeMCU menyala dan siap digunakan, NodeMCU akan dilakukan unggah program kedalamnya supaya dapat melakukan koneksi ke *wifi* lokal dan menjadikannya sebagai MQTT-Publisher.

Source Code NodeMCU koneksi WIFI	
1	`void setup() {`
2	`// Setup hardware serial for logging`
3	`Serial.begin(HW_UART_SPEED);`
4	`while (!Serial);`
6	`// Setup WiFi network`
7	`WiFi.mode(WIFI_STA);`
8	`WiFi.hostname("ESP_" MQTT_ID);`
9	`WiFi.begin("Ruko Permata Jingga No 11 LT 2", "akbar250188");`
11	`LOG_PRINTFLN("\n");`
12	`LOG_PRINTFLN("Connecting to WiFi");`
13	`while (WiFi.status() != WL_CONNECTED) {`
14	`delay(500);`
15	`LOG_PRINTFLN(".");`
16	`}`
17	`LOG_PRINTFLN("Connected to WiFi");`
18	`LOG_PRINTFLN("IP: %s", WiFi.localIP().toString().c_str());`

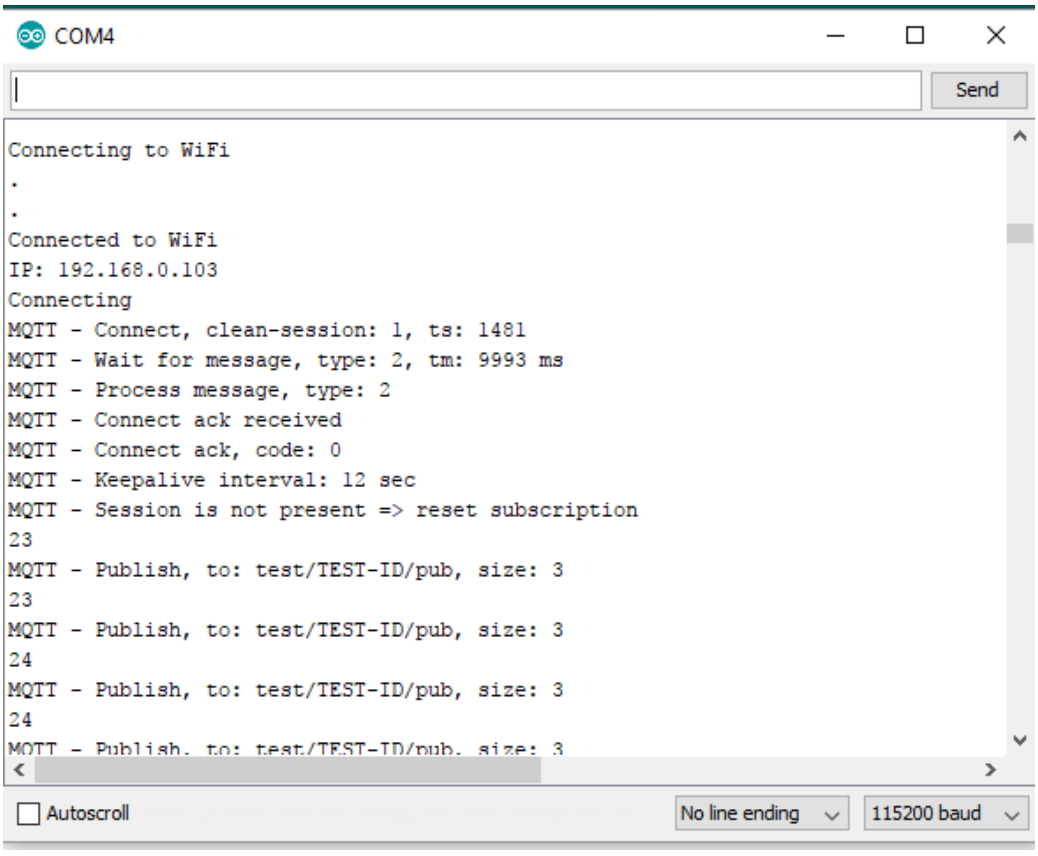
Gambar 5. 9. Source Code NodeMCU koneksi WIFI

NodeMCU harus melakukan konfigurasi SSID dan kata sandi *wifi* supaya dapat melakukan koneksi ke jaringan *wifi*. Setelah dilakukan konfigurasi SSID dan kata sandi *wifi*, dilakukan konfigurasi koneksi protokol TCP untuk akses koneksi ke jaringan komputer.

```
Source Code NodeMCU koneksi TCP
1   void loop() {
2
3   // Check connection status
4   if (!mqtt->isConnected()) {
5           // Close connection if exists
6           network.stop();
7           // Re-establish TCP connection with MQTT broker
8           LOG_PRINTFLN("Connecting");
9           network.connect("192.168.0.107", 1883);
10          if (!network.connected()) {
11              LOG_PRINTFLN("Can't establish the TCP connection");
12             delay(5000);
13             ESP.reset();
14             }
15  // Start new MQTT connection
16  MqttClient::ConnectResult connectResult;
17  // Connect
```

Gambar 5. 10. Source code koneksi TCP NodeMCU

Setelah semua program selesai dikonfigurasi, program di-upload kedalam mikrokontroler NodeMCU dengan memilih tombol *upload* didalam arduino.



Gambar 5. 11. NodeMCU berhasil koneksi ke wifi dan melakukan koneksi TCP

Gambar 5.11 diatas menjelaskan bahwa alamat IP untuk NodeMCU adalah 192.168.0.103. setelah mendapatkan alamat IP, program akan berusaha melakukan koneksi TCP kedalam komputer yang alamat IPnya juga berada didalam *wifi* tersebut. Jika bisa terhubung maka program akan menampilkan pemberitahuan MQTT-Connect. Jika MQTT-Broker belum dinyalakan, program akan menunjukkan kalimat "Session is not present => reset subscription". Jika sudah dinyalakan *brokernya*, maka program akan menunjukkan kalimat "Publish, to: test/TEST-ID/pub, size : 3" yang menandakan bahwa data sudah terkirim.

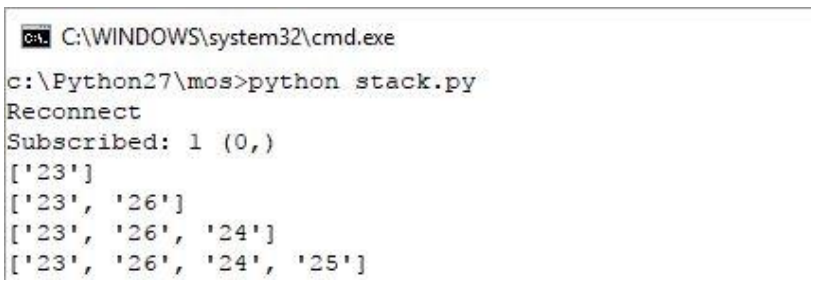
5.2.3 Implementasi MQTT Broker

Sesuai dengan perancangan perangkat keras MQTT *broker*, implementasi MQTT *broker* terdiri dari MQTT *broker* yang berjalan di *Command Prompt* Windows.

Source Code MQTT-Broker	
1	`def on_publish(mqttc, obj, mid):`
2	`print("Published : " + str(mid))`
3	`def on_subscribe(mqttc, obj, mid, granted_qos):`
4	`    print("Subscribed: " + str(mid) + " " + str(granted_qos))`
5	`def on_log(mqttc, obj, level, string):`
6	`    print(string)`
7	`mqttc = mqtt.Client("Broker")`
8	`mqttc.on_message = on_message`
9	`mqttc.on_connect = on_connect`
10	`mqttc.on_publish = on_publish`
11	`mqttc.on_subscribe = on_subscribe`
12	`mqttc.connect("127.0.0.1", 1883)`
13	`mqttc.subscribe("test/TEST-ID/pub", qos=0)`
14	`mqttc.loop_forever()`

Gambar 5. 12. Potongan Source Code Python untuk MQTT-Broker

Gambar 5.12 adalah *source code* python untuk program *broker* MQTT, terdapat kalimat ""127.0.0.1", 1883" yang artinya MQTT-Broker dikonfigurasi didalam *localhost* komputer dan menggunakan *port* 1883.

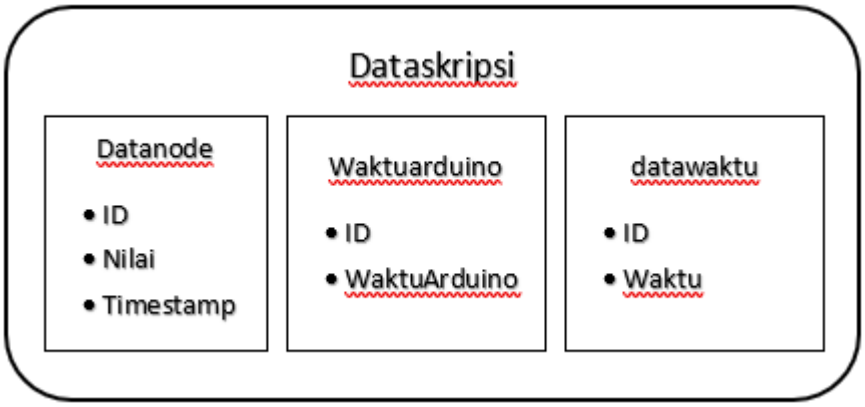


Gambar 5. 13. Program MQTT-Broker dijalankan

MQTT-Broker dijalankan lewat command prompt sebagai media transmisi data, data yang dikirim oleh NodeMCU sebagai *publisher* akan ditimbun sampai ke-N data sesuai programnya. Terdapat tulisan "Reconnect" menunjukkan *broker* mencoba melakukan koneksi ke sistem MQTT, "Subscribed" menunjukkan banyaknya *subscriber* yang melakukan *subscribe* ke *broker* dan angka nol menunjukkan protokol MQTT dengan QoS 0. Dan terdapat angka 23 sampai 26 adalah data yang masuk kedalam MQTT-Broker, lalu data di-stack sebanyak N-data, ketika Panjang data sudah sampai N-data maka *broker* melakukan pengiriman data kedalam *subscriber*.

5.2.4 Implementasi Database MongoDB

Sistem basis data MongoDB yang sudah di-install merupakan versi 3.4 didalam *operating system* Windows. Sistem basis data MongoDB akan menerima data yang berisi nilai pembacaan data sensor dan *timestamp* dari MQTT-Broker, dan MongoDB dapat membuat *unique ID* secara otomatis.



Gambar 5. 14. ERD Database "Dataskripsi"

Diagram ERD MongoDB pada gambar 5.14 terdapat satu database Dataskripsi yang mempunyai collection "Datanode", "WaktuArduino", dan "datawaktu". "Datanode" berisi informasi tentang data pembacaan sensor. "WaktuArduino" mempunyai informasi tentang pencatatan waktu ketika program *stack* didalam *broker* berjalan. Dan "datawaktu" mempunyai informasi tentang pencatatan waktu ketika MQTT-Broker mengirim data yang sudah ditumpuk kedalam *database*.

```
> db.datanode.find().pretty()
{
        "_id" : ObjectId("5a445d63f8e4980e5c469d27"),
        "timestamp" : ISODate("2017-12-28T09:56:31.852Z"),
        "nilai" : "26"
}
{
        "_id" : ObjectId("5a445d63f8e4980e5c469d28"),
        "timestamp" : ISODate("2017-12-28T09:56:31.923Z"),
        "nilai" : "25"
}
```

Gambar 5. 15. Collection "Datanode"

"Datanode" adalah *collection* yang terdapat didalam basis data dataskripsi yang berisikan data ID, timestamp dan nilai yang dikirimkan oleh MQTT-Publisher.

```
> db.WaktuArduino.find().pretty()
{
        "_id" : ObjectId("5a445d63f8e4980e5c469d5a"),
        "WaktuArduino" : 3.4660000801086426
}
```

Gambar 5. 16. Collection "WaktuArduino"

"WaktuArduino" adalah *collection* yang terdapat didalam basis data dataskripsi yang menyimpan data ID, dan waktu lama arduino mengirim data dan *broker* menerima data kedalam *stack*. Waktu yang diperoleh didapatkan dari rumus "Waktu akhir broker melakukan stack data - Waktu Awal Broker menerima data".

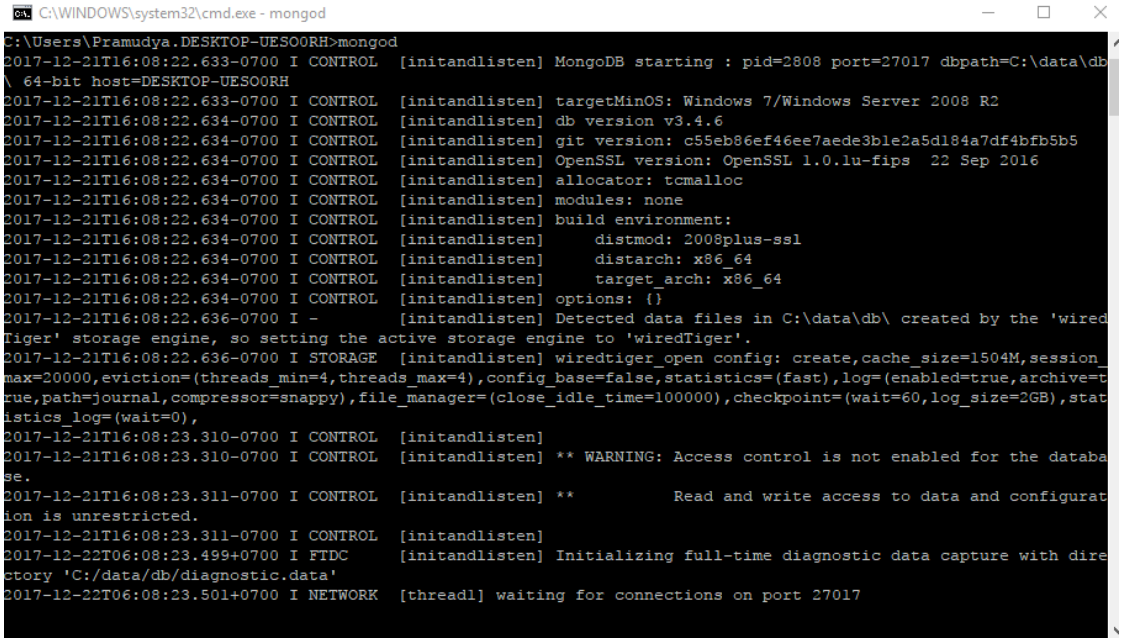
```
> db.datawaktu.find().pretty()
{
        "_id" : ObjectId("5a445d63f8e4980e5c469d59"),
        "waktu" : 0.10800004005432129
}
{
        "_id" : ObjectId("5a445d66f8e4980e5c469d8d"),
        "waktu" : 0.006999969482421875
}
{
        "_id" : ObjectId("5a445d69f8e4980e5c469dc1"),
        "waktu" : 0.006999969482421875
}
```

Gambar 5. 17. Collection "datawaktu"

"Datawaktu" adalah *collection* yang terdapat didalam basis data dataskripsi yang berisikan ID, dan data waktu pengiriman data *stack* yang berada didalam *broker* kedalam basis data dataskripsi. Data waktu didapatkan dari rumus "Akhir waktu pengiriman data – Awal waktu pengiriman data".

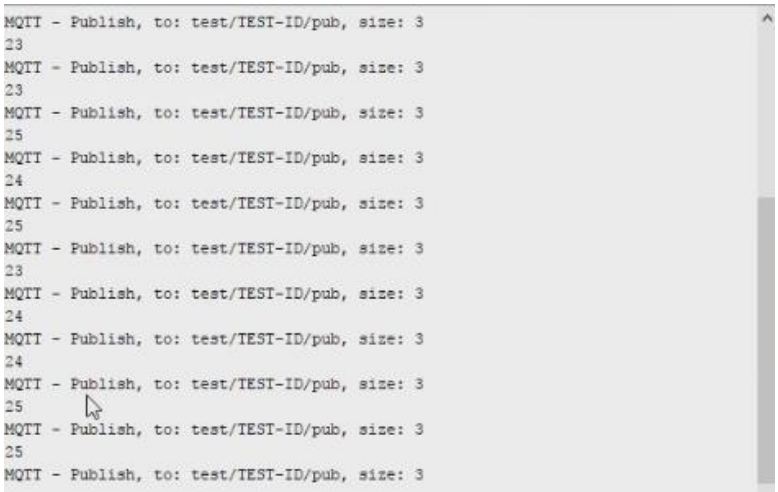
5.2.5 Implementasi Simulasi MQTT

Sesuai dengan perancangan simulasi MQTT, implementasi MQTT merupakan jalannya program *publish and subscribe*. Sebelum simulasi MQTT dijalankan, MongoDB harus pertama kali diaktifkan, dengan memasukkan perintah "mongod" kedalam *command prompt*.



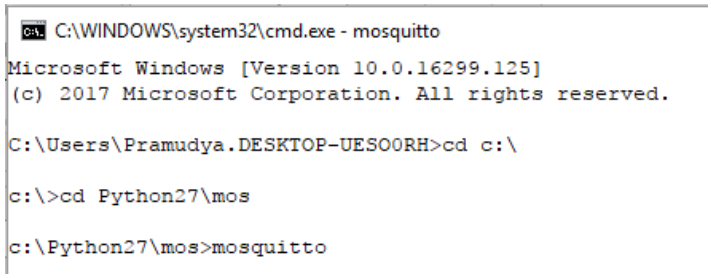
Gambar 5. 18. MongoDB aktif untuk menunggu koneksi masuk

Setelah MongoDB diaktifkan, program Arduino untuk *generate* data random dapat dijalankan, tetapi masih belum dapat mengirim data kedalam MongoDB karena program *mosquitto* dan *broker* belum dijalankan.



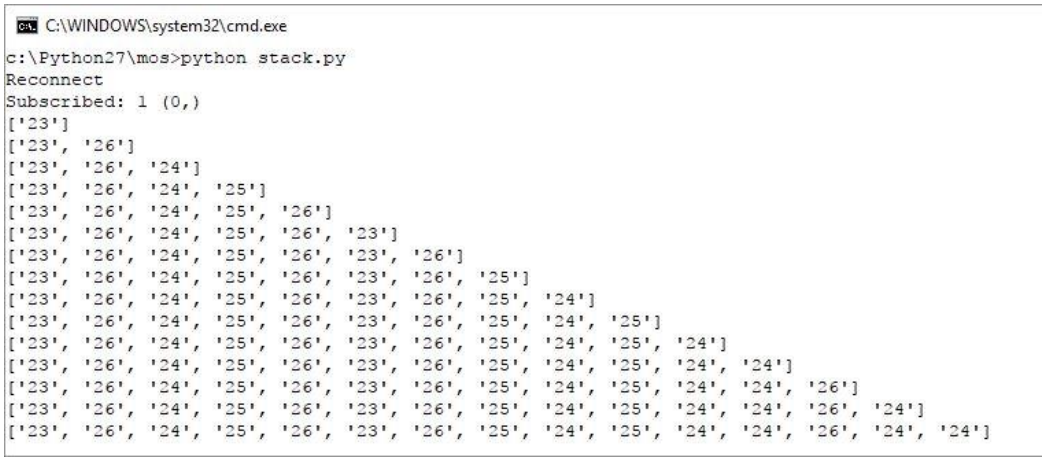
Gambar 5. 19. Program Generate Data NodeMCU

Setelah program arduino diaktifkan, maka program *mosquito* dan *broker* dapat dijalankan untuk mengirim data dari NodeMCU kedalam MongoDB.



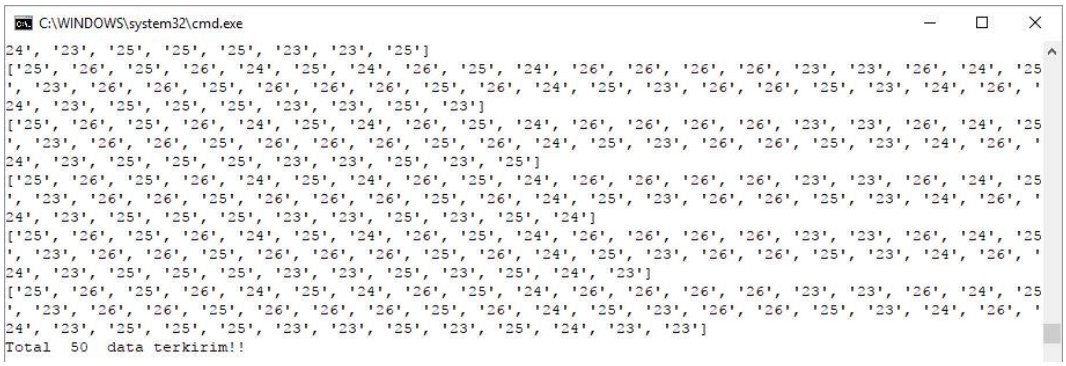
Gambar 5. 20. Program Mosquitto dijalankan

Program Mosquitto dijalankan lewat *command prompt* untuk menjalankan protokol MQTT yang akan digunakan sebagai dasar sistem. Setelah program mosquitto dijalankan, program *broker* dapat dijalankan untuk melakukan transmisi data.



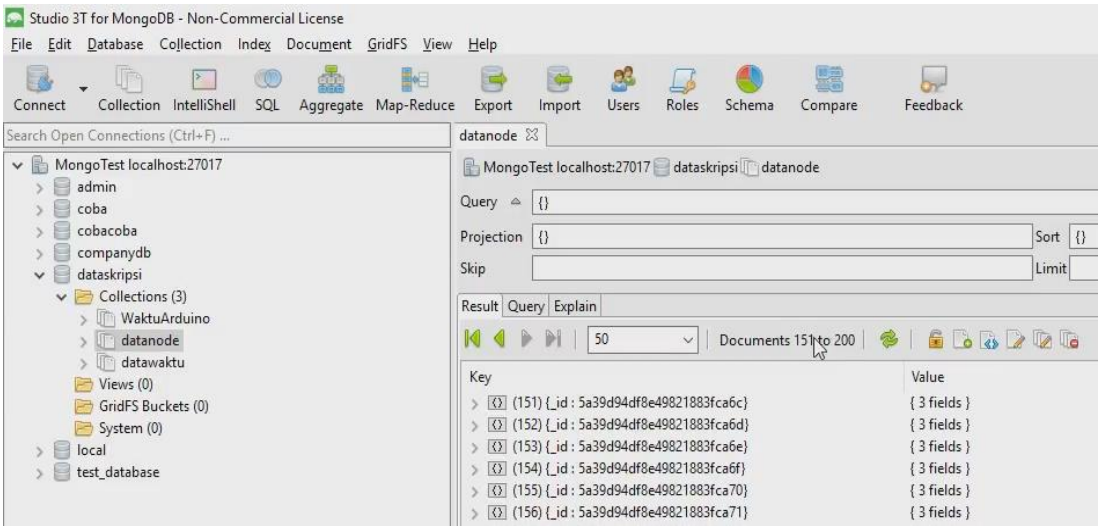
Gambar 5. 21. Program Broker dijalankan

Program *broker* berjalan didalam *command prompt* yang akan melakukan koneksi ke MQTT dan melakukan *stack* data sampai data ke N, bila Panjang data sudah sampai N, maka data akan dikirimkan kedalam basis data dataskripsi.



Gambar 5. 22. Contoh 50 program terkirim menggunakan Broker

Gambar 5.22 adalah contoh program broker yang dikonfigurasi untuk melakukan *stack* sebanyak 50-data dimana nantinya data tersebut akan dikirim kedalam basis data dataskripsi.



Gambar 5. 23. Data didalam basis data MongoDB

Setelah data terkirim dari *broker,* data dapat dilihat menggunakan *platform* Studio 3T, untuk melihat sistem basis data MongoDB.

Data terkirim kedalam sistem basis data MongoDB dan masuk kedalam basis data dataskripsi. Dapat dilihat bahwa dataskripsi mempunyai 3 *collections* yang isinya datanode, waktuarduino, dan datawaktu.