INTEGRASI *TIMING-SYNC PROTOCOL FOR SENSOR NETWORK* DENGAN *TIME DIVISION MULTIPLE ACCESS* PADA PROTOKOL MQTT-SN

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rania Akhmalia
NIM: 135150300111001

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

INTEGRASI *TIMING-SYNC PROTOCOL FOR SENSOR NETWORK* DENGAN *TIME DIVISION MULTIPLE ACCESS* PADA PROTOKOL MQTT-SN

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rania Akhmalia
NIM: 135150300111001

Skripsi ini telah diuji dan dinyatakan lulus pada
24 Januari 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika A., S.T, M.Eng	Mochammad Hannats Hanafi I., S.ST, M.T
NIP: 19820809 201212 1 004	NIK: 201405 881229 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 Januari 2017

Rania Akhmalia

NIM: 135150300111001

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Integrasi *Timing-Sync Protocol For Sensor network* dengan *Time Division Multiple Access* pada Protokol MQTT-SN" ini dapat terselesaikan.

Penulis menyadari bahwa penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh sebab itu, penulis menyampaikan rasa hormat dan terima kasih kepada:

1. Allah yang Maha Esa karena atas kehendak dan nikmat-Nya laporan skripsi ini telah selesai dengan baik.
2. Ibu Ida Wahyuni, Ayah Machmud dan keluarga atas nasehat, kasih sayang dan segala bentuk dukungan serta doanya.
3. Bunda Nila, Ayah Nushdi dan keluarga atas nasehat, kasih sayang dan segala bentuk dukungan serta doanya.
4. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng selaku dosen pembimbing pertama yang telah memberikan support dan bimbingannya kepada ananda untuk segera menyelesaikan skripsi ini.
5. Bapak Mochammad Hannats Hanafi I., S.ST, M.T selaku dosen pembimbing kedua yang selalu membantu ananda dalam menuntaskan laporan skripsi.
6. Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
7. Helfi Pangestu atas dukungannya, kesediaan meluangkan waktu dan tenaga juga doanya sampai pengerjaan skripsi selesai.
8. Zainul Bahar, Farid Azis, Wildan, Rint Zata, Novia Ulfa dkk atas kesediannya menjadi teman diskusi seama pengerjaan skripsi.
9. Teman-temanku tercinta di program studi teknik komputer angkatan 2013 yang selalu memberikan dukungan sehingga penulis dapat menyelesaikan skripsi ini.
10. Dan orang-orang yang selalu mensupprot dan mendokan ananda yang tidak datap ananda ucapakan satu persatu. Terimakasih atas semua doa dan support baik inmateril maupun non meteril.

Penulis mengucapkan banyak terima kasih dan penulis sadar akan adanya beberapa kekurangan pada laporan ini. Sehingga penulis berharap adanya penyempurnaan dari pihak-pihak terkait. Semoga laporan ini dapat memberikan manfaat dan referensi untuk melakukan penelitan dalam penyempurnaan sistem.

Malang, 24 Januari 2017

Penulis
raniaakhmalia@gmail.com

ABSTRAK

Internet of Things (IoT) merupakan konsep komunikasi *machine* to *machine* atau *machine* to *application* yang diturunkan dari teknologi *Wireless Sensor network* (WSN). Salah satu protokol komunikasi *machine to machine* yang pada implementasinya baik untuk menghemat sumber daya adalah *Message Queue Telemetry Transport* (MQTT). Pada ranah *sensor network*s Protokol MQTT dikenal dengan *Message Queue Telemetry Transport for Sensor network* (MQTT-SN). Terdapat 2 tipe arsitektur antara MQTT-SN *client* dengan MQTT-SN *gateway* yaitu trans*parent gateway* dan *aggregating gateway*. *Aggregating gateway* yaitu tipe arsitektur dengan single MQTT-SN *gateway*, dimana setiap MQTT-SN *client* terhubung pada satu MQTT-SN *gateway*. Permasalahan yang timbul pada arsitektur *aggregating gateway* adalah adanya kemungkinan tabrakan pengiriman data oleh klien. Agar terhindar dari permasalahan tersebut, maka diterapkan metode penjadwalan pengiriman dengan pembagian waktu yaitu Protokol *Time Division Multiple Access* (TDMA)pada masing-masing klien. Untuk menunjang metode penjadwalan, diperlukan penyetaraan waktu antar klien menggunakan metode *Timing-sync Protocol for Sensor network* (TPSN). Dari hasil implementasi didapatkan, bahwa klien dapat mengirimkan data sesuai dengan format pesan MQTT-SN. Pengiriman pesan juga berhasil dilakukan sesuai dengan slot waktu, dimana pada implementasinya slot waktu dibagi menjadi 10 detik. Pengiriman pesan dilakukan setelah masing-masing klien memiliki waktu yang setara dengan *gateway*. Implementasi sinkronisasi waktu berjalan cukup baik dengan rata-rata waktu tunggu sinkronisasi kurang dari 30 detik.

Kata kunci: MQTT-SN, *sensor networks*, TPSN, TDMA

ABSTRACT

Internet of Things (IoT) is the concept of communications machine to machine or machine to application derived technology Wireless Sensor network (WSN). One of machine to machine communication protocol which better to save resources is the Message Queue Telemetry Transport (MQTT). In the realm of sensor networks MQTT Protocol known as MQTT-SN (Message Queue Telemetry Transport for Sensor network). There are two types of architecture between MQTT-SN *client* and MQTT-SN *gateway* that is trans*parent gateway* and aggregating *gateway*. Aggregating *gateway* is the type of architecture with single *gateway*, where each MQTT-SN *client* connected on single MQTT-SN *gateway*. The problem of aggregating *gateway* architecture is collision data sending by the *client*. To avoid these problems then applied a *method* of scheduling data sending with Time Division Multiple Access (TDMA) Protocol on each *client*. To support the scheduling *method*, *client* needs a *synchronized* time between *client*s using Timing-sync Protocol for Sensor network (TPSN) Protocol. The results of implementation are *client* sent data to *gateway* using format message of MQTT-SN Protocol. The message was sent by *client* successfully match with a slot time where implementation of slot time divided as 10 seconds. Message delivery is made after each *client* has a time equivalent to the *gateway*. The Implementation of time *synchronization* is running well in average less than 30 seconds.

Keywords: MQTT-SN, sensor networks, TPSN, TDMA

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Internet of Things (IoT) merupakan konsep komunikasi *machine to machine* atau *machine to application* yang diturunkan dari teknologi *Wireless Sensor network* atau biasa disingkat dengan WSN. IoT mulai dikenalkan oleh Kevin Ashton pada tahun 1999. Salah satu protokol komunikasi *machine to machine* yang pada implementasinya baik untuk menghemat sumber daya adalah *Message Queue Telemetry Transport* (MQTT). MQTT adalah jenis protokol komunikasi data sentris yang memanfaatkan mekanisme *publish-subscribe* dengan protokol TCP/IP sebagai latar belakang komunikasinya. Pada ranah *sensor networks* Protokol MQTT dikenal dengan MQTT-SN (*Message Queue Telemetry Transport for Sensor network*). MQTT-SN didesain mirip dengan MQTT yang membedakan format pesan MQTT-SN lebih sederhana dibandingkan dengan MQTT. *Topic* pesan MQTT-SN menggunakan inisial sehingga panjang data lebih singkat (Stanford-Clark, 2013).

MQTT-SN memiliki dua komponen utama pada implementasinya, komponen pertama disebut MQTT-SN *client* sedangkan komponen yang kedua disebut dengan MQTT-SN *gateway*. MQTT-SN *client* terhubung dengan *broker* pada MQTT melalui MQTT-SN *gateway*. Terdapat 2 tipe arsitektur antara MQTT-SN *client* dengan MQTT-SN *gateway* yaitu trans*parent gateway* dan *aggregating gateway*. Arsitektur trans*parent gateway* yaitu terdapat multiple *gateway* yang terhubung dengan MQTT server artinya setiap satu MQTT-SN *client* terhubung dengan masing-masing satu MQTT-SN *gateway*. Sedangkan aristektur yang berbeda pada tipe *aggregating gateway* yaitu hanya terdapat satu MQTT-SN yang terhubung dengan MQTT server artinya terdapat setiap MQTT-SN *client* terhubung pada satu MQTT-SN *gateway* (Stanford-Clark, 2013).

Pemasalahan yang timbul pada arsitektur *aggregating gateway* ada pada proses pengumpulan data dari MQTT-SN *client* ke MQTT-SN *gateway*. Pengumpulan data merupakan fitur penting yang ada pada teknologi *sensor networks* dimana data sensor dikumpulkan dari *multiple* MQTT-SN *client* dikirim dengan media komunikasi nirkabel ke single MQTT-SN *gateway*. Permasalahnya adalah bagaimana data dapat dikirimkan menuju single MQTT-SN *gateway* tanpa kehilangan informasi. Kehilangan informasi dapat terjadi ketika komunikasi antara MQTT-SN *client* dengan MQTT-SN *gateway* mengalami tabrakan dengan MQTT-SN *client* yang lain sehingga data tidak dapat diterima dan informasi tidak dapat tersampaikan. Oleh sebab itu, dibutuhkan sebuah metode penjadwalan waktu pengiriman data antar MQTT-SN *client* menuju MQTT-SN *gateway* dengan tujuan menghindari terjadinya tabrakan data.

Untuk menunjang penjadwalan waktu pengiriman data antar MQTT-SN *client* menuju MQTT-SN *gateway* perlu dilakukan sinkronisasi waktu antar MQTT-SN *client*. Sinkronisasi waktu atau dikenal dengan *time-syncronization* dilakukan sebab MQTT-SN *client* tidak dapat aktif dalam waktu yang bersamaan. Dengan setaranya waktu yang dimiliki setiap MQTT-SN *client* akan dapat dilakukan

penjadwalan waktu pengiriman sehingga data dari multiple MQTT-SN *client* dapat dikirimkan dengan tepat menuju single MQTT-SN *gateway*.

Salah satu metode sinkronisasi waktu yang dapat digunakan adalah *Timing-sync Protocol for Sensor networks* (TPSN). Pada TPSN sinkronisasi waktu dilakukan pada sisi *sender-receiver* dimana dalam kasus ini yang bertindak sebagai sender adalah MQTT-SN *gateway* sedangkan receiver adalah MQTT-SN *client*. Sinkronisasi waktu dengan menggunakan metode TPSN menerapkan topologi hirarki, dimana setiap node akan dibagi pada level-level tertentu. Node dengan level *n* dapat berkomunikasi dengan node pada level *n-1.* Metode sinkronisasi waktu TPSN dianggap lebih *scalable* dibandingkan dengan metode sinkronisasi waktu lainnya seperti metode *Reference Broadcast Synchronization* (RBS) karena menerapkan topologi hirarki (Sundararaman, 2005). Protokol sinkronisasi waktu RBS adalah protokol yang pada implementasinya menggunakan node *beacon* atau node tambahan yang berguna melakukan *broadcast* paket sinkronisasi menuju seluruh node yang ada pada jaringan.

Penelitian ini bertujuan menerapkan metode sinkronisasi waktu jenis *Timing-sync Protocol for Sensor networks* (TPSN) dan penjadwalan waktu pengiriman menggunakan metode *Time Division Multiple Access* (TDMA) pada MQTT-SN *client* sebagai solusi agar terhindar dari resiko terjadinya tabrakan saat melakukan pengiriman data menuju MQTT-SN *gateway*. Diharapkan penelitian ini mampu memberikan alternatif solusi pengiriman dan penerimaan informasi pada jaringan sensor.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dikemukakan, rumusan masalah pada penelitian ini antara lain:

1. Bagaimana implementasi sinkronisasi waktu menggunakan protokol TPSN untuk menyetarakan waktu lokal antara MQTT-SN *client* dengan MQTT-SN *gateway*?
2. Bagaimana implementasi penjadwalan pengiriman pesan dengan format pesan MQTT-SN menggunakan protokol TDMA?
3. Bagaimana MQTT-SN *gateway* dapat meneruskan pesan menuju jaringan lokal?
4. Bagaimana jaringan lokal dapat menerima dan menampilkan pesan yang diterima melalui web?

1.3 Tujuan

Tujuan dilakukannya penelitian ini adalah:

1. Agar mampu mengimplementasikan metode sinkronisasi waktu jenis *Timing-sync Protocol for Sensor network*s (TPSN) pada *multiple* MQTT-SN *client*
2. MQTT-SN *client* mampu menggunakan metode penjadwalan *Time Division Multiple Access* (TDMA) dalam rangka melakukan pengiriman data dengan format pesan MQTT-SN.

3. MQTT-SN *gateway* mampu meneruskan pesan dari MQTT-SN *client* menuju *publisher* pada lingkungan simulasi MQTT jaringan lokal.
4. MQTT *subscriber* pada jaringan lokal simulasi MQTT dapat menerima data dan menampilkannya melalui grafik pada web dengan memanfaatkan port websocket aplikasi broker.

1.4 Manfaat

Sebagai solusi untuk menghindari *collision* saat MQTT-SN *client* melakukan pengiriman data dengan mengimplementasi metode sinkronisasi waktu Timing-sync Protocol for *Sensor network*s (TPSN) dan penjadwalan waktu pengiriman data menggunakan metode *Time Division Multiple Access* (TDMA) pada arsitektur *aggregating gateway* protokol MQTT-SN.

1.5 Batasan masalah

Agar pembahasan dalam penelitian ini dapat dilakukan secara terarah dan mendapatkan hasil sesuai dengan yang diharapkan, maka perlu diterapkan batasan permasalahan yaitu implementasi sistem pada penelitian ini berupa *prototype* dengan pesan yang dikirim oleh klien merupakan nilai suhu dan kelembaban.

1.6 Sistematika pembahasan

Uraian singkat mengenai metodologi penelitian pada masing-masing bab adalah sebagai berikut:

BAB I Pendahuluan
Menjelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari "Integrasi *Timing-Sync Protocol for Sensor Network* dengan *Time Division Mutiple Access* pada Protokol MQTT-SN"

BAB II Landasan Kepustakaan
Pada bab ini akan menjelaskan tentang landasan teori yang terkait dengan penelitian. Pada bab ini juga dijelaskan tentang penelitian serupa yang pernah dilakukan.

BAB III Metode Penelitian
Membahas tentang langkah kerja yang dilakukan dalam penulisan diantaranya studi literatur, analisis kebutuhan system, desain sistem, dan impementasi dari protokol MQTT-SN, TPSN dan TDMA.

BAB IV Rekayasa Kebutuhan
Bab ini menjelaskan secara rinci yang terkait meliputi deskripsi umum dari sistem, rekayasa kebutuhan antar-muka sistem, kebutuhan perangkat keras dan lunak, kebutuhan fungsional, kebutuhan komunikasi, kebutuhan performansi, batasan desain sistem dan alur kerja sistem.

BAB V Perancangan dan Implementasi
Bab ini menjelaskan perancangan sistem penelitian serta

implementasi sistem penelitian berupa implementasi perangkat keras, implementasi perangkat lunak, implementasi perancangan database dan implementasi aplikasi web sebagai *output* dari sistem.

BAB VI Pengujian dan Analisa

Bab ini membahas scenario pengujian, hasil pengujian beserta analisis hasil pengujian.

BAB VII Penutup

Bab ini membahas tentang kesimpulan dan saran yang diperoleh dari rumusan masalah penelitian dengan melakukan analisis dan pengujian.

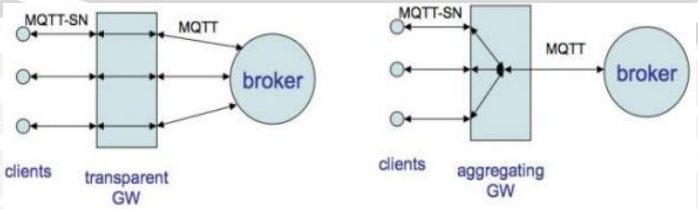
BAB 2 LANDASAN KEPUSTAKAAN

Pada penelitian ini, penulis mengkaji dari buku pedoman tentang konsep protokol MQTT-SN yang berjudul "MQTT *for Sensor networks* (MQTT-SN) *Protocol Spesification*" ditulis oleh Andy Stanford-Clark. Adapun jurnal penelitian sebelumnya yang penulis jadikan referensi berisi percobaan implementasi *wireless sensor network* protokol yang berjudul "Implementasi *Wireless sensor network Publisher* Menggunakan Protokol *Message Queue Telemetry Transport – Sensor network* (MQTT-SN)" ditulis oleh Muhammad Azka Azzahidin. Jurnal yang dikaji berkaitan dengan sinkronisasi waktu berjudul "Implementasi *Time Synchronization* Pada WSN Untuk Metode TDMA Menggunakan Algoritma TPSN" ditulis oleh Ardy Novian Erwanda dan jurnal lain yang berjudul "Timing-sync Protocol for *Sensor network*s." Ditulis oleh Saurabh Ganeriwal, dkk. Sedangkan untuk beberapa referensi dasar teori sebagai pengetahuan tentang komponen dan teknologi yang digunakan meliputi spesifikasi protokol MQTT, koneksi MQTT dengan *database*, Arduino Nano, Modul *wireless* NRF24L01 dan sensor DHT11.

2.1 Tinjauan Pustaka

Teknologi WSN ini pada masa mendatang merupakan bagian dari perangkat *Internet of Things* (IoT) (Khalil, 2014). Perangkat WSN tersebut harus mampu berkomunkasi menggunakan protokol dan arsitektur tertentu yang ada pada konsep IoT. Oleh karena itu perlu diterapkan sebuah protokol IoT pada perangkat WSN. Salah satu protokol yang bisa diimplementasikan yaitu protokol MQTT-SN. Yaitu sebuah protokol MQTT yang secara spesifik diiplementasikan pada jaringan sensor (Govindan, 2015).

Pada buku pedoman konsep dari protokol MQTT-SN yang berjudul "MQTT *For Sensor networks* (MQTT-SN) *Protocol Spesification*" ditulis oleh Andy Stanford-Clark menerangkan tentang keseluruhan konsep dari protokol MQTT-SN yang meliputi perbedaan MQTT-SN dengan MQTT konvensional, terletak pada format pengiriman pesan, arsitektur, dan deskripsi berbagai macam fungsi dari MQTT-SN. Dari sumber yang sama dijelaskan bahwa *gateway* merupakan *forwarder* dari node MQTT-SN *client* menuju *broker* MQTT atau server. Dalam proses *forward* dari node MQTT-SN *client* menuju *gateway* ini menggunakan protokol MQTT-SN. Terdapat 2 tipe dari *gateway* yaitu agregating *gateway* dan trans*parent gateway*. Kedua tipe *gateway* ini dapat dijelaskan melalui Gambar 2.1 dibawah ini.



Gambar 2.1 Tipe *Gateway*
Sumber : (Stanford-Clark, 2013)

Dari Gambar 2.1 Tipe *Gateway* dapat disimpulkan perbedaan kedua tipe *gateway* tersebut, *gateway* yang pertama yaitu trans*parent gateway* merupakan *gateway* yang terhubung secara *wireless* pada setiap node MQTT-SN *client*. Sedangkan *aggregating gateway* yaitu *gateway* yang terhubung secara *wireless* pada semua node MQTT-SN *client*. Pada tipe *aggregating* hanya ada satu koneksi untuk semua node MQTT-SN *client* menuju *broker* melalui *gateway* lain dengan transparansi dimana setiap node WSN *Publisher* memiliki koneksi *one-to-one* menuju *broker* melalui *gateway*.

Penelitian oleh Muhammad Azka Azzahidin yang berjudul "Implementasi *Wireless sensor network Publisher* Menggunakan Protokol *Message Queue Telemetry Transport – Sensor network* (MQTT-SN)" mengaplikasikasikan *single* node MQTT-SN *client* dan MQTT-SN *gateway* pada sistem monitoring suhu dan kelembaban. Lalu pada jaringan lokal MQTT, penelitian ini mengimplementasikan aplikasi publisher dan broker (Azzahidin, 2016).

Penelitian yang dilakukan oleh Ardy Novian Erwanda yang berjudul "Implementasi *Time Synchronization* Pada WSN Untuk Metode TDMA Menggunakan Algoritma TPSN" menyatakan bahwa TPSN dapat berjalan sebagaimana konsep topologi hirarki begitu juga dengan penjadwalan TDMA dapat berjalan sesuai dengan slot waktu yang telah diset sebelumnya dengan tingkat akurasi sinkronisasi waktu pada level 1 kurang dari 2 detik sedangkan pada level 2 kurang dari 9 detik (Erwanda, 2016).

Berdasarkan penelitian-penelitian diatas, dapat diketahui bahwa Protokol sinkronisasi waktu TPSN dan metode penjadwalan pengiriman pesan TDMA telah berhasil diimplementasikan dan menghasilkan nilai akurasi kurang dari 26 detik. Protokol MQTT-SN juga telah diimplementasikan untuk sistem *monitoring* suhu dan kelembaban menggunakan *single* MQTT-SN *client* dan *single* MQTT-SN *gateway*. Fokus pada penelitian ini adalah, melakukan integrasi protokol sinkronisasi waktu TPSN dan protokol penjadwalan pengiriman pesan TDMA dengan protokol *publish-subscribe* pada arsitektur *aggregating gateway* protokol MQTT-SN.

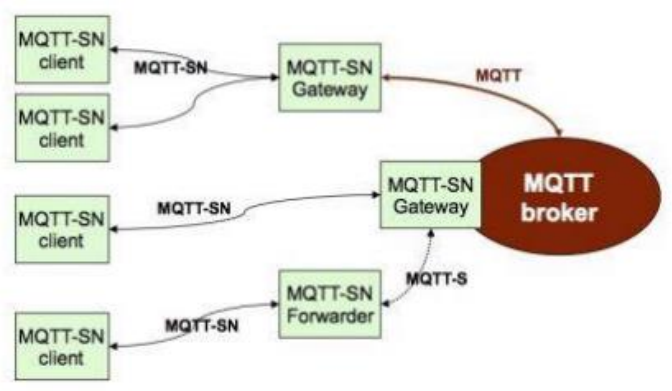
2.2 Dasar Teori

Wireless sensor network (WSN) merupakan teknologi yang sedang dikembangkan secara pesat beberapa tahun ini. WSN merupakan salah satu generator untuk membangkitkan konsep *Internet of Things* (IoT). *Internet of Things* yang mengusung konsep komunikasi *machine-to-machine* memiliki beberapa protokol, salah satunya *Message Queue Telemetry Transport* (MQTT). Pada subbab dasar teori akan dijelaskan referensi dasar dan teori-teori pendukung mengenai protokol MQTT berbasis *sensor network,* metode sinkronisasi waktu TPSN dan protokol penjadwalan TDMA serta komponen penunjang implementasi penelitian ini seperti mikrokontroler, komunikasi *wireless* menggunakan media NRF, sensor temperatur dan kelembaban.

2.2.1 MQTT-SN

MQTT atau *Message Queue Telemetry Transport* merupakan protokol data sentris yang menerapkan metode *publish-subscribe* pada pengiriman datanya. Protokol MQTT memiliki kemampuan pemrosesan data yang rendah sehingga ideal untuk diterapkan pada jaringan *sensor network* dengan *bandwith* dan *resource* terbatas. MQTT menerapkan tipe komunikasi TCP/IP dimana TCP/IP bersifat *connection-oriented* dengan data yang *reliable*.

Implementasi protokol MQTT pada lingkungan sensor network dikembangkan menjadi MQTT-SN (*Message Queue Telemetry Transport – Sensor* Network). Protokol ini diterapkan sebagai penghubung antara jaringan sensor dengan *gateway* dimana *gateway* sendiri merupakan penghubung antara jaringan sensor dengan jaringan lokal. Jaringan lokal menerapkan komunikasi MQTT yang berbasis protokol TCP, berbeda dengan MQTT-SN yang menerapkan protokol UDP. Untuk lebih jelasnya, Gambar 2.2 merupakan gambar arsitektur MQTT-SN.



Gambar 2.2 Arsitektur MQTT-SN
Sumber : (Stanford-Clark, 2013)

Metode pengiriman pesan dengan protokol MQTT-SN terbagi menjadi 3 *QoS* (*Quality of Service*). *QoS* 0 dengan mekanisme pengiriman pesan tanpa *acknowledge*. Mekanisme pengiriman pesan *QoS* 1 menggunakan *acknowledge* sehingga data lebih terjamin untuk sampai pada *receiver*. Sedangkan *QoS* 2 menawarkan tingkat keamanan yang paling tinggi dengan mekanisme pengiriman *double acknowledge*. *Acknowledge* yang pertama digunakan untuk memastikan data yang diterima sesuai dengan *topic* atau tidak. *Acknowledge* yang kedua berguna sebagai pesan konfirmasi bahwa data berhasil diterima oleh *receiver*.

2.2.2 Mikrokontroler

Mikrokontroler adalah komputer mikro yang kompleks dan dirancang untuk mengatur operasi atau menjadi kontrol pada sistem *embedded* ataupun dunia robotika. Adapun jenis mikrontroler yang akan digunakan pada penelitian ini adalah Arduino Nano v3.0. Arduino merupakan salah satu mikrokontroler yang sering dijumpai dipasaran dan mudah untuk digunakan. Arduino Nano versi 3.0 merupakan bentuk pengembangan arduino nano. Arduino nano memiliki

body yang relatif kecil jenis *breadboard-friendly board* dengan mikroprosesor ATmega328 pada arduino nano versi 3.

Gambar 2.3 Arduino Nano v3.0
Sumber: (https://www.arduino.cc/en/Main/ArduinoBoardNano)

Spesifikasi yang dimiliki Arduino Nano v3.0 dijelaskan pada Tabel 2.1.

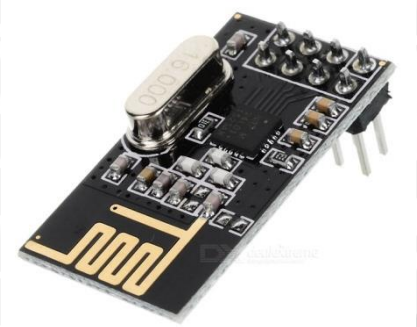
Tabel 2.1 Spesifikasi Arduino Nano v3.0

Microcontroller	Atmel ATmega328
Operating Voltage	5 V
Input Voltage	7-12 V
Digital I/O Pins	14 (of which 6 provide PWM *output*)
Analog *Input* Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

Sumber: (https://www.arduino.cc/en/Main/ArduinoBoardNano)

2.2.3 Modul *Wireless* nRF24L01

Wireless Module nRF24L01 adalah sebuah modul komunikasi jarak jauh yang memanfaatkan pita gelombang RF 2.4GHz ISM (*Industrial, Scientific and Medical*). Modul ini menggunakan antarmuka SPI untuk berkomunikasi. Tegangan kerja dari modul ini adalah 5V DC. NRF24L01 memiliki *baseband logic Enhfanced ShockBurst™ hardware protocol accelerator* yang support *"high-speed* SPI *interface for the application controller"*. nRF24L01 memiliki *true ULP solution*, yang memungkinkan daya tahan baterai berbulan-bulan hingga bertahun-tahun. Modul ini dapat digunakan untuk pembuatan *pheriperal* PC, piranti permainan, piranti fitnes dan olahraga, mainan anak-anak dan alat lainnya. Modul ini memiliki 8 buah pin seperti pada Gambar 2.4 dibawah ini.



Gambar 2.4 *Wireless* Module NRF24L01
Sumber: (www.dx.com, 2016)

Spesifikasi lain yang dimiliki oleh modul nRF24L01 dijelaskan pada Tabel 2.2.

Tabel 2.2 Spesifikasi Modul nRF24L01

Power supply	1.9V~3.6V
I/O port working voltage	0~3.3, 5 V
I/O port working current	13.5mA at 2Mbps
Data rate	256kbps / 1Mbps / 2Mbps
Receiving sensitivity	-85dBm at 1Mbps
Transmission range	70~100 meter at 256kbps
*Temperature*s	Operating:-40°C ~ 85°C / Storage:-40°C ~ 125°C

Sumber: (Ball, 2008)

2.2.4 Sensor Temperatur dan Kelembaban

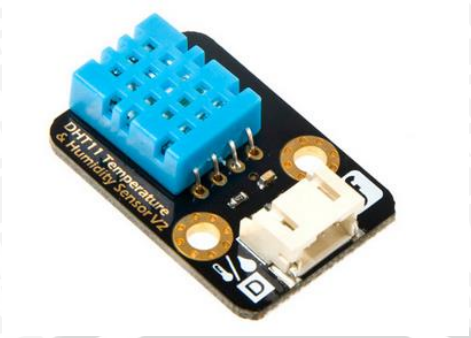
Sensor suhu dan kelembaban merupakan perangkat elektronika yang berguna untuk mengukur besaran suhu pada satuan celcius (^{o}C) dan kelembaban dalam satuan persen (%). Sensor suhu dan kelembaban biasanya digunakan untuk keperluan sistem monitoring ataupun sistem antisipasi terjadinya bencana didalam atau diluar ruangan. Salah satu jenis sensor suhu dan kelembaban adalah DHT11.

Sensor DHT11 merupakan sensor digital dengan tingkat stabilitas yang sangat baik serta fitur kalibrasi yang sangat akurat. Sensor DHT11 juga menyediakan *library* khusus yang bernama DHT *Library*, *library* tersebut berguna untuk memudahkan pengguna memprogram di mikrokontroler. Tabel 2.3 adalah spesifikasi sensor DHT11 dan Gambar 2.5 merupakan gambar sensor DHT11.

Tabel 2.3 Spesifikasi Sensor DHT11

Supply Voltage	3- 5 Volt
Temperature range	0-50 °C Error ± 2 °C
Humidity	20-90% RH Error ± 5% RH

Sumber: (D-Robotics UK, 2010)



Gambar 2.5 Sensor DHT11
Sumber: (D-Robotics UK, 2010)

2.2.5 *Broker*

Broker pada metode komunikasi *publish-subscribe* bertindak sebagai server yang bertugas mendistribusikan pesan dari *broker* menuju *subscriber*. Dibutuhkan aplikasi yang dapat menerapkan fungsi *broker*, salah satu aplikasi MQTT *broker open source* yang dapat digunakan adalah *Apache ActiveMQ*. *Apache ActiveMQ broker* adalah aplikasi *broker open source* dan berlisensi Apache v2.0 yang bersifat *scalable* dan *clusterable* serta support dengan banyak bahasa pemrograman juga berbagai macam protokol (ActiveMQ, 2016). Untuk dapat menerapkan aplikasi *broker* tersebut pada jaringan diperlukan aplikasi yang berperan sebagai *publisher* dan *subscriber*. Aplikasi *Publisher* berfungsi untuk melakukan *publish* pesan menuju *broker* sesuai dengan *topic*nya. Sedangkan aplikasi *subscriber* berfungsi untuk melakukan *subscribe* ke *broker* untuk mendapatkan pesan sesuai dengan *topic* yang ada pada *broker*. Dalam proses membangun aplikasi *publisher* dan *subscriber* tersebut diperlukan suatu *library* tertentu agar aplikasi terebut dapat berperan sebagai *klien* pada *broker*. Salah satu contoh dari *library* tersebut adalah paho *library* yang digunakan dalam bahasa pemrograman python (Python, 2015).

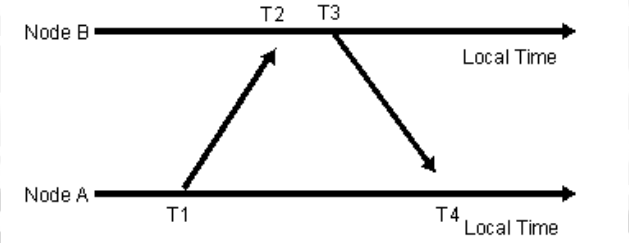
2.2.6 Timing-Sync Protocol for *Sensor network*

Timing-Sync Protocol for Sensor networks (TPSN) adalah sebuah algoritma sinkronisasi waktu pada WSN, yang menggunakan pemodelan topologi *tree*. Dimana terdapat sebuah node sebagai *root* yang berada di pangkal topologi hirarki, sinkronisasi waktu dimulai dari *root* tersebut. Dengan menggunakan aliran data yang disebut *timestamp* yang dikirim dari *root* ke hierarki dibawahnya, waktu disamakan melalui *timestamp* tersebut. TPSN memiliki keunggulan dibandingkan beberapa algoritma lain seperti skala yang lebih besar daripada algoritma RBS (Elson).

Terdapat dua fase pada algoritma TPSN, fase pertama adalah fase *discovery*. Fase ini digunakan saat pertama kali seluruh node membentuk topologi hirarki. Fase *discovery* diawali oleh node root yang berada pada level 0 berlanjut ke node level 1, level 2, dan seterusnya. Seluruh node akan mengingat posisinya sebagai penghuni dari level masing-masing, serta mengenali alamat *parent*-nya. Sistem ini dilakukan dengan pertama kali aliran data paket *discovery*

dari node *root* diteruskan sampai ke level paling ujung, sehingga semua node masuk ke daftar hirarki.

Fase kedua adalah fase sinkronisasi, pada fase ini, semua node akan memulai sinkronisasi waktu. Semua diawali dari node dengan level paling kecil terlebih dahulu.



Gambar 2.6 Mekanisme TPSN
Sumber: (Sundararaman, 2005)

Berdasarkan Gambar 2.6 mekanisme sinkronisasi waktu menggunakan protokol TPSN, T1 dan TP adalah waktu lokal yang dihasilkan oleh node A sedangkan T2 dan T3 merupakan waktu lokal yang dihasilkan oleh node B. Diasumsikan level node A lebih besar dari level node B. Tahapan sinkronisasi waktu menggunakan protokol TPSN berdasarkan Gambar 2.6 dijelaskan sebagai berikut:

1. Node A mengirimkan paket sinkronisasi menuju node B. Paket sinkronisasi berisi level node A sendiri dan nilai T1. T1 adalah waktu pengiriman paket sinkronisasi.
2. Node B menerima paket sinkronisasi dari node A pada waktu T2. Dimana T2 = T1 + δ + d; δ merepresentasikan nilai *clock offset* antara node A dan node B; sedangkan d merepresentasikan delay propagasi anatara node A dan B.
3. Node B mengirimkan paket *acknowledgement* menuju node A. paket ack berisi level dari node B dan nilai dari T1, T2 dan T3.
4. Node A menghitung clock offset dan delay yang diterima dari node B. Node A melakukan sinkronisasi berdasarkan berdasarkan perhitungan dibawah ini.

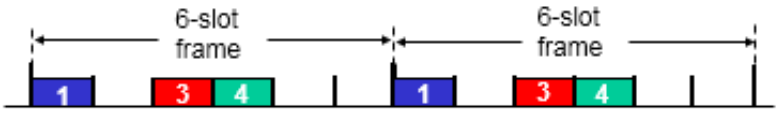
$$\delta = \frac{(T2 - T1) - (T4 - T3)}{2} \quad (2.1)$$

$$d = \frac{(T2 - T1) + (T4 - T3)}{2} \quad (2.2)$$

2.2.7 Time Division Multiple Access

Protokol *Multi Access* memiliki tiga mekanisme utama, yaitu *channel partitioning, random access,* dan *taking turns. Channel Partitioning* adalah mekanisme Protokol *Multi Access* dengan algoritma terdistribusi yang mengatur bagaimana *channel* atau *link* yang digunakan secara bersama-sama untuk dibagi menjadi beberapa "*pieces*" yang selanjutnya akan dialokasikan ke tiap-tiap node. Terdapat berbagai metode pada protokol *channel partitioning*, diantaranya, *Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA),* dan *Code Division Multiple Access*

Taking Turn merupakan mekanisme *multiple access* yang mengatur pengiriman tiap node berdasarkan *polling* atau *token* yang dikirimkan oleh sebuah *master* node sebagai "undangan", kemudian node penerima baru akan mengirimkan datanya sehingga tidak terjadi pengiriman bersama-sama. Mekanisme ini memiliki beberapa kelemahan seperti *polling/tokens overhead, latency* pada pengiriman, dan *single point of failure* (Kurose & Ross, 2012).

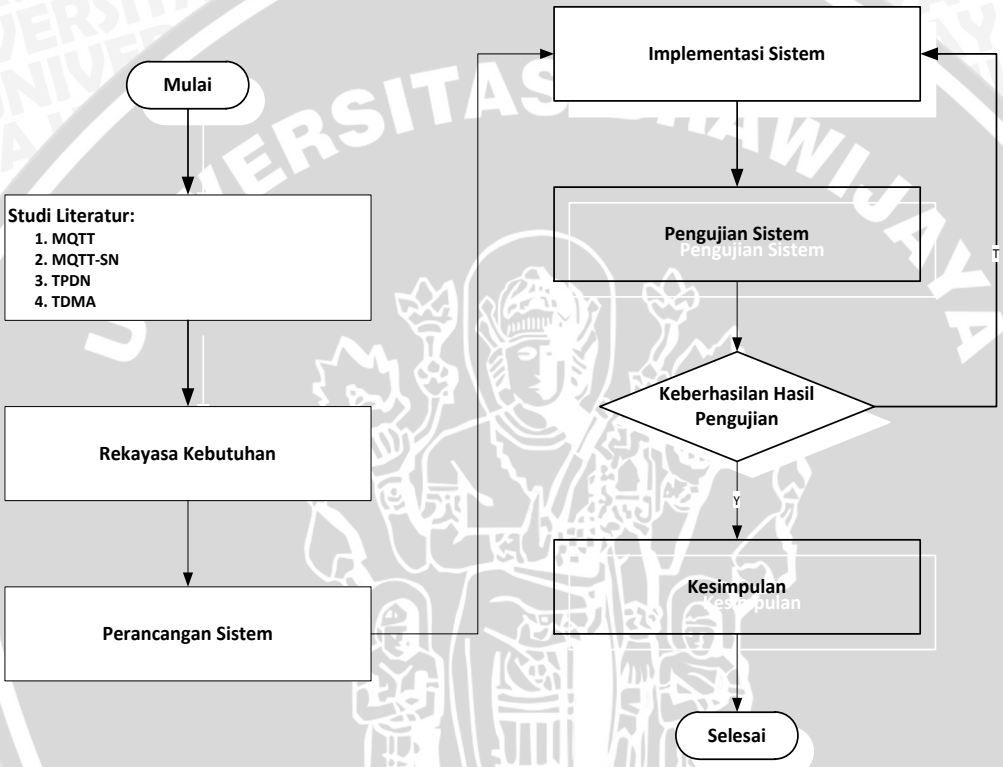


Gambar 2.7 Ilustrasi TDMA
Sumber: (Kurose & Ross, 2012)

Time Division Multiple Access atau TDMA adalah teknologi untuk peralatan pengiriman data digital menggunakan teknik multipleksi pembagian waktu. TDMA bekerja dengan membagi frekuensi radio ke dalam slot waktu yang dapat mendukung pemakaian aliran kanal data secara bersama-sama. Transmisi dalam bentuk urutan *frame,* dimana tiap frame dibagi menjadi beberapa slot waktu, dan tiap slot waktu bersifat *dedicated* untuk sebuah transmitter tertentu. (Stallings, 2005). Setiap pengirim akan memiliki jadwal pengiriman yang berbeda dengan jadwal pengirim yang lain, dan semua pengirim memiliki waktu yang tersinkronisasi. Seperti ditunjukkan pada Gambar 2.7, terdapat 6 *time slot* pada sebuah *frame,* dimana beberapa slot digunakan oleh pengirim 1, 3, dan 4; sedangkan slot 2,5, dan 6 sedang dalam kondisi *idle.* Pengiriman kembali dilakukan pada *frame* yang lainnya.

BAB 3 METODOLOGI

Penelitian diawali dengan melakukan studi literatur terkait kajian pustaka dan dasar teori. Penelitian ini bersifat implementatif, berupa integrasi metode sinkronisasi waktu jenis TPSN dengan penjadwalan *Time Division Multiple Access* (TDMA) pada protokol *Message Queue Telemetry Transport for Sensor network* (MQTT-SN). Penentuan alur metode penelitian sebagai langkah yang ditempuh untuk menyelesaikan penelitian secara sistematis. Alur metode penelitian yang dilakukan untuk pembuatan sistem ini dapat dilihat dari diagram alir pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

3.1 Studi Literatur

Studi literatur merupakan tahap mencari dan melakukan penyusanan teori dasar dan referensi yang mendukung sistem yang dibuat sebagai penunjang penelitian. Studi literatur pada penelitian ini dilakukan dengan pemahaman terhadap tinjauan pustaka dan dasar teori yang terkait dengan:

1. *Wireless Sensor network* (WSN) dan *Internet of Things* (IoT)
   Studi literatur mengenai *Wireless sensor network* dan *Internet of Things* dilakukan dengan mencari jurnal juga berita perkembangan teknologi *Wireless Sensor network* (WSN) dan *Internet of Things* (IoT)
2. Spesifikasi MQTT-SN
   Studi literatur mengenai protokol MQTT-SN dilakukan dengan mengkaji buku pedoman MQTT-SN dan memahami arsitektur, komunikasi, format pengiriman data pada protokol tersebut.

3. Spesifikasi MQTT
   Studi literatur spesifikasi Protokol MQTT dilakukan dengan mengkaji buku pedoman MQTT-SN dan memahami arsitektur, komunikasi, format pengiriman data pada protokol tersebut.
4. *Timing-sync Protocol for Sensor network* (TPSN)
   Studi literatur tentang prinsip kerja sinkronisasi menggunakan metode tradisional sender receiver berbasis topologi hirarki dilakukan dengan mengkaji jurnal yang berkaitan.
5. *Time Division Multiple Access* (TDMA)
   Studi literatur tentang prinsip kerja pengiriman data dengan metode pembagian slot-slot waktu dilakukan dengan mengkaji jurnal yang berkaitan.
6. *Realtime Web Application*
   Studi literatur mengenai komunikasi Protokol MQTT memanfaatkan port *websocket* agar dapat menampilkan data secara *realtime* dilakukan dengan mengkaji artikel dan laman-laman tutorial

3.2 Rekayasa Kebutuhan

Analisis kebutuhan ditujukan untuk melakukan analisis pada beberapa kebutuhan yang diperlukan sistem pada penelitian ini. Analisis kebutuhan pada penelitian ini dijelaskan sebagai berikut.

3.2.1 Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras (*hardware*) yaitu menganalisa kebutuhan perangkat keras apa saja yang digunakan untuk mewujudkan sistem pada penelitian ini. Adapun perangkat keras yang digunakan antara lain:

1. Laptop
2. Arduino Nano v3.0
3. Modul *Wireless* nRF24L01
4. DHT11 *Temperature* and *Humidity* Sensor

3.2.2 Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak yaitu menganalisa kebutuhan perangkat lunak apa saja yang digunakan untuk mewujudkan sistem pada penelitian ini. Adapun perangkat keras yang digunakan antara lain:

1. Arduino IDE 1.6
2. Apache ActiveMQ *Broker*
3. Paho Eclipse *Library*

3.2.3 Kebutuhan Fungsional

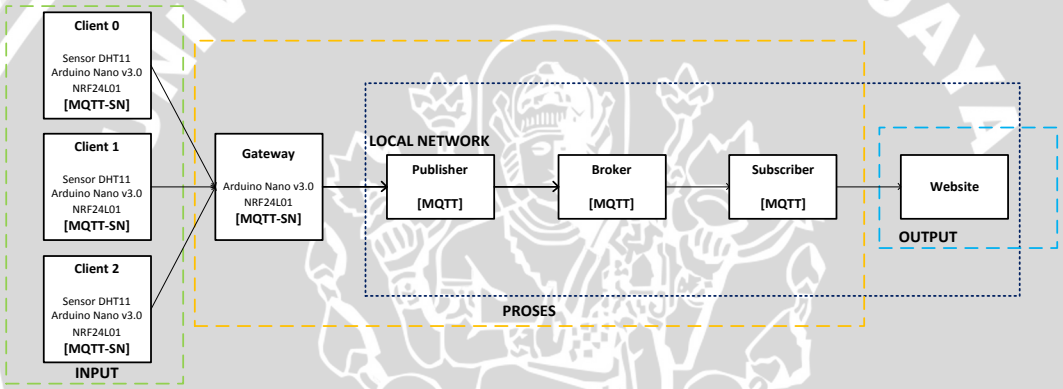
Analisis kebutuhan fungsional yaitu menganalisa kebutuhan fungsi apa saja yang akan dicapai sistem pada penelitian ini. Adapun kebutuhan fungsional yang akan dicapai antara lain:

1. Masing-masing MQTT-SN *client* memiliki waktu setara antara satu dengan yang lain.

2. Masing-masing MQTT-SN *client* dapat mengirimkan data dengan *topic* yang sama menggunakan protokol TDMA.
3. Masing-masing MQTT-SN *client* dapat mengirimkan data dengan *topic* yang berbeda menggunakan protokol TDMA.
4. MQTT-SN *gateway* dapat melakukan pengiriman data melalui komunikasi serial menuju *broker*.
5. MQTT *subscriber* dapat menerima data dan menampilkannya melalui aplikasi web menggunakan port websocket pada *broker*.

3.3 Perancangan Sistem

Perancangan sistem adalah tahap yang dilakukan setelah analisis kebutuhan sistem. Perancangan dilakukan apabila seluruh kebutuhan sistem telah terpenuhi. Tujuan perancangan sistem secara umum, agar implementasi sistem berjalan sistematis dan terstruktur. Perancangan sistem penelitian digambarkan pada blok diagram Gambar 3.2.



Gambar 3.2 Blok Diagram Sistem

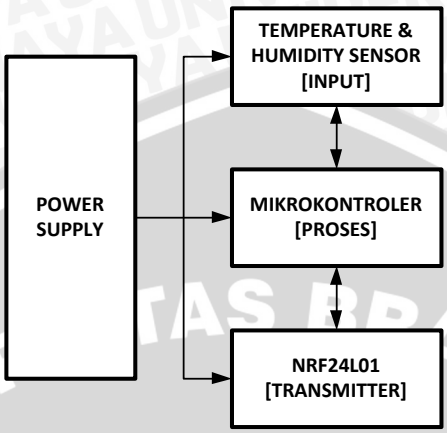
Berdasarkan Gambar 3.2 perancangan sistem dapat dijelaskan pada poin-poin berikut:

1. Masing-masing MQTT-SN *client* bertugas mengirim hasil akuisisi data sensor berupa nilai suhu dan kelembaban menuju MQTT-SN *gateway* menggunakan komunikasi *wireless* dengan format pesan MQTT-SN
2. MQTT-SN *gateway* bertugas menerima dan meneruskan pesan yang diterima dari MQTT-SN *client* menuju MQTT *publisher* menggunakan komunikasi serial.
3. MQTT *publisher* bertugas menerima dan memetakan pesan sesuai *topic* yang didapat dari MQTT-SN *gateway* kemudian meneruskan pesan menuju MQTT *subscriber* melalui *broker*.
4. MQTT *subscriber* berupa aplikasi web, menerima data dengan memanfaatkan port websoket yang disediakan oleh *broker*.

Dari Gambar 3.2 diatas, perancangan sistem pada lingkungan *sensor network* menjadi 2 bagian yaitu perancangan node dan perancangan penyetaraan waktu serta penjadwalan.

3.3.1 Perancangan Node WSN

Perancangan node WSN dalam sistem ini terdiri dari MQTT-SN *client* dan MQTT-SN *gateway.* MQTT-SN *client* memiliki arsitektur yang dijelaskan pada diagram dibawah ini.

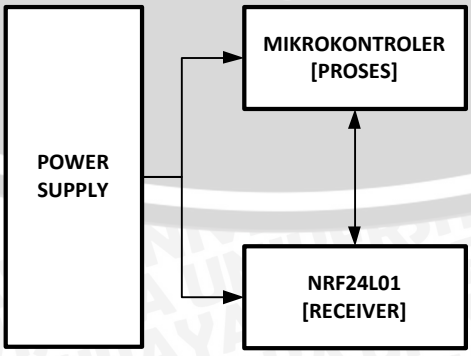


Gambar 3.3 Diagram node klien

Penjelasan perancangan MQTT-SN *client*:

1. Power Supply bertindak sebagai sumber daya node agar dapat menjalankan fungsinya yaitu mengukur besaran temperatur dan kelembaban. Power yang dibutuhkan berupa tegangan sebesar 5 Volt.
2. Temperatur dan *Humidity* Sensor dalam penelitian ini menggunakan sensor jenis DHT11 yang bertindak sabagai media pengukur objek dan dapat disebut modul *input*. Sensor DHT11 mengukur suhu dan kelembaban yang nilainya dikirim oleh klien menuju *gateway*.
3. Mikrokontroler dalam penelitian ini menggunakan jenis Arduino Nano v3.0 yang bertindak sebagai perangkat pengolah data hasil *sensing* dan sebagai pengontrol komunikasi dengan *receiver*/*gateway*.
4. *Wireless module* dalam penelitian ini menggunakan jenis NRF24L01 yang pada klien bertindak sebagai media transmisi.

Sedangkan MQTT-SN *gateway* memiliki arsitektur yang tidak berbeda dengan MQTT-SN *client*, dijelaskan pada diagram dibawah ini:



Gambar 3.4 Diagram node *gateway*

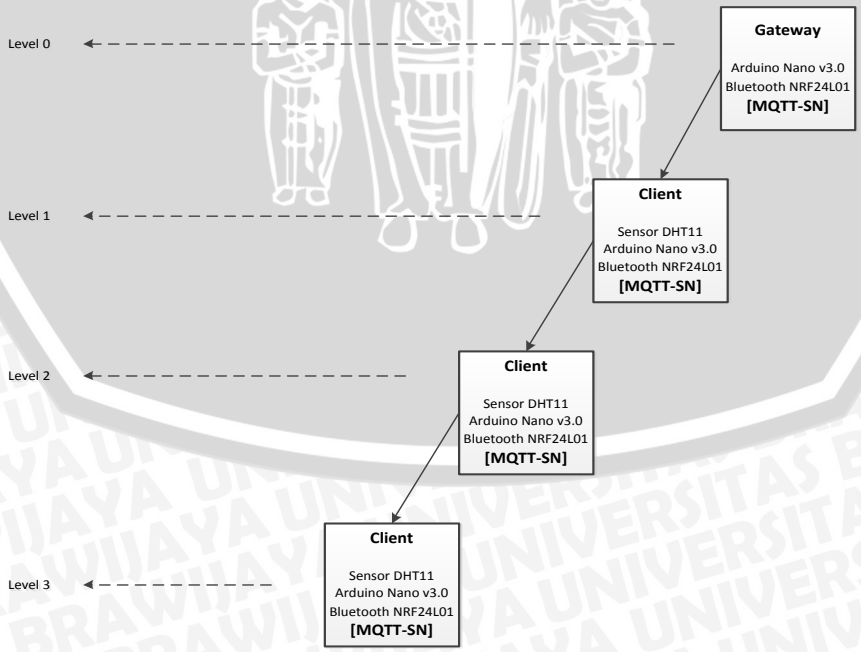
Penjelasan perancangan MQTT-SN *gateway*:

1. Power Supply bertindak sebagai sumber daya node agar dapat menjalankan fungsinya yaitu mengukur besaran temperatur dan kelembaban. Power yang dibutuhkan berupa tegangan sebesar 5 Volt.
2. Mikrokontroler dalam penelitian ini menggunakan jenis Arduino Nano v3.0 yang bertindak sebagai perangkat pengolah data hasil *sensing* dan sebagai pengontrol komunikasi dengan *transmitter/*klien.
3. *Wireless module* dalam penelitian ini menggunakan jenis NRF24L01 yang pada klien bertindak sebagai media receiver.

3.3.2 Perancangan TPSN dan TDMA

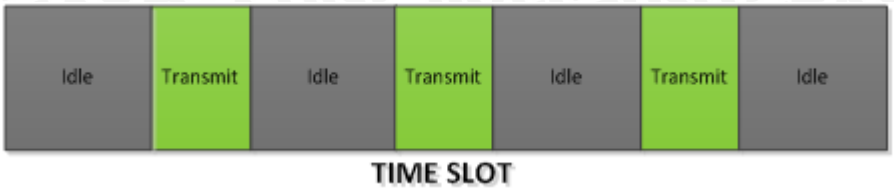
Perancangan topologi hirarki dengan protokol TPSN pada arsitektur aggregating *gateway* terbagi menjadi level 0 sampai level 3. Level 1-3 diberikan secara urut sesuai dengan urutan MQTT-SN *client* yang aktif sedangkan level 0 secara *default* diduduki oleh MQTT-SN *gateway*. Diagram perancangan topologi hirarki menggunakan protokol TPSN ditunjukkan pada Gambar 3.5. TPSN bekerja melalui dua fase. Fase pertama adalah fase *discovery*, yaitu fase pembentukan topologi hirarki dengan cara menentukan level MQTT-SN *client*. Sedangkan fase kedua adalah fase *synchronization*, yaitu fase untuk mencocokkan waktu antara MQTT-SN *client* dengan MQTT-SN *gateway*.

Fase *discovery* mengirimkan paket dimulai dari level *root* kemudian diterima oleh level berikutnya (level 1) setelah node pada level 1 menerima, maka paket akan diteruskan pada level selanjutnya begitu seterusnya sampai semua node tergabung dalam topologi. Fase sinkronisasi waktu adalah fase penyetaraan waktu semua klien untuk disinkronkan dengan waktu yang dimiliki *gateway*. Sama seperti fase pertama, sinkronisasi juga dimulai dari level paling kecil. Sebuah node akan mengirim paket *synchronization* yang berisi waktu lokal dari node itu sendiri.



Gambar 3.5 Perancangan TPSN dan TDMA

Perancangan penjadwalan dengan metode *Time Division Multiple Access* (TDMA) adalah dengan membuat slot waktu transmit dan waktu *idle* seperti pada Gambar 3.6. Setelah itu dilakukan pengiriman data secara bergantian antar *klien* pada slot waktu tertentu sesuai dengan ketentuan konfigurasi.



Gambar 3.6 Ilustrasi Perancangan Pembagian Slot Waktu
Sumber: (Darmawan, 2016)

3.4 Implementasi Sistem

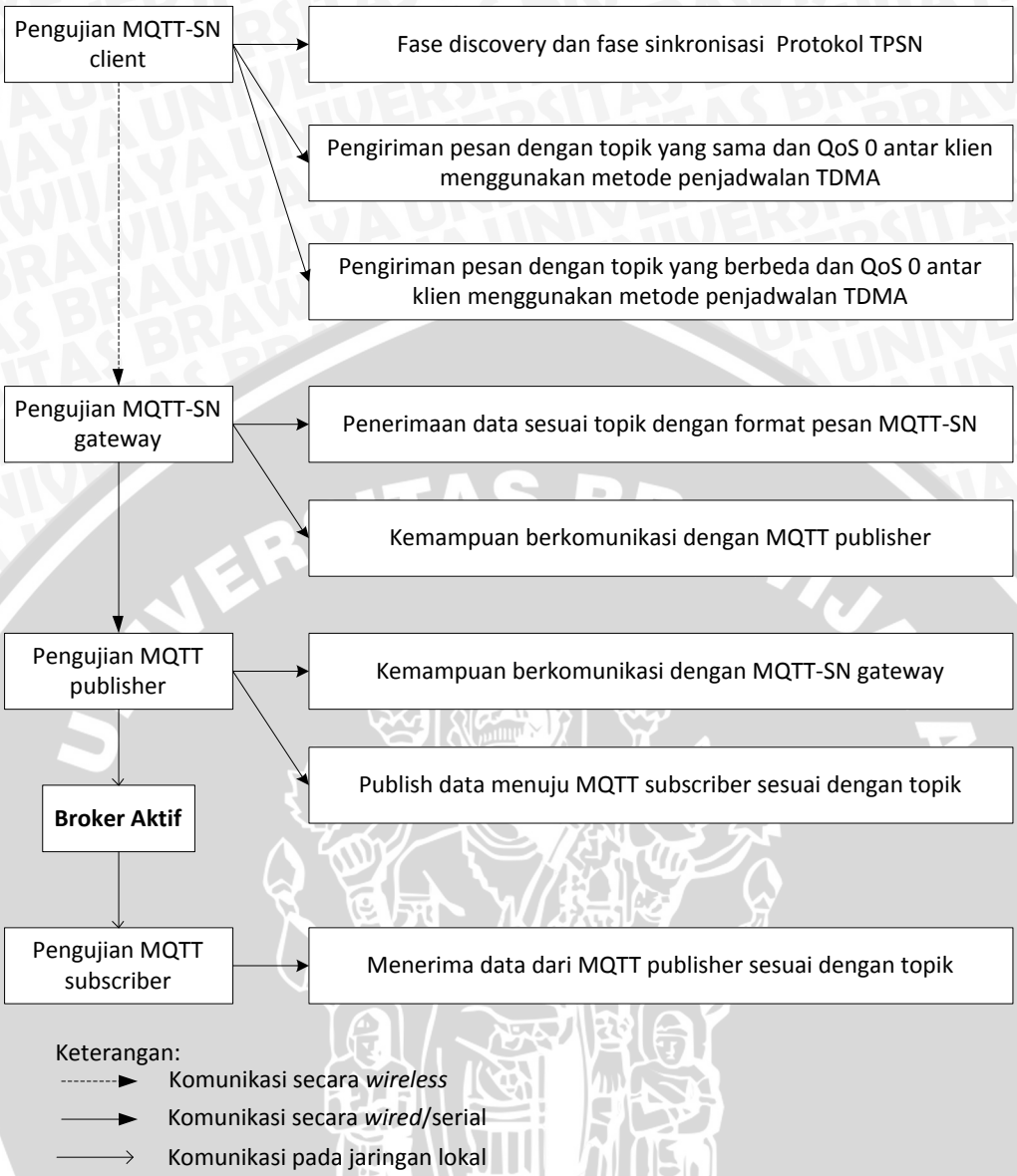
Pada tahap implementasi dilakukan perancangan rangkaian sensor pada MQTT-SN *client* yang tediri dari sensor suhu dan kelembaban, mikrokontroler dan modul komunikasi. MQTT-SN *client* dapat melakukan akuisisi data sensor berupa data temperatur dan kelembaban dalam ruangan. Kemudian data dikirim menuju *MQTT-SN gateway* melalui komunikasi *wireless*. Sebelum melakukan pengiriman pesan pada MQTT-SN *client* ini harus terlebih dulu dapat melakukan sinkronisasi waktu antara satu *klien* dengan *klien* lainnya menggunakan metode TPSN. Selanjutnya dilakukan pengiriman secara bergantian sesuai dengan slot waktu yang telah ditentukan menggunakan metode TDMA.

Pengiriman pesan MQTT-SN *client* menggunakan format pesan protokol MQTT-SN. MQTT-SN *gateway* juga harus dapat menerima pesan menggunakan protokol MQTT-SN. MQTT-SN *gateway* meneruskan pesan yang telah diterima dari MQTT-SN *client* menuju *broker* melalui MQTT *publisher* menggunakan komunikasi serial. MQTT *Subscriber* harus dapat mengumpulkan data yang di*publish* oleh MQTT *publisher* melalui *broker*, kemudian menampilkan data secara *realtime* melalui grafik menggunakan aplikasi *web* yang memanfaatkan komunikasi port *websocket* pada *broker.*

3.5 Pengujian Sistem

3.5.1 Alur Pengujian

Alur pengujian sistem digambarkan pada Gambar 3.7, terdapat 8 poin pengujian yang harus dipenuhi dalam penelitian ini.



Gambar 3.7 Alur Pengujian Sistem

3.5.2 Prosedur Pengujian

Untuk mendapatkan hasil pengujian yang baik dan dapat dianalisis, maka diperlukan sebuah prosedur pengujian. Prosedur pengujian berkaitan dengan kebutuhan fungsional yang diuji pada setiap komponen.

3.5.2.1 Prosedur Pengujian MQTT-SN *Client*

Kebutuhan fungsional MQTT-SN *client* telah dijelaskan pada sub bab 3.5.1, dengan demikian maka dapat dibentuk 5 tabel prosedur pengujian MQTT-SN *client*.

Tabel 3.1 Prosedur Pengujian Fase *Discovery*

Kasus Pengujian	Proses pembentukan topologi hirarki dengan Fase *Discovery*.
Objek Pengujian	Kesesuaian penempatan sebuah *child* ditinjau dari alamat *parent* dan level hirarki; alamat acak baru yang berbeda antar *child* dalam topologi hirarki.
Tujuan Pengujian	Membuktikan paket *discovery* yang dikirimkan dapat membentuk topologi hirarki secara bertahap pada tiap-tiap level.
Prosedur Pengujian	1. Penguji mengunggah kode program untuk ketiga klien/*child* dan kode program untuk satu *gateway*. 2. Penguji mengaktifkan *gateway* sebagai *root* atau *parent* (Level 0). 3. Node klien diaktifkan secara bergantian dengan syarat, setiap klien telah berhasil melakukan sinkronisasi dengan node *root*-nya. 4. Penguji mencatat alamat baru, alamat *parent* dan level dari node baru yang bergabung. 5. Pengujian dilakukan sebanyak 5 kali percobaan.

Tabel 3.2 Prosedur Pengujian Fase Sinkronisasi

Kasus Pengujian	Proses penyetaraan waktu antar *child* pada topologi hirarki.
Objek Pengujian	Keberhasilan *child* melakukan permintaan sinkronisasi; keberhasilan *parent* menjawab permintaan sinkronisasi;
Tujuan Pengujian	Membuktikan paket sinkronisasi yang dikirimkan oleh klien/*child* dapat diterima oleh *parent* dan membuktikan bahwa *parent* dapat menjawab permintaan sinkronisasi.
Prosedur Pengujian	1. Penguji mengunggah kode program untuk ketiga klien/*child* dan kode program untuk satu *gateway*. 2. Penguji menunggu fase *discovery* selesai dan topologi hirarki telah terbentuk. 3. Penguji mencatat durasi waktu tunggu sebelum dilakukan proses sinkronisasi. 4. Penguji mengamati dan mencatat keberhasilan pemintaan sinkronisasi dan jawaban permintaan sinkronisasi. 5. Pengujian dilakukan sebanyak 5 kali percobaan.

Tabel 3.3 Prosedur Pengujian Pengiriman Pesan dengan *Topic* yang sama

Kasus Pengujian	Proses pengiriman pesan menggunakan metode TDMA dengan *topic* antar klien sama.
Objek Pengujian	Keberhasilan klien mengirimkan pesan menuju *gateway* sesuai dengan slot waktu TDMA.
Tujuan Pengujian	Membuktikan pesan format MQTT-SN dengan *topic* sama antar klien dapat dikirim menggunakan metode TDMA.
Prosedur Pengujian	1. Penguji mengunggah kode program untuk ketiga klien dan kode program untuk satu *gateway*. 2. Penguji menunggu klien telah selesai melakukan sinkronisasi waktu. 3. Penguji menunggu klien berhasil mengirimkan pesan menuju *gateway*. 4. Penguji mencatat level klien, slot waktu, waktu pengiriman dan *topic* masing-masing klien. 5. Penguji mengamati keberhasilan pengiriman dan keberhasilan penerimaan. 6. Pengujian dilakukan sebanyak 5 kali percobaan selama 5 menit.

Tabel 3.4 Prosedur Pengujian Pengiriman Pesan dengan *Topic* yang berbeda

Kasus Pengujian	Proses pengiriman pesan menggunakan metode TDMA dengan *topic* antar klien berbeda.
Objek Pengujian	Keberhasilan klien mengirimkan pesan menuju *gateway* sesuai dengan slot waktu TDMA.
Tujuan Pengujian	Membuktikan pesan format MQTT-SN dengan *topic* berbeda antar klien dapat dikirim menggunakan metode TDMA.
Prosedur Pengujian	1. Penguji mengunggah kode program untuk ketiga klien dan kode program untuk satu *gateway*. 2. Penguji menunggu klien telah selesai melakukan sinkronisasi waktu. 3. Penguji menunggu klien berhasil mengirimkan pesan menuju *gateway*. 4. Penguji mencatat level klien, slot waktu, waktu pengiriman dan *topic* masing-masing klien. 5. Penguji mengamati keberhasilan pengiriman dan keberhasilan penerimaan. 6. Pengujian dilakukan sebanyak 2x5 kali percobaan. 7. Percobaan 1x5 tahap pertama, 2 diantara 3 klien memiliki *topic* berbeda. Percobaan dilakukan selama 5 menit.

	8. Percobaan 1x5 tahap kedua, masing-masing klien memiliki *topic* yang berbeda. Percobaan dilakukan selama 7 menit.

3.5.2.2 Prosedur Pengujian MQTT-SN *Gateway*

Kebutuhan fungsional MQTT-SN *gateway* telah dijelaskan pada sub bab 3.5.1, dengan demikian maka disusun tabel prosedur pengujian MQTT-SN *gateway*.

Tabel 3.5 Prosedur Pengujian MQTT-SN *Gateway*

Kasus Pengujian	Komunikasi MQTT-SN *gateway* dengan MQTT-SN *client* melalui media nirkabel.
Objek Pengujian	MQTT-SN *gateway* dapat melakukan *broadcast* paket *discovery*; MQTT-SN *gateway* dapat menjawab permintaan sinkronisasi dari klien; MQTT-SN *gateway* dapat menerima data sesuai dengan slot waktu yang telah ditentukan.
Tujuan Pengujian	Membuktikan *root* dapat mengirimkan paket *discovery* selama 7 detik; Membuktikan *root* dapat mengirimkan paket sinkronisasi kepada *child*nya; dan membuktikan *root* dapat menerima data dengan format pesan MQTT-SN sesuai dengan slot waktu TDMA.
Prosedur Pengujian	1. Penguji mengunggah kode program untuk ketiga klien dan kode program untuk satu *gateway*. 2. Penguji mengaktifkan *gateway* sebagai *root* atau node pertama (Level 0). 3. Penguji mengamati keberhasilan proses *discovery*, sinkronisasi dan penerimaan data dari klien.

3.5.2.1 Prosedur Pengujian Simulasi MQTT

Kebutuhan fungsional simulasi MQTT telah dijelaskan pada sub bab 3.5.1, dengan demikian maka dapat dibentuk tabel prosedur pengujian simulasi MQTT. Prosedur pengujian terbagi menjadi 2 kasus, yaitu prosedur pengujian untuk kasus MQTT *publisher* dan untuk kasus MQTT *subscriber*.

Tabel 3.6 Prosedur pengujian MQTT *publisher*

Kasus Pengujian	Fungsionalitas MQTT *publisher*
Objek Pengujian	MQTT *publisher* dapat berkomunikasi dengan MQTT-SN *gateway* menggunakan port serial; MQTT *publisher* dapat mem*publish* data sesuai dengan *topic*.
Tujuan Pengujian	Membuktikan MQTT *publisher* dapat menggantikan fungsi serial monitor MQTT-SN *gateway*; membuktikan bahwa data pada serial monitor

	MQTT-SN *gateway* dapat diseleksi melalui *topic* untuk di*publish* menuju *subscriber*.
Prosedur Pengujian	1. Penguji mengaktifkan aplikasi *broker* apache activemq hingga *broker* dalam keadaan ready. 2. Penguji menjalankan program aplikasi *publisher* melalui command prompt. 3. Penguji mengamati luaran aplikasi MQTT *publisher* yang serupa luaran MQTT-SN *gateway* pada serial monitor.

Tabel 3.7 Prosedur pengujian MQTT *subscriber*

Kasus Pengujian	MQTT *subscriber* dapat berlangganan *topic* dari MQTT *publisher*
Objek Pengujian	MQTT *publisher* dapat mem*publish* data sesuai dengan *topic*; MQTT *subscriber* menampilkan data yang telah di*publish* oleh MQTT *publisher* melalui halaman web
Tujuan Pengujian	Membuktikan *broker* dapat menjadi jembatan bagi MQTT *publisher* dan MQTT *subscriber*; Membuktikan MQTT *publisher* dapat memetakan data sesuai dengan *topic*nya; Membuktikan MQTT *subscriber* dapat menampilkan data sesuai dengan *topic* pada halaman web.
Prosedur Pengujian	1. Penguji mengaktifkan aplikasi *broker* apache activemq hingga *broker* dalam keadaan ready. 2. Penguji menjalankan program aplikasi *publisher* melalui command prompt. 3. Penguji menjalankan aplikasi MQTT *subscriber* pada halaman browser. 4. Penguji mengamati data yang berhasil di*subscribe* oleh MQTT *subscriber*.

BAB 4 REKAYASA KEBUTUHAN

Pada bab rekayasa kebutuhan akan dijelaskan tentang kebutuhan sistem secara umum. Dimulai dari gambaran sistem, kebutuhan sistem dan alur kerja sistem.

4.1 Gambaran Umum Sistem

Arsitektur *aggregating gateway* protokol MQTT-SN menerapkan *single gateway* sebagai perantara lingkungan *sensor network* dengan lingkungan simulasi MQTT. Hal ini menimbulkan permasalahan pada proses pengiriman data jika MQTT-SN *client* berjumlah lebih dari satu. Permasalahan yang dapat terjadi pada *multiple* MQTT-SN *client* saat pengiriman data yaitu *collision* atau tabrakan data.

Untuk mengatasi permasalahan tersebut dibutuhkan penjadwalan pengiriman data pada *multiple* MQTT-SN *client*. Pada penelitian ini penjadwalan pengiriman data menggunakan protokol TDMA (*Time Division Multiple Access*) yaitu metode pembagian waktu pengiriman. Untuk menjadwalkan pengiriman data antara klien satu dengan klien lainnya, terlebih dahulu antar klien harus memiliki waktu yang sinkron.

Sinkronisasi waktu antar MQTT-SN *client* menggunakan protokol TPSN (*Time-Syncronization Protocol for Sensor network*) yaitu protokol sinkronisasi waktu yang menerapkan metode konvesional klien server. TPSN menerapkan struktur hirarki saat proses sinkronisasi waktu sehingga mudah untuk menambahkan klien pada jaringan tersebut.

4.2 Kebutuhan Sistem

Kebutuhan sistem berisi penjelasan kebutuhan fungsional, kebutuhan komunikasi, kebutuhan perangkat keras dan kebutuhan perangkat lunak dianalisis sesuai dengan kebutuhan sistem sehingga mempermudah dalam mendesain dan mengimplementasikan sistem.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional yang harus dipenuhi pada penelitian ini antara lain:

1. **Masing-masing MQTT-SN *client* memiliki waktu setara antara satu dengan yang lain**

   Pada penelitian ini menggunakan MQTT-SN *client* yang berjumlah 3 buah node. Masing-masing dari 3 node tersebut terdiri dari arduino nano sebagai mikrokontroler, NRF24L01 sebagai modul komunikasi berbasis *wireless* dan sensor DHT 11 sebagai modul input data suhu dan kelembaban. Dilakukan penyetaraan waktu pada 3 node tersebut menggunakan prokol TPSN yang menerapkan struktur hirarki. Sehingga pada implementasinya masing-masing node berada pada level-level yang berbeda.

2. **Masing-masing MQTT-SN *client* dapat mengirimkan data dengan *topic* yang sama menggunakan protokol TDMA**

   MQTT-SN *client* yang bertindak sebagai transmitter dengan melakukan pengukuran suhu dan kelembaban. Data sensor kemudian dikirim menuju MQTT-SN *gateway* dengan protokol penjadwalan TDMA. Data sensor dikemas menggunakan format pesan MQTT-SN dengan *topic* yang sama antar 3 klien. Pengiriman data sensor menuju *gateway* dibagi berdasarkan waktu agar dapat mebgurangi resiko tabrakan data antar klien.

3. **MQTT-SN *client* dapat melakukan pengiriman data dengan *topic* yang berbeda menggunakan protokol TDMA.**

   MQTT-SN *client* yang bertindak sebagai transmitter dengan melakukan pengukuran suhu dan kelembaban. Data sensor kemudian dikirim menuju MQTT-SN *gateway* dengan protokol penjadwalan TDMA. Data sensor dikemas menggunakan format pesan MQTT-SN dengan *topic* yang berbeda antar 3 klien. Pengiriman data sensor menuju *gateway* dibagi berdasarkan waktu agar dapat mengurangi resiko tabrakan data antar klien.

4. **MQTT-SN *gateway* dapat melakukan pengiriman data melalui komunikasi serial menuju MQTT *publisher***

   Broker bertindak sebagai jembatan antara MQTT *publisher* dengan MQTT *subscriber*. Dimana MQTT *publisher* bertugas menampung data sensor yang diterima melalui port serial MQTT-SN *gateway*. Data pada MQTT *publisher* di*publish* sesuai *topic* menuju MQTT *subscriber*.

5. **MQTT *subscriber* dapat menerima data dan menampilkannya melalui aplikasi web menggunakan port websocket pada *broker*.**

   MQTT *subscriber* melakukan *subscribe* berdasarkan *topic* kemudian menampilkan data melalui aplikasi *web* dengan memanfaatkan port *websocket* pada *broker* 61614.

4.2.2 Kebutuhan Perangkat Keras

Untuk menunjang implementasi sistem, dibutuhkan perangkat yang memadai. Berikut adalah kebutuhan perangkat keras yang digunakan pada proses implementasi sistem:

1. Komputer

   Komputer digunakan untuk memonitoring pengolahan data pada mikrokontroler. Selain itu juga untuk memberkan input yang dibutuhkan saat program berjalan. Spesifikasi computer dijelaskan sebagai berikut:

   - Model Perangkat: ACER Aspire 4732Z
   - Prosessor: Pentium Dual-core CPU 2,30 GHz
   - RAM: DDR2 2 Gb

2. Arduino Nano v3.0

Mikrokontroler merupakan perangkat yang ditanami program sehingga dapat bekerja sesuai yang diinginkan program tersebut. Selain itu juga merupakan pengolah data yang dikirimkan atau diterima.

3. Modul *Wireless* NRF24L01
   Pada setiap node terdapat modul tranceiver. Dimana modul ini digunakan untuk saling berhubungan dengan node lain. Modul tranceiver menggunakan frekuensi yang sama untuk saling berhubungan.
4. Sensor temperatur dan kelembaban DHT11
   Sensor temperatur dan kelembaban berfungsi untuk mendapatkan data nilai temperatur dan kelembaban di area sekitar jangkauan sensor.

4.2.3 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak pada penelitian ini dijelaskan melalui poin-poin berikut:

1. Arduino IDE 1.6
   Perangkat lunak ini digunakan untuk menuliskan program, kompile dan *upload* program ke mikrokontroler. Selain itu pada perangkat lunak ini terdapat fitur yang dapat digunakan untuk membantu komputer untuk melakukan monitoring jalannya program, yaitu serial monitor.
2. Apache *ActiveMQ*
   Apache ActiveMQ merupakan aplikasi *broker* open source yang cepat dan support dengan berbagai bahasa pemrograman seperti C, C++, Python maupun Java. Pada sistem ini Apache ActiveMQ bertindak sebagai server atau jembatan antara *publisher* dengan *subscriber* yang berjalan pada port 61614. Tidak hanya menghubungkan *publisher* dan *subscriber*, fitur lain yang digunakan pada sistem ini adalah websoket. Fitur *websocket* digunakan agar *realtime* web aplikasi dapat berjalan sesuai fungsinya.
3. Paho Eclipse *Library*
   Paho merupakan *library* yang digunakan untuk aplikasi *publisher* dan *subscriber* agar dapat berjalan sebagai klien.

4.3 Batasan Desain Sistem

Pada proses penerapannya, sistem ini memiliki beberapa keterbatasan yang disebabkan dari spesifikasi hardware yang digunakan pada sistem. Seperti keterbatasan tingkat akurasi sensor DHT11 dengan tingkat error sebesar 2^{o}C dan hanya mampu mengakuisisi data antara 0 ^{o}C hingga 5^{o}C selain itu, sensor juga memiliki keterbatasan tingkat akurasi kelembaban sebesar 5% RH dan hanya mampu mengakuisisi data antara 20% hingga 90% RH (D-Robotics UK, 2010). Sedangkan modul *wireless* yaitu NRF24L01 memiliki keterbatasan jarak transmisi 70-100 meter dengan lingkungan bebas interferensi gelombang (Ball, 2008). Agar implementasi sistem realtime monitoring temperatur dan kelembaban ruang data center berbasis protokol MQTT-SN dapat dilakukan secara terarah dan sesuai dengan yang diharapkan, maka perlu diterapkan batasan-batasan implementasi desain sistem, antara lain :

1. Sistem menggunakan mikrokontroler jenis arduino nano v3.0.

2. Sistem menggunakan sensor temperatur dan kelembaban jenis DHT11.
3. Sistem memanfaatkan *Wireless* modul NRF2401 sebagai media komunikasi antar node.
4. MQTT-SN *client* terdiri dari 3 node yang masing-masing mengirim 2 *topic* berupa suhu dan kelembaban menuju *gateway*.
5. Alamat MQTT-SN *client* dapat diketahui setelah fase *discovery* berhasil dilakukan.
6. Pengalamatan MQTT-SN *client* dilakukan secara acak dengan rentang alamat sepanjang 4 karakter bilangan.
7. Alamat MQTT-SN *gateway* telah diketahui.
8. MQTT-SN *gateway* secara *default* berfungsi sebagai *root* pada level 0.
9. Penyetaraan waktu menggunakan metode TPSN hanya dilakukan pada lingkungan *sensor network*.
10. Implementasi protokol MQTT-SN menggunakan *QoS* 0.
11. *Topic* yang ada pada sistem diregistrasi terlebih dahulu.
12. Lingkungan simulasi MQTT dan *Broker* dilakukan di jaringan lokal.
13. Aplikasi *broker* yang digunakan adalah Apache Activemq.
14. *Subscriber* mampu menerima data dari *broker* dan menampilkan melalui aplikasi web.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan perancangan dan proses implementasi sistem yang terdiri dari perancangan perangkat keras, perancangan perangkat lunak, perancangan algoritma, perancangan format payload, alur diagram tiap perangkat. Implemetasi sistem terdiri dari implementasi perangkat keras, implementasi perangkat lunak, implementasi algoritma, implementasi format payload.

5.1 Perancangan Sistem

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

5.1.1 Perancangan Perangkat Keras

5.1.1.1 Perancangan MQTT-SN *Client*

MQTT-SN *client* berperan sebagai perangkat pengambil data, karena itu MQTT-SN *client* terdiri dari sensor temperatur dan kelembaban sebagai media pengukur temperatur dan kelembaban di dalam ruangan. Juga terdapat mikrokontroler sebagai modul pemroses data hasil ukur, perangkat lain yang terdapat pada MQTT-SN *client* yaitu modul *Wireless* yang bertindak sebagai media transmisi data. Tabel 5.1 merupakan tabel spesifikasi sensor DHT11 sedangkan Tabel 5.2 merupakan tabel pin yang digunakan pada sensor DHT11.

Tabel 5.1 Spesifikasi Sensor DHT11

Supply Voltage	3- 5 Volt
Temperature range	0-50 °C Error ± 2 °C
Humidity	20-90% RH Error ± 5% RH

Tabel 5.2 Spesifikasi Pin DHT11

VCC	Power Source
Signal	Data
Gnd	Ground

Selain sensor perangkat yang dibutuhkan oleh MQTT-SN *client* adalah mikrokontroler, pada sistem ini mikrokontroler yang digunakan adalah Arduino Nano. Tabel 5.3 menjelaskan Spesifikasi Arduino Nano v3.0 juga pin-pin yang digunakan.

Tabel 5.3 Spesifikasi Arduino Nano v3.0 dan Pin

Mikrontroler	Atmel Atmega328
Voltage Processing	5 Volt
Input Voltage	6-20 Volt

Current per I/O pin	40 mA
Pin Digital	D2-D13
Pin Analog	A0-A7

MQTT-SN *client* selain bertugas melakukan akuisisi data, tugas lainnya adalah mengirimkan hasil akuisisi data ke MQTT-SN *gateway* untuk kemudian diteruskan menuju jaringan lokal. Pengiriman data hasil akuisisi memanfaatkan komunikasi *wireless* yang mana pada sistem ini menggunakan modul *Wireless* NRF24L01. Tabel 5.4 menjelaskan spesifikasi operasional modul *Wireless* NRF24L01 dan Tabel 5.5 berisi penjelasan fungsi tiap pinnya.

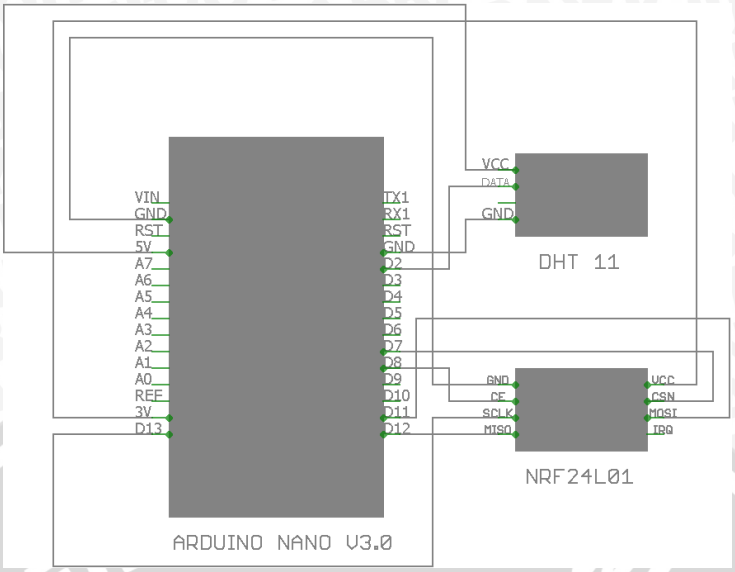
Tabel 5.4 Spesifikasi Modul *wireless* NRF24L01

Power supply	1.9V~3.6V
I/O port working voltage	0~3.3, 5 V
I/O port working current	13.5mA at 2Mbps
Data rate	256kbps / 1Mbps / 2Mbps
Receiving sensitivity	-85dBm at 1Mbps
Transmission range	70~100 meter at 256kbps

Tabel 5.5 Pin modul *wireless* NRF24L01

CE	Enable RX/TX Mode
CSN	SPI Chip select
SCLK	SPI Clock
MOSI	SPI data in
MISO	SPI data out
IRQ	Interrupt Pin
VCC	Power Supply
GND	Ground (0 Volt)

Agar MQTT-SN *client* dapat melakukan akuisisi data, oleh karena itu ketiga modul tersebut dirangkai seperti rancangan rangkaian pada Gambar 5.1.



Gambar 5.1 Rancangan Rangkaian MQTT-SN *client*

5.1.1.2 Perancangan MQTT-SN *Gateway*

Pada sistem ini, MQTT-SN *gateway* menerima pesan yang berisi hasil akuisisi data dari MQTT-SN *client*. MQTT-SN *Gateway* terdiri dari mikrokontroler, pada sistem ini menggunakan Arduino Nano v3.0 dan modul komunikasi yaitu nrf24l01. Spesifikasi perangkat yang digunakan untuk MQTT-SN *gateway* sama dengan yang digunakan untuk MQTT-SN *client*.

Perangkat yang dibutuhkan oleh MQTT-SN *gateway* salah satunya adalah mikrokontroler. Sistem ini menggunakan mikrokontroler jenis Arduino Nano. Tabel 5.6 menjelaskan Spesifikasi Arduino Nano v3.0 juga pin-pin yang digunakan.

Tabel 5.6 Spesifikasi Arduino Nano v3.0 dan Pin

Mikrontroler	Atmel Atmega328
Voltage Processing	5 Volt
Input Voltage	6-20 Volt
Current per I/O pin	40 mA
Pin Digital	D2-D13
Pin Analog	A0-A7

MQTT-SN *gateway* bertugas sebagai penerima data secara *wireless*. Penerimaan data hasil akuisisi memanfaatkan modul *Wireless* NRF24L01. Tabel 5.7 menjelaskan spesifikasi operasional modul *Wireless* NRF24L01 dan Tabel 5.8 berisi penjelasan fungsi tiap pinnya.

Tabel 5.7 Spesifikasi Modul *wireless* NRF24L01

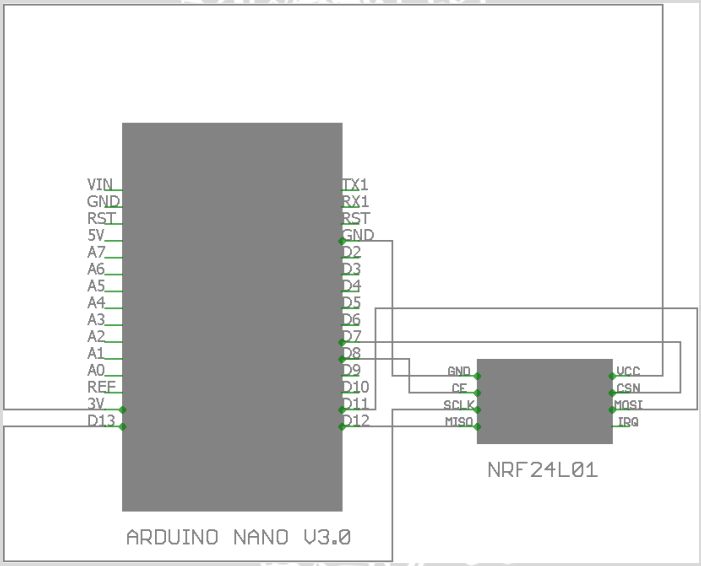
Power supply	1.9V~3.6V
I/O port working voltage	0~3.3, 5 V
I/O port working current	13.5mA at 2Mbps

Data rate	256kbps / 1Mbps / 2Mbps
Receiving sensitivity	-85dBm at 1Mbps
Transmission range	70~100 meter at 256kbps

Tabel 5.8 Pin modul *wireless* NRF24L01

CE	Enable RX/TX Mode
CSN	SPI Chip select
SCLK	SPI Clock
MOSI	SPI data in
MISO	SPI data out
IRQ	Interrupt Pin
VCC	Power Supply
GND	Ground (0 Volt)

Agar *gateway* dapat menjalankan fungsinya, mikrokontroler dan modul *wireless* dirangkai seperti rancangan Gambar 5.2dibawah ini.



Gambar 5.2 Rancangan Rangkaian MQTT-SN *gateway*

5.1.1.3 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.

5.1.2 Perancangan Perangkat Lunak

Pada tahap ini dijelaskan, proses perancangan perangkat keras. Pada bab sebelumnya dijelaskan bahwa perangkat lunak sistem terdiri dari arduino IDE untuk kompilasi kode program WSN *publisher* dan *gateway*, sedangkan untuk simulasi MQTT menggunakan aplikasi *broker* apache activeMQ.

5.1.2.1 Perancangan MQTT-SN *Client*

Perancangan perangkat lunak MQTT-SN *client* mencakup pada fungsionalitas kebutuhan perangkat lunak yang digunakan untuk menulis dan kompilasi program. Pemrograman untuk perangkat MQTT-SN *client* menggunakan bahasa C Arduino dengan software Arduino IDE untuk kompilasinya. *Library* yang digunakan antara lain; *library* DHT digunakan untuk mengaktifkan fungsi sensor DHT11. *Library* SPI digunakan untuk mengaktifkan fungsi komunikasi serial. *Library* mirf dan nrf24l01 digunakan untuk mengaktifkan fungsi modul *wireless* dan komunikasi *wireless* itu sendiri.

5.1.2.2 Perancangan MQTT-SN *Gateway*

Perancangan perangkat lunak MQTT-SN *gateway* mencakup pada fungsionalitas kebutuhan perangkat lunak yang digunakan untuk menulis dan kompilasi program *gateway*. Pemrograman untuk perangkat MQTT-SN *gateway* menggunakan bahasa C Arduino dengan software Arduino IDE untuk kompilasinya. *Library* yang digunakan antara lain; *Library* SPI digunakan untuk mengaktifkan fungsi komunikasi serial. *Library* mirf dan nrf24l01 digunakan untuk mengaktifkan fungsi modul *Wireless* dan komunikasi *wireless* itu sendiri.

5.1.2.3 Perancangan Simulasi MQTT

Perancangan perangkat lunak untuk simulasi MQTT mencakup fungsionalitas aplikasi *broker* yang berperan sebagai jembatan antara *publisher* dan *subscriber* pada simulasi ini. Pada jaringan lokal MQTT terdapat *publisher* dan *subscriber* yang bertindak sebagai klien. Aplikasi *broker* yang digunakan adalah *broker* Apache ActiveMQ (activemq.apache.org, 2016). Klien *publisher* menggunakan *library* paho mqtt yang sudah di-*include* dengan aplikasi *broker* melalui program klien *publisher*.

Pemrograman klien *publisher* menggunakan bahasa python. Selain *library* paho yang berguna mengaktifkan fungsi klien pada simulasi mqtt, *library* lain yang digunakan pada program *publisher* mqtt yaitu *library* serial berguna mengaktifkan fungsi komunikasi serial.

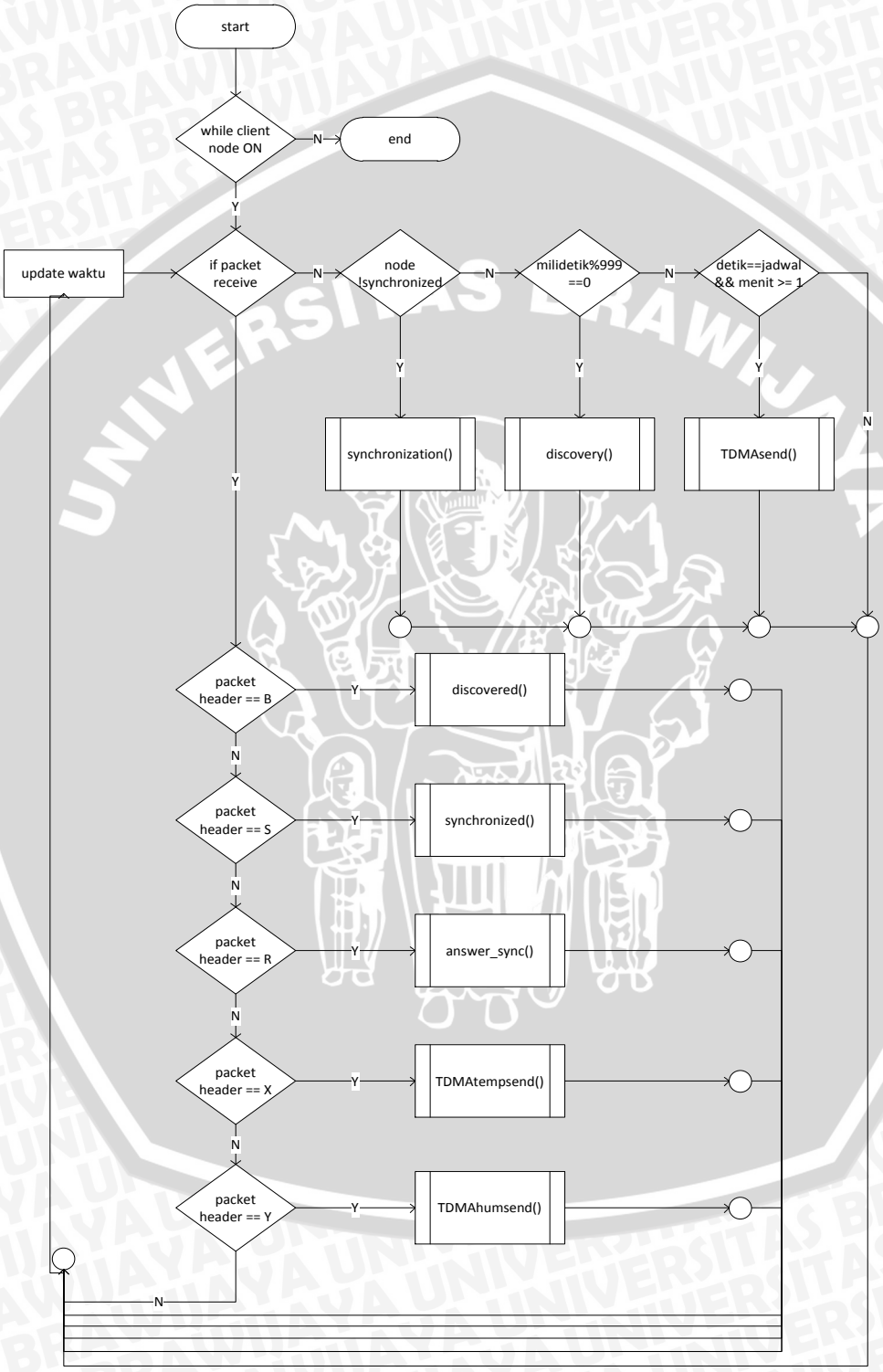
MQTT *subscriber* adalah aplikasi *subscriber* dengan bahasa pemrograman HTML dan javascript yang berfungsi menampilkan hasil akuisisi data dalam bentuk grafik pada halaman web.

5.1.3 Perancangan Algoritma

Subbab ini berisi diagram alir tiap fungsi dan perangkat daam bentuk flow*chart* beserta penjelasannya.

5.1.3.1 Perancangan MQTT-SN *Client*

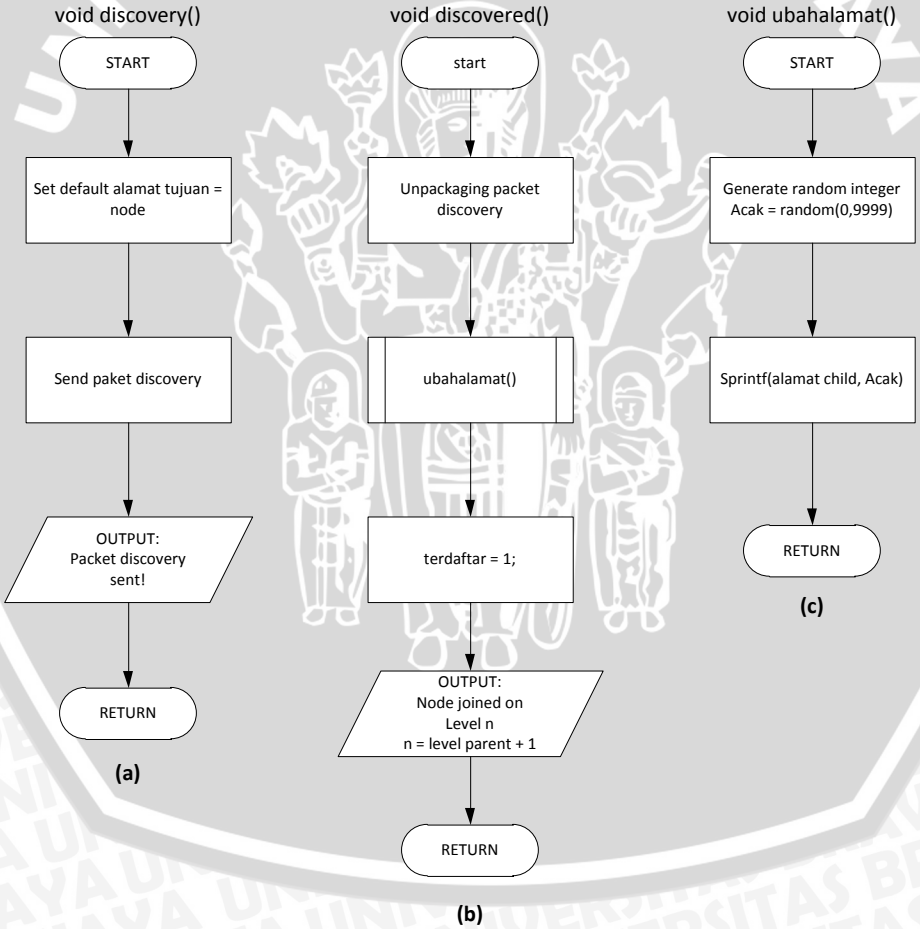
MQTT-SN *client* berfungsi sebagai transmitter yang melakukan akuisisi data nilai temperatur dan kelembaban suatu ruangan. Alur kerja MQTT-SN *client* pada Gambar 5.3 dibawah ini.



Gambar 5.3 Diagram Alir Utama MQTT-SN *client*

Alur kerja sistem MQTT-SN *client* dimulai dari inisialisasi baudarate, pin sensor, channel, payload dan waktu tunggu. Inisialisasi baudarate dilakukan untuk menentukan mikrokontroler bekerja pada rate yang sesuai yaitu 9600. Inisialisasi pin sensor dilakukan untuk menentukan sensor dapat bekerja melalui pin yang juga telah ditentukan. Inisialisasi channel dilakukan untuk menentukan channel komunikasi *wireless* antara MQTT-SN *client* dengan MQTT-SN *gateway*. Inisialisasi payload dilakukan dengan menentukan besar payload yang berisi paket data MQTT-SN, TPSN dan TDMA.

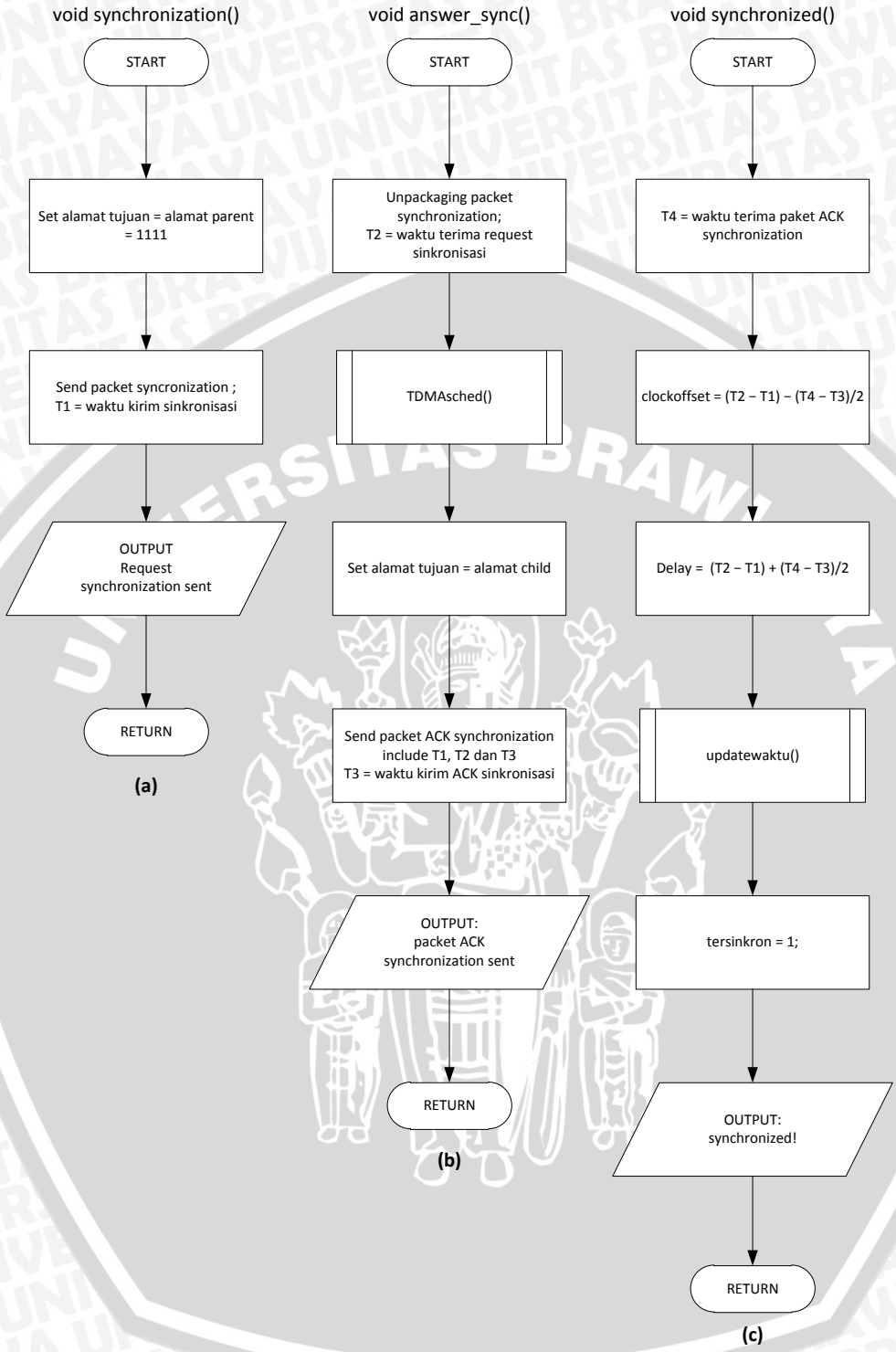
Setelah MQTT-SN *client* aktif maka node melakukan proses sinkronisasi waktu menggunakan metode TPSN dan memastikan apakah node menerima paket data atau tidak. Proses sinkronisasi menggunakan protokol TPSN terdiri dari dua fase. Fase pertama yaitu *discovery* yang berguna untuk menentukan level masing-masing node dan membentuk topologi hirarki. *Root* teratas yaitu MQTT-SN *gateway*. Fase kedua, node dengan level n saling bertukar paket sinkronisasi dengan node n+1. Proses sinkronisasi waktu dilakukan dengan cara dua node tersebut saling berkirim waktu lokal.



Gambar 5.4 Alur Fase *Discovery* MQTT-SN *client*

Pada Gambar 5.4 terdapat 3 *method* yang menjelaskan alur fase *discovery* pada MQTT-SN *client*. MQTT-SN *client* dapat bertindak sebagai *parent* maupun *child* sehingga dapat menjalankan *method discovery*() yang berguna untuk membentuk topologi hirarki dengan memberi inisialisasi level pada *child*nya.

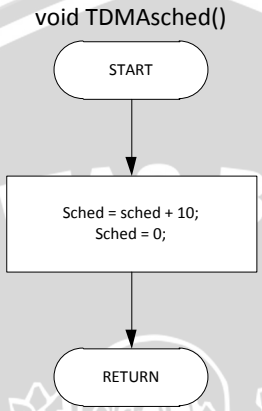
Sedangkan *method* discovered() dijalankan oleh klien saat menerima paket *discovery* dari *parent*nya. Saat menerima paket *discovery* klien akan mendapat level sebagai inisial kedudukannya pada topologi hirarki. Klien juga mengubah alamatnya secara random dengan memanggil *method* ubahalamat(). Setelah fase *discovery* berhasil dilakukan, selanjutnya seluruh node yang ada pada topologi hirarki tersebut melakukan sinkronisasi waktu yaitu fase kedua.



Gambar 5.5 Alur Fase Sinkronisasi MQTT-SN *client*

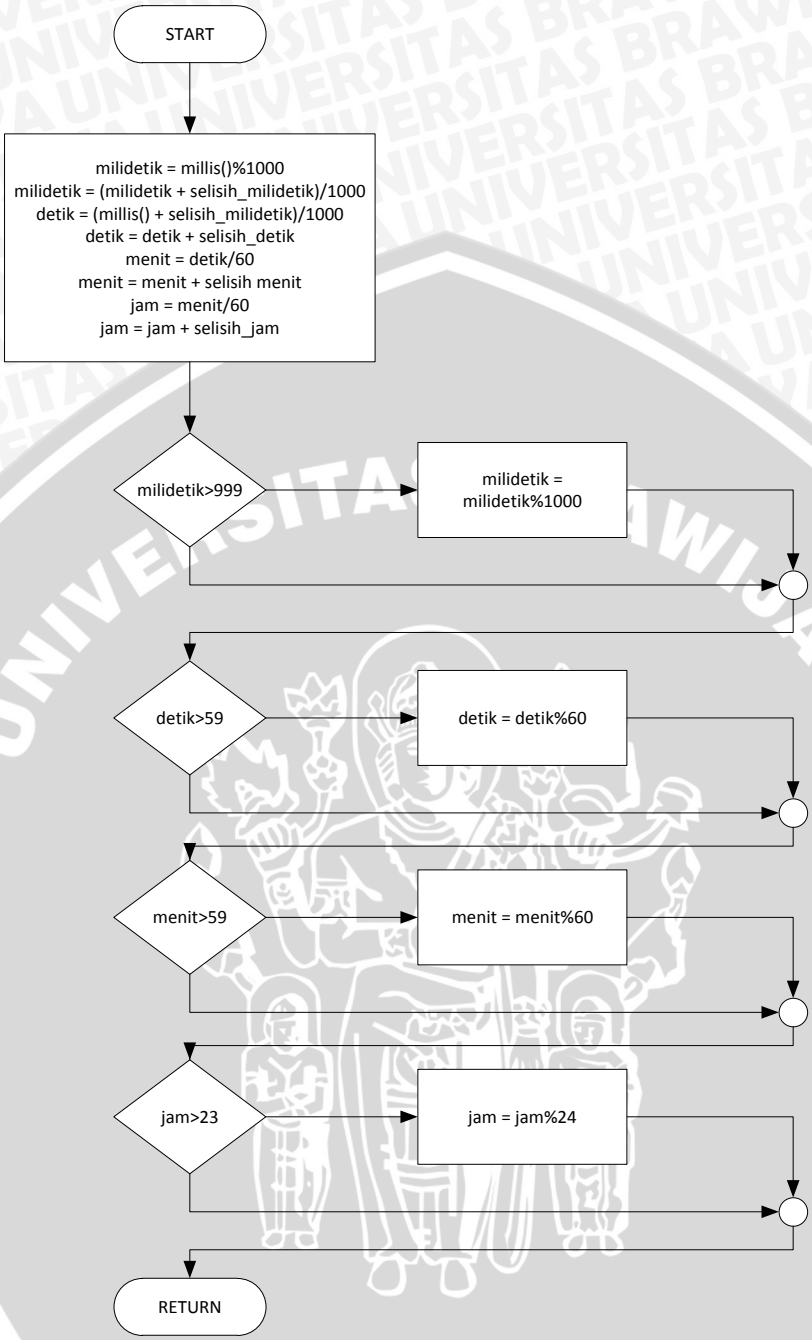
Pada Gambar 5.5 terdapat 3 *method* yang menjelaskan fase sinkronisasi. *Method* pertama yaitu *synchronization*(), *method* ini dijalankan saat fase *discovery* telah selesai dilakukan. *Method synchronization*() mengirimkan paket yang berisi level node dan waktu lokal node saat meakukan permintaan sinkronisasi. *Method* yang kedua yaitu answer_sync(), *method* ini dijalankan oleh node yang menerima permintaan sinkronisasi. *Method* answer_sync()

mengirimkan paket acknowledgment (ACK) yang berisi nilai T1, T2 T3 dan jadwal pengiriman berupa slot waktu dimana penjelasan T1, T2 dan T3 terdapat pada Gambar 5.5. *method* yang terakhir adalah *synchronized*(), yaitu *method* yang dijalankan oleh node yang menerima paket ACK. Alur sinkronisasi seperti yang telah telah dijelaskan pada subbab 2.2.6. Paket ACK diterima pada waktu T4, kemudian node tersebut memperbarui waktunya. Jadwal pengiriman berupa pembagian slot waktu dengan protokol TDMA dijelaskan pada Gambar 5.6.



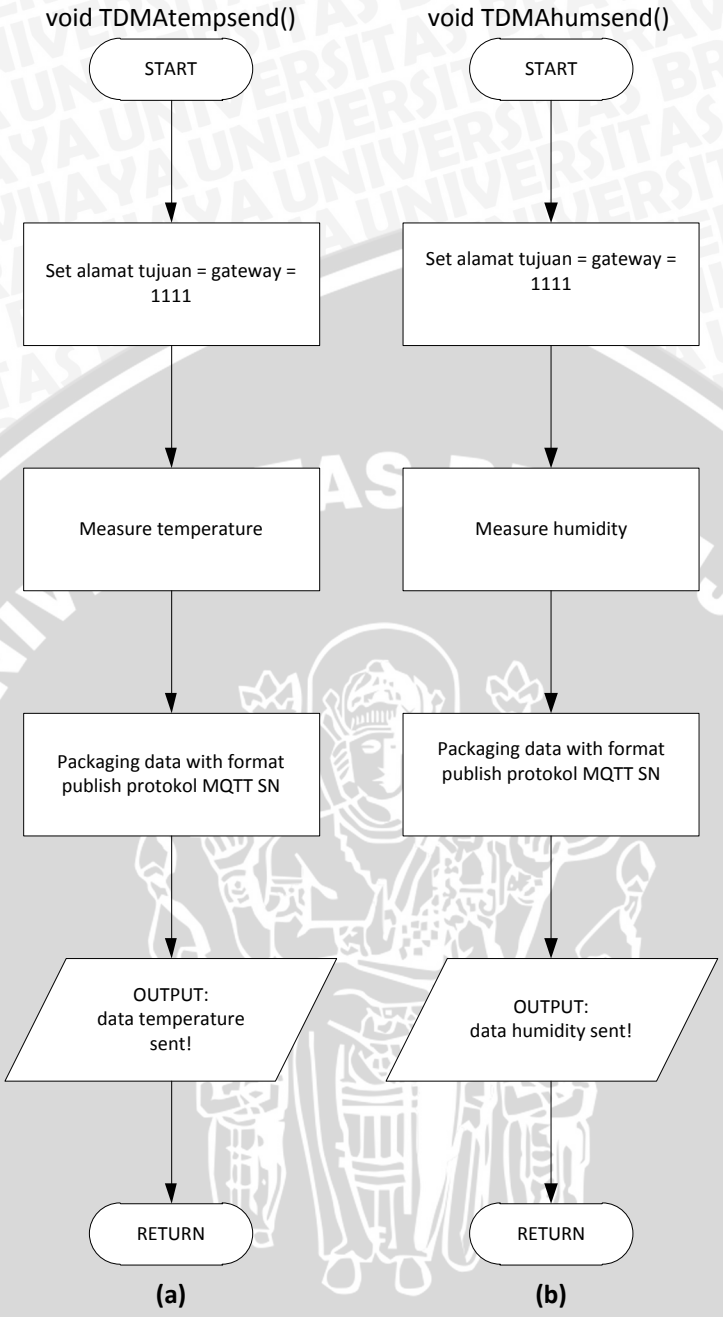
Gambar 5.6 Alur Pembagian Slot Waktu

Pada Gambar 5.7 ditunjukkan sebuah fungsi *update*waktu() yang berguna untuk melakukan pemisahan fungsi milis() menjadi jam, menit, detik, dan milidetik. Variabel selisih_jam, selisih_menit, selisih_detik, dan selisih_milidetik didapatkan dari nilai T4.



Gambar 5.7 Alur *Update* Waktu

Setelah fase sinkronisasi berhasil dilakukan antara node *child* atau MQTT-SN *client* dengan *parent*nya, maka *child* dapat melakukan pengiriman pesan menuju MQTT-SN *gateway*. Pesan yang dikirim menggunakan format pesan publish MQTT-SN dengan QOS 0 sehingga *gateway* tidak mengirimkan pesan ACK. Masing-masing MQTT-SN *client* yang waktunya telah setara dengan waktu MQTT-SN *gateway* dapat melakukan pengiriman pesan dengan *topic* suhu dan kelembaban.

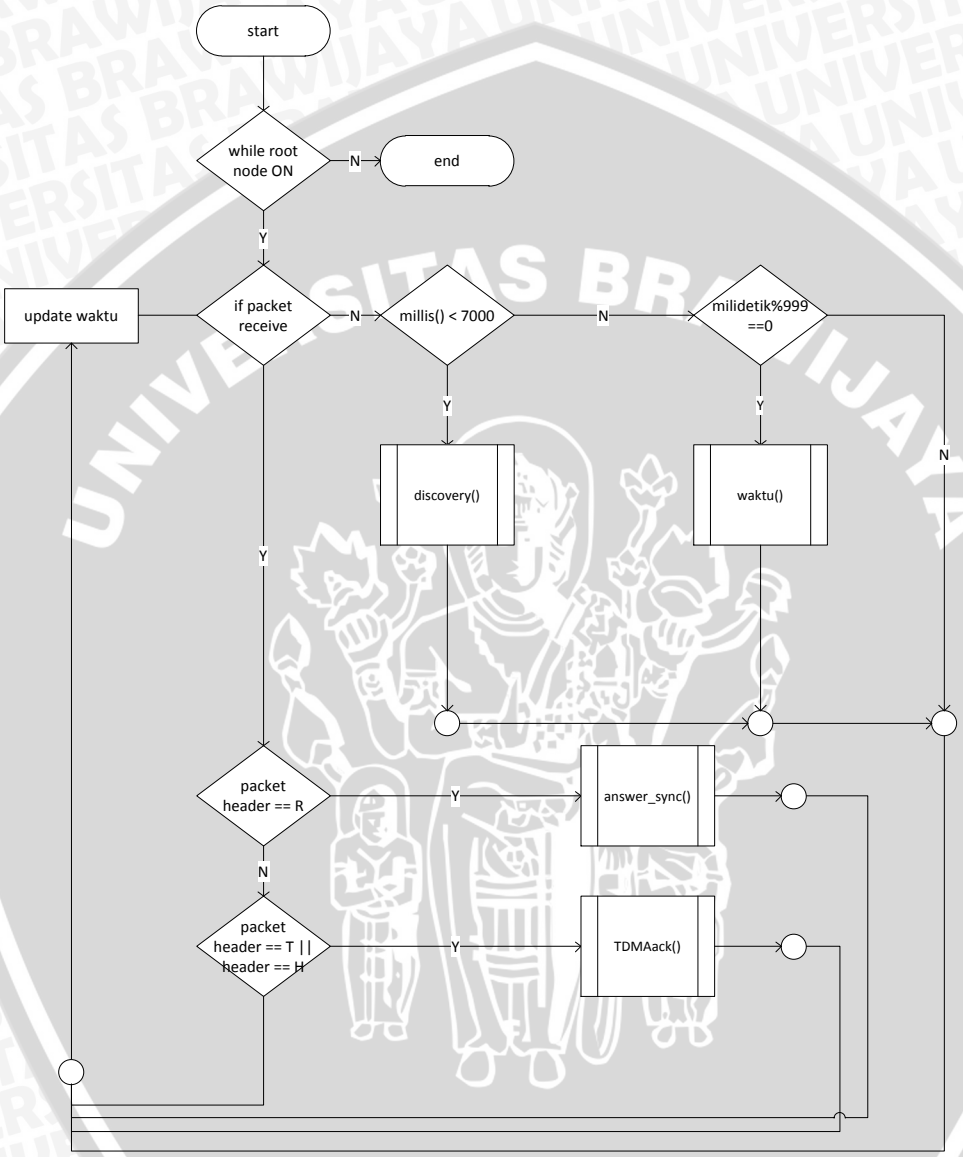


Gambar 5.8 Alur Pengiriman Pesan MQTT-SN *client*

Pengiriman pesan oleh MQTT-SN *client* menuju MQTT-SN *gateway* dimulai dengan melakukan setting alamat tujuan yaitu alamat *gateway*, 1111. Kemudian MQTT-SN *client* melakukan pengukuran suhu dan kelembaban dan melakukan *packaging* pesan sesuai dengan perancangan format pada Gambar 5.15. Luaran yang dihasilkan ketika MQTT-SN *client* berhasil mengirimkan pesan ditampilkan melalui serial monitor.

5.1.3.2 Perancangan MQTT-SN *Gateway*

Gateway merupakan perangkat yang berfungsi sebagai *receiver* juga sebagai *forwarder* pesan yang diterima dari MQTT-SN *client* menuju MQTT *publisher*. Untuk mengetahui alur kerja dari receiver *gateway*, berikut dijelaskan melalui diagram alir dibawah ini.

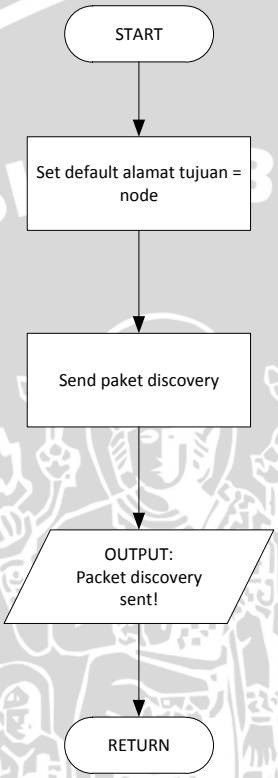


Gambar 5.9 Diagram Alir Utama MQTT-SN *gateway*

Alur kerja sistem MQTT-SN *gateway* yang secara *default* sebagai *root* yang mendudukin node pada level 0 pada sistem ini dimulai dari inisialisasi baudarate, pin sensor, channel, payload dan waktu tunggu. Inisialisasi baudarate dilakukan untuk menentukan mikrokontroler bekerja pada rate sesuai dengan yang sudah ditentukan. Inisialisasi pin sensor dilakukan untuk menentukan sensor dapat bekerja melalui pin yang juga telah ditentukan. Inisialisasi channel dilakukan untuk menentukan channel komunikasi *wireless* antara MQTT-SN *client* dengan MQTT-SN *gateway*. Inisialisasi payload dilakukan dengan

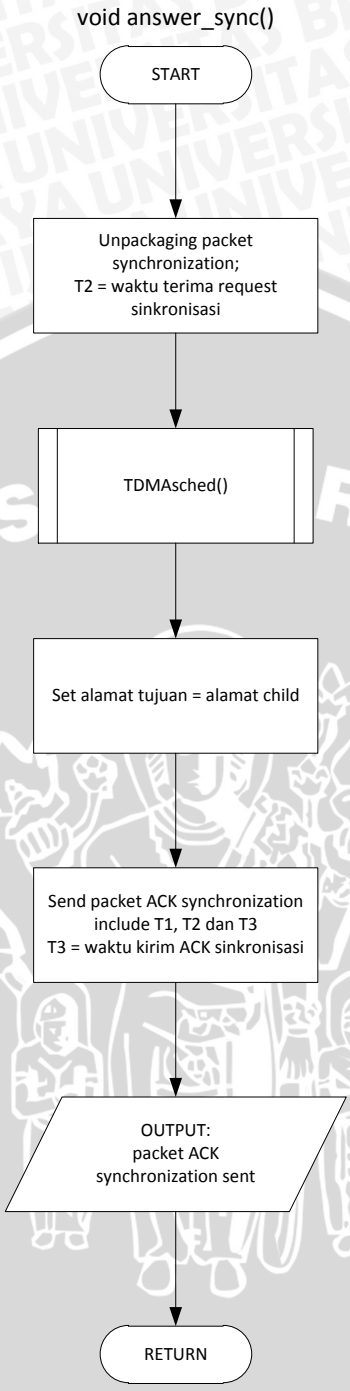
menentukan besar payload yang berisi protokol pesan MQTT-SN, TPSN dan TDMA.

Setelah MQTT-SN *gateway* aktif sebagai *root,* maka MQTT-SN *gateway* akan menjalankan fungsionalitas dari protokol TPSN yaitu fase *discovery* dan fase sinkronisasi. Fase *discovery* pada MQTT-SN *gateway* tidak jauh berbeda dengan MQTT-SN *client*. Namun pada perancangannya, MQTT-SN *gateway* yang menduduki level 0 tidak menerima paket *discovery* dari node manapun. Sehingga hanya ada satu *method* pada fase *discovery* yaitu *method discovery*().



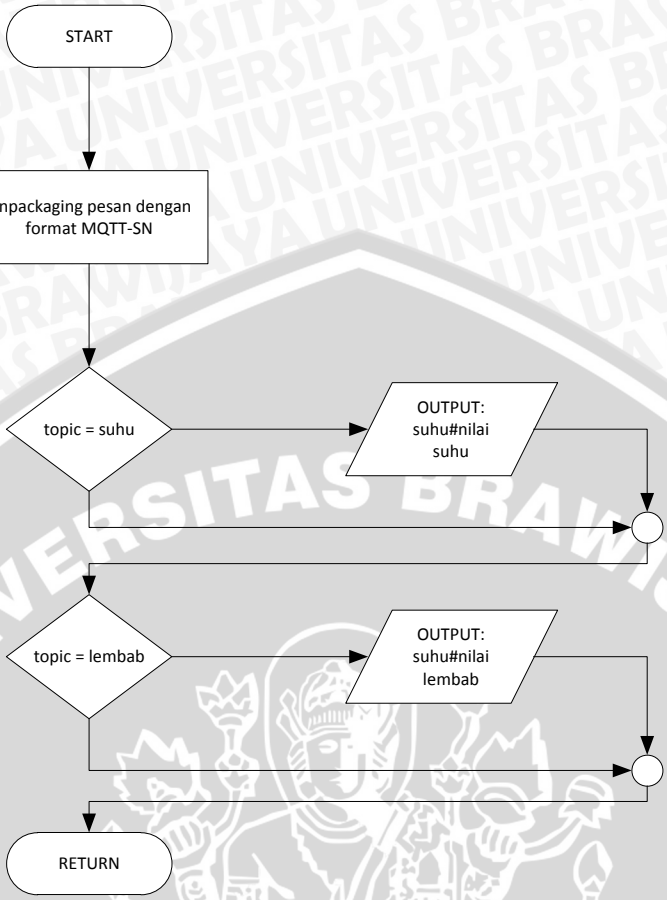
Gambar 5.10 Fase *Discovery* pada MQTT-SN *gateway*

Alur fase *discovery* MQTT-SN *gateway* ditunjukkan pada Gambar 5.10. Fase *discovery* berjalan selama 7 detik dengan melakukan broadcast menuju node yang aktif. Fase yang kedua adalah fase sinkronisasi. Fase sinkronisasi waktu pada *gateway* tidak sama dengan klien. Hal itu dikarenakan, *gateway* tidak mengirimkan permintaan sinkronisasi menuju node manapun karena kedudukannya sebagai root pada level 0. *Gateway* hanya menerima permintaan sinkronisasi dan mengirimkan paket ACK menuju klien yang melakukan permintaan. Alur fase sinkronisasi dijelaskan pada Gambar 5.11.



Gambar 5.11 Alur Fase Sinkronisasi MQTT-SN *gateway*

Penjelasan fase sinkronisasi protokol TPSN telah disampaikan pada subbab 2.2.6. Tugas *gateway* adalah mengirimkan pesan ACK *synchronization* yang berisi T1, T2 dan T3 beserta slot waktu penjadwalan pengiriman yang alurnya sama dengan MQTT-SN *client* terdapat pada Gambar 5.6. Fungsionalitas MQTT-SN *gateway* selanjutnya adalah menerima pesan yang berisi nilai dan *topic* suhu dan kelembaban. Alur penerimaan pesan ada pada Gambar 5.12.

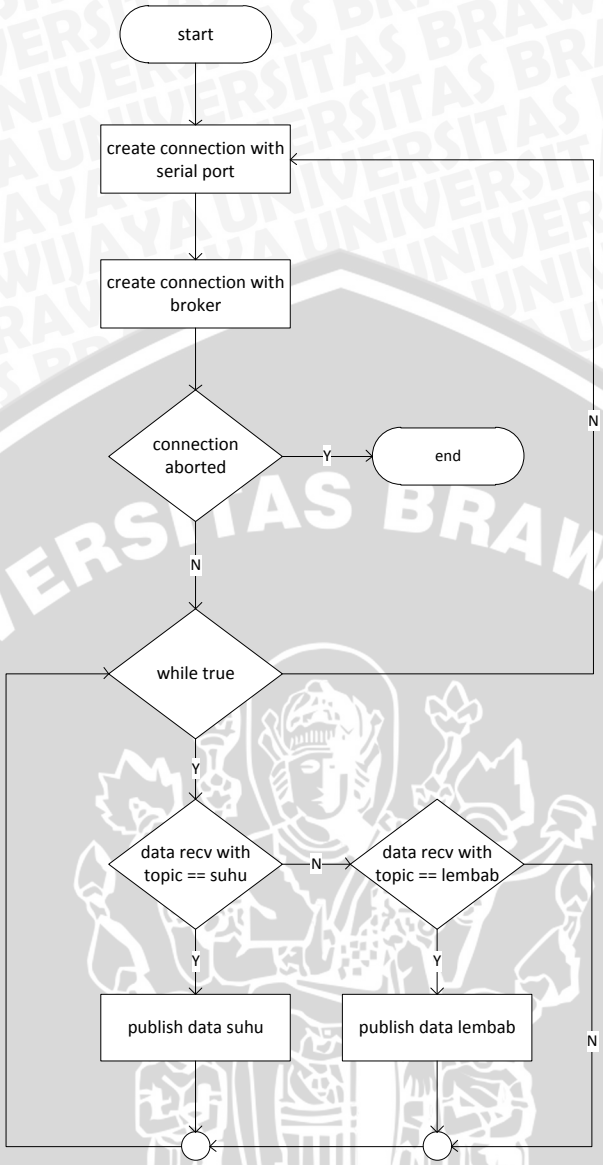


Gambar 5.12 Alur Penerimaan Pesan MQTT-SN *gateway*

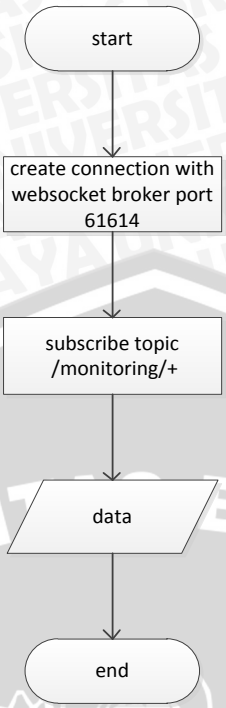
Alur penerimaan pesan dimulai dengan melakukan *unpackaging* pada pesan. Setalah itu didapatkan *topic* dari pesan tersebut apakah *topic* suhu ataukah lembab. Alur penerimaan diakhir dengan menampilkan *topic* beserta nilainya pada serial monitor.

5.1.3.3 Perancangan Simulasi MQTT

Perancangan algoritma simulasi MQTT mencakup alur kerja MQTT *publisher* dan MQTT *subscriber* beserta penjelasan komunikasinya dengan *broker*. Selain itu, dijelaskan pula algoritma komunikasi *broker* dengan MQTT *subscriber*. Dibawah ini merupakan alur kerja MQTT *publisher* dan MQTT *subscriber* dalam bentuk flowchart.



Gambar 5.13 Diagram Alir MQTT *publisher*



Gambar 5.14 Diagram Alir MQTT *subscriber*

MQTT *publisher* sebagai aplikasi yang berhubungan dengan perangkat MQTT-SN *gateway* membutuhkan inisialisasi port serial. Agar dapat terkoneksi dengan *broker*, MQTT *publisher* harus melakukan inisialisasi alamat jaringan lokal yaitu localhost dan port *broker* yaitu 1883. Selanjutnya setelah *broker* menerima pesan yang berisi *topic* dan payload dari MQTT *publisher*, kemudian *broker* akan menyalurkan data kepada MQTT *subscriber* sesuai dengan *topic*nya. Data yang diterima *subscriber* ditampilkan dalam bentuk grafik pada halaman web menggunakan port websocket *broker* yaitu port 61614.

5.1.4 Perancangan Format Pesan MQTT-SN

Pada sistem ini pesan menggunakan format protokol MQTT-SN. Pesan sebagai media komunikasi antara MQTT-SN *client* dengan MQTT-SN *gateway*. Pesan menggunakan *QoS* 0 sehingga pada proses komunikasinya tidak membutuhkan acknowledge.

Pesan berukuran 6 array string dengan susunan seperti pada Gambar 5.15, header pesan adalah *QOS* yaitu 0. Kemudian pada array ke 1 berisi identitas dari *topic* suhu dan kelembaban. Array ke 2 berisi identitas pesan dimulai dari angka 1 di*crement* sampai sejumlah pesan yang terkirim. Array ke 3 berisi type pesan, pada protokol MQTT-SN type pesan harus didefinisikan. Penelitian ini menggunakan type pesan *publish*. Array ke 4 adalah adalah bagian paling penting pada isi pesan yaitu, berisi nilai suhu atau kelembaban hasil sensing. Array yang terakhir berisi *topic* pesan dapat berupa suhu atau kelembaban.

QOS	Topic ID	Msg ID	Type Msg	Data	Msg Topic

Gambar 5.15 Format pesan MQTT-SN

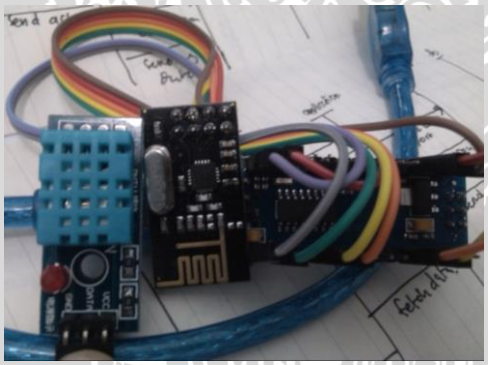
5.2 Implementasi Sistem

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

5.2.1 Implementasi Perangkat Keras

5.2.1.1 Implementasi MQTT-SN *Client*

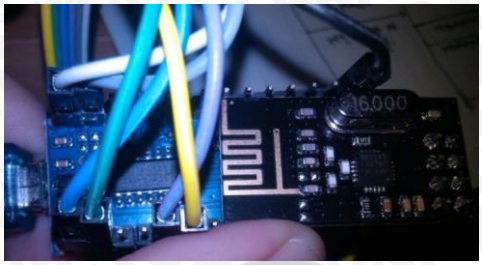
Sesuai dengan perancangan perangkat keras MQTT-SN *client*, implementasi MQTT-SN *client* terdiri dari input/sensor module yaitu sensor temperatur dan kelembaban DHT11, mikrokontroler jenis arduino nano v3.0 dan transmitter module. Pin-pin pada perangkat tersebut dihubungkan sesuai dengan rancangan pada Gambar 5.1. Sistem ini menggunakan modul komunikasi *wireless* jenis NRF24L01. Spesifikasi masing-masing perangkat telah disebutkan pada sub bab 5.1.1.1. Gambar 5.16 merupakan hasil implementasi dari perancangan perangkat keras MQTT-SN *client*.



Gambar 5.16 Perangkat Keras MQTT-SN *client*

5.2.1.2 Implementasi MQTT-SN *Gateway*

Sesuai dengan perancangan perangkat keras MQTT-SN *gateway*, implementasi MQTT-SN *gateway* terdiri dari mikrokontroler jenis arduino nano v3.0 dan receiver modul. Pin-pin pada perangkat tersebut dihubungkan sesuai dengan rancangan pada Gambar 5.2. Sistem ini menggunakan modul komunikasi *wireless* jenis NRF24L01. Spesifikasi masing-masing perangkat telah disebutkan pada sub bab 5.1.1.2. Gambar 5.17 merupakan hasil implementasi dari perancangan perangkat keras MQTT-SN *gateway*.



Gambar 5.17 Perangkat Keras MQTT-SN *gateway*

5.2.2 Implementasi Perangkat Lunak, Algoritma dan Format Pesan

5.2.2.1 Implementasi MQTT-SN *Client*

Agar dapat menjalankan fungsinya sebagai media transmisi yang mengirimkan data berupa nilai suhu dan kelembaban maka, harus ditanamkan program pada mikrokontroller arduino. MQTT-SN *client* membutuhkan *library* untuk menunjang fungsionalitas perangkat MQTT-SN *client* itu sendiri. *Library* akan di*include* ke dalam program yang akan ditanamkan ke perangkat. Implementasi kode program *include library* ada pada Gambar 5.18.

```
1  #include <DHT.h>
2  #include <SPI.h>
3  #include <Mirf.h>
4  #include <nRF24L01.h>
5  #include <MirfHardwareSpiDriver.h>
6  #include <stdio.h>
7  #include <string.h>
8  #include <stdlib.h>
```

Gambar 5.18 *Library* yang digunakan MQTT-SN *client*

Fungsionalitas lain yang harus dipenuhi oleh MQTT-SN adalah pengiriman pesan menggunkana protokol TDMA yang mengharuskan waktu antar klien harus sinkron terlebih dulu. Langkah selanjutnya pada proses implementasi perangkat lunak MQTT-SN *client* adalah inisialisasi variabel. Ada variabel yang digunakan untuk menjalankan protokol sinkronisasi waktu dan penjadwalan, juga ada variabel yang berrfungsi untuk mendukung format pesan protokol MQTT-SN. Gambar 5.19 menunjukkan variabel yang digunakan untuk mengimplementasikan protokol MQTT-SN. Format pesan yang berisikan data hasil sensing klien ada pada Gambar 5.20.

```
1  #define MQTTSN_TYPE_PUBLISH            0x0C
2  #define MQTTSN_TOPIC_TYPE_PREDEFINED   0b01
3  #define MQTTSN_FLAG_QOS_0              0b00
4  #define MQTTSN_RC_ACCEPTED             0x00
5  #define MQTTSN_TOPIC_ID_SUHU           1
6  #define MQTTSN_TOPIC_ID_LEMBAB         2
7  #define MQTTSN_PROTOCOL_ID             0x01
```

Gambar 5.19 Variabel yang digunakan MQTT-SN *client*

```
1  byte datasensorsend[6];
2  int t=0;
3  t+=1;
4  float lembab = dht.readHumidity();
5  float hum = dht.computeHeatIndex(lembab, false);
6  byte msg_typ = MQTTSN_TYPE_PUBLISH;
```

```
byte QoS = MQTTSN_FLAG_QOS_0;
byte topic_typ = MQTTSN_TOPIC_TYPE_PREDEFINED;
byte msg_id = t;
int topic_id = MQTTSN_TOPIC_ID_LEMBAB;
float data = hum;
datasensorsend[0] = QoS;
datasensorsend[1] = topic_id;
datasensorsend[2] = msg_id;
datasensorsend[3] = msg_typ;
datasensorsend[4] = data;
datasensorsend[5] = topic_typ;
```

Gambar 5.20 Format Pesan MQTT-SN

Metode penjadwalan TDMA mengharuskan masing-masing klien memiliki waktu yang setara, oleh karena itu dilakukan proses penyetaraan waktu menggunakan protokol TPSN. Implementasi protokol TPSN terbagi menjadi 2 fase yaitu, fase *discovery* dimana klien dan *gateway* membentuk sebuah topologi hirarki. *Gateway* bertindak sebagai *root* pada level 0 dan klien dapat bertindak sebagai *child* juga dapat bertindak sebagai *parent* bagi *child* pada level dibawahnya.

Fase kedua adalah, fase sinkronisasi dimana seluruh node pada topologi saling berkirim waktu untuk menyetarakan waktunya masing-masing. Pada fase sinkronisasi waktu mula-mula, *child* mengirimkan waktu lokalnya menuju *parent* pada level diatasnya sebagai permintaan sinkronisasi waktu. Kemudian *parent* membalas dengan mengirimkan waktu lokalnya. *Child* lalu memperbarui waktunya dengan cara menyetarakan waktunya dengan waktu *parent*nya.

Implementasi program fase *discovery* terbagi menjadi 2 *method* utama yaitu, *discovery*() dan *discovered*(). *Method discovery*() adalah *method* dengan berfungsi melakukan pengiriman paket *discovery* menuju node. Sedangkan *discovered*() adalah *method* dengan berfungsi melakukan penerimaan paket *discovery* dari *root*. Implementasi kode program ada pada Gambar 5.21.

```
void discovery(){
 sprintf(LevelKirim,"%d", Level);
 Mirf.setTADDR((byte *)"node");
 strcpy(DataKirim,"B");
 strcat(DataKirim,AlamatSendiri);
 strcat(DataKirim,LevelKirim);
 Mirf.send((byte *)&DataKirim);
 Serial.println("packet discovery sent to all node!");
  while(Mirf.isSending()){
  }
}

void discovered(){
  for (int d=0; d<4; d++){
  AlamatParent[d] = DataTerima[d+1];
  }
  Serial.print("packet discovery received from: ");
  Serial.println(AlamatParent);
  Leveltemp[0] = DataTerima[5];
  Level = strtoul(Leveltemp, NULL, 0);
  Level = Level+1;
  Serial.print("node joined on level: ");
  Serial.println(Level);

  UbahAlamat();
  Serial.print("address changed: ");
```

```
27    Serial.println(AlamatSendiri);
28    Mirf.setRADDR((byte *)AlamatSendiri);
29    terdaftar = 1;
30  }
```

Gambar 5.21 *Method discovery*() dan *discovered*() pada MQTT-SN client

Klien mengirimkan paket dengan header "B" menuju node yang belum menjadi *root*. Isi paket lainnya yaitu alamat node dan level node itu sendiri. Selanjutnya, klien merubah alamatnya setelah menerima paket discovered. Perubahan alamat terdapat pada *method UbahAlamat*() yang implementasi kodenya ada pada Gambar 5.22.

```
1  void UbahAlamat(){
2    int Acak = random(analogRead(A0),9999);
3    while(Acak<1000){
4    Acak = random(analogRead(A0),9999);
5    }
6    sprintf(AlamatSendiri,"%d", Acak);
7  }
```

Gambar 5.22 *Method UbahAlamat()*

Fase kedua yaitu fase sinkronisasi, implementasi fase sinkronisasi terbagi menjadi 2 *method* utama yaitu, *synchronize*() dan *synchronized*(). *Method synchronize*() adalah *method* dengan berfungsi melakukan permintaan sinkronisasi waktu menuju *root*. Sedangkan *synchronized*() adalah *method* dengan berfungsi melakukan penerimaan permintaan sinkronisasi waktu dari *root*. Implementasi kode program ada pada Gambar 5.23

```
1  void synchronize(){
2    waktu();
3    Mirf.setTADDR((byte *)Alamat Parent);
4    strcpy(DataKirim,"R");
5    strcat(DataKirim,AlamatSendiri);
6    sprintf(TimeStamp1,"%d",milidetik);
7    if(milidetik<10){
8      strcat(DataKirim,"0");
9      strcat(DataKirim,"0");
10   }
11   else if(milidetik<100){
12     strcat(DataKirim,"0");
13   }
14   strcat(DataKirim,TimeStamp1);
15   sprintf(LevelKirim,"%d", Level);
16   strcat(DataKirim,LevelKirim);
17   Latency = millis();
18   Mirf.send((byte *)&DataKirim);
19   while(Mirf.isSending()){
20     Serial.println("request synchronization sent!");
21     delay(2);
22   }
23 }
24
25 void synchronized(){
26    Serial.print("begin: ");
27    waktu();   cetakwaktu();
28    Serial.println("");
29    waktu();
30    Latency = millis() - Latency;
31    ubahwaktu();   waktu();   cetakwaktu();
32    Serial.println("synchronized!");
33    Serial.print("delay total: ");
34    Serial.println(Latency);
```

```
35      Serial.print("delay propagasi: ");
36      Serial.println(TS4);
37      if (TS4<20 && TS4>0){
38      tersinkron = 1;
39      }
40      Sched = jadwalpatent + 30;
41  }
```

Gambar 5.23 *Method synchronize*() dan *synchronized*() pada MQTT-SN client

Klien mengirimkan paket dengan header "R" sebagai tanda permintaan sinkronisasi menuju *root*. Isi paket sinkronisasi antara lain, alamat node waktu pengiriman dan level node. Klien merubah waktunya setelah menerima paket sinkronisasi. Perubahan alamat terdapat pada *method ubahwaktu*() yang implementasi kodenya ada pada Gambar 5.24.

```
1   void ubahwaktu(){
2     ubahmili = milibaru - milidetik;
3     ubahdetik = detikbaru - detiklama;
4     ubahmenit = menitbaru - menit;
5     ubahjam = jambaru - jam;
6     ubahmili = ubahmili + TS4;
7   }
```

Gambar 5.24 *method ubahwaktu*()

MQTT-SN *client* dapat berfungsi sebagai *child* maupun parent sehingga pada proses sinkronisasi seperti yang dijelaskan pada subbab 2.2.6 MQTT-SN client dapat mengirimkan pesan ACK synchronization menuju child yang berada pada level dibawahnya.

```
1   void answer_sync(){
2     TDMAsched();
3     for (int x=0; x<4; x++){
4       AlamatTerima[x] = DataTerima[x+1];
5     }
6       for (int x=0; x<3; x++){
7       TimeStamp1[x] = DataTerima[x+5];
8     }
9     Serial.print("request synchronization from: ");
10    Serial.println(AlamatTerima);
11    Mirf.setTADDR((byte *)AlamatTerima);
12    kirimwaktu();
13    strcat(DataKirim,TimeStamp1);
14    if(TimeStampInt<10){
15      strcat(DataKirim,"0");
16      strcat(DataKirim,"0");
17    }
18    else if(TimeStampInt<100){
19      strcat(DataKirim,"0");
20    }
21    strcat(DataKirim,TimeStamp2);
22    sprintf(Sched2,"%d", Sched);
23    strcat(DataKirim,Sched2);
24    Mirf.send((byte *)&DataKirim);
25    akb = millis(); akb = akb-akk;
26    while(Mirf.isSending()){
27      Serial.println("synchronization sent!");
28    }
29  }
```

Gambar 5.25 Method answer_sync() pada MQTT-SN client

Implementasi kode program pengiriman pesan ACK synchronization ada pada Gambar 5.25. Pesan ACK synchronization berisi T1, T2, T3 dan pembagian slot waktu jadwal pengiriman pesan menuju gateway. Kemudian dijalankan method synchronized() pada node penerima paket ACK synchronization dan melakukan *update* waktu node yang melakukan permintaan sinkronisasi. Jika waktu antar klien masing-masing sudah setara, maka selanjutnya dilakukan pengiriman data sesuai slot waktu yang ditentukan. Penjadwalan sesuai slot waktu berguna agar masing-masing klien mengirim pada slot waktu yang berbeda sehingga tidak terjadi tabrakan. Implementasi pembagian slot waktu ada pada *method TDMAsched*().

```
1  void TDMAsched(){
2    Sched = Sched + 10;
3  }
```

Gambar 5.26 *method TDMAsched()* pada klien

Implementasi yang terakhir adalam fungsi pengiriman data. Pengiriman data dilakukan jika waktu antar node telah setara. Pembagian slot waktu dilakukan secara acak kepada klien, sehingga tidak dapat dipastikan klien dapat melakukan pengiriman pada slot ke berapa. Implementasi kode program pengiriman data suhu ada pada *method TDMAtempsend*() sedangkan data lembab ada pada *method TDMAhumsend*(). Sampel kode program pengiriman data sensor ada pada Gambar 5.27. Gambar 5.27 menunjukkan pengiriman data suhu sehingga dilakukan pengukuran data suhu terlebih dahulu, *topic* pesan juga disesuaikan yaitu, suhu. Sedangkan pengiriman data kelembaban, dilakukan pengukuran data kelembaban sesuai dengan fungsinya. *Topic* pesannya pun juga menyesuaikan yaitu, lembab.

```
1  void TDMAtempsend(){
2    Mirf.setTADDR((byte *)"1111");
3    byte datasensorsend[6];
4    t+=1;
5    float suhu = dht.readTemperature();
6    float temp = dht.computeHeatIndex(suhu, false);
7    byte msg_typ = MQTTSN_TYPE_PUBLISH;
8    byte QoS = MQTTSN_FLAG_QOS_0;
9    byte topic_typ = MQTTSN_TOPIC_TYPE_PREDEFINED;
10   byte msg_id = t;
11   int topic_id = MQTTSN_TOPIC_ID_SUHU;
12   float data = temp;
13   datasensorsend[0] = QoS;
14   datasensorsend[1] = topic_id;
15   datasensorsend[2] = msg_id;
16   datasensorsend[3] = msg_typ;
17   datasensorsend[4] = data;
18   datasensorsend[5] = topic_typ;
19   Mirf.setTADDR((byte *)"1111");
20   Mirf.send((byte *)&datasensorsend);
21   while(Mirf.isSending());
22   Serial.println("data temperature sent!");
23 }
```

Gambar 5.27 Kode program pengiriman data sensor

5.2.2.2 Implementasi MQTT-SN *Gateway*

MQTT-SN *gateway* dalam implementasi metode penjadwalan TDMA dan protokol sinkronisasi waktu TPSN berperan sebagai *root* yang menduduki level 0. Sama halnya dengan MQTT-SN *client*, *gateway* juga menerapkan 2 fase pada proses sinkronisasi waktu. Pada fase *discovery*, fungsi *root* adalah mengirimkan paket *discovery* selama 7 detik. Namun *root* tidak dapat menerima paket *discovery* karena terletak pada level 0. Implementasi kode program pada Gambar 5.28.

```
1  void discovery(){
2    Mirf.setTADDR((byte *)"node");
3    strcpy(DataKirim,"B");
4    strcat(DataKirim,AlamatSendiri);
5    strcat(DataKirim,Level);
6    Mirf.send((byte *)&DataKirim);
7    while(Mirf.isSending()){
8      Serial.println("packet discovery sent to all node!");
9      delay(2);
10   }
11 }
```

Gambar 5.28 *Method discovery()* pada *gateway*

Root mengirimkan paket dengan header "B" menuju node yang belum menjadi *root*. Isi paket lainnya yaitu alamat *root* dan level *root* itu sendiri. Setelah mengirimkan paket *discovery*, *root* akan menunggu sampai terdapat node yang mengirimkan permintaan sinkronisasi waktu. *Root* menerima paket sinkronisasi waktu, kemudian mengirimkan balasan yang berisi waktu *root* saat itu dan jadwal pengiriman data. Impelementasi kode program pada *method synchronize*().

```
1  void answer_sync(){
2    TDMAsched();
3    for (int x=0; x<4; x++){
4      AlamatTerima[x] = DataTerima[x+1];
5    }
6      for (int x=0; x<3; x++){
7      TimeStamp1[x] = DataTerima[x+5];
8    }
9    Serial.print("request synchronization from: ");
10   Serial.println(AlamatTerima);
11   Mirf.setTADDR((byte *)AlamatTerima);
12   kirimwaktu();
13   strcat(DataKirim,TimeStamp1);
14   if(TimeStampInt<10){
15     strcat(DataKirim,"0");
16     strcat(DataKirim,"0");
17   }
18   else if(TimeStampInt<100){
19     strcat(DataKirim,"0");
20   }
21   strcat(DataKirim,TimeStamp2);
22   sprintf(Sched2,"%d", Sched);
23   strcat(DataKirim,Sched2);
24   Mirf.send((byte *)&DataKirim);
25   akb = millis(); akb = akb-akk;
26   while(Mirf.isSending()){
27     Serial.println("synchronization sent!");
```

```
28    }
29  }
```

Gambar 5.29 *Method synchronize()* pada *gateway*

Root mengirimkan waktu sebagai parameter penyetaraan waktu lokal dengan klien. Waktu dikirim lewat balasan permintaan sinkronisasi dari klien beserta jadwal pengiriman data sensor. Setelah klien berhasil menerima paket balasan permintaan sinkronisasi dari *root*, maka *root* menunggu sampai data sensor diterima. Data yang diterima kemudian ditampilkan melalui serial monitor sesuai dengan *topic*nya. Implementasi kode program penerimaan data oleh *gateway* terdapat pada *method TDMAack*()

```
1   void TDMAack(){
2    Mirf.getData((byte*)&datasensorrecv);
3    byte msg_typ = MQTTSN_TYPE_PUBACK;
4    byte topic_id = MQTTSN_TOPIC_ID_SUHU;
5    byte returncode = MQTTSN_RC_ACCEPTED;
6    if (datasensorrecv[3] == MQTTSN_TYPE_PUBLISH){
7     if (datasensorrecv[0] == MQTTSN_FLAG_QOS_0){
8      if (datasensorrecv[5]== MQTTSN_TOPIC_TYPE_PREDEFINED){
9       if (datasensorrecv[1] == MQTTSN_TOPIC_ID_SUHU){
10       Serial.print("suhu");
11       Serial.print("#");
12       Serial.println(datasensorrecv[4]);
13      }
14      else if (datasensorrecv[1]==MQTTSN_TOPIC_ID_LEMBAB){
15       Serial.print("lembab");
16       Serial.print("#");
17       Serial.println(datasensorrecv[4]);
18      }
19     }
20    }
21   }
22  }
```

Gambar 5.30 *Method TDMAack*()

5.2.2.3 Implementasi Simulasi MQTT *Publisher*

Implementasi MQTT *publisher* dimulai dengan inisialisasi *library* yang akan digunakan pada program aplikasi MQTT *publisher*. Aplikasi program MQTT *publisher* menggunakan bahasa python pada implementasinya. MQTT *publisher* membutuhkan aplikasi *broker* dalam keadaan ready untuk menjalankan fungsinya. MQTT *publisher* berjalan pada port 1883 sebagai *client* dari *broker* yang berjalan di jaringan lokal dengan alamat "*localhost*". Gambar 5.31 merupakan inisialisasi beberapa *library* yang digunakan pada program MQTT *publisher*.

```
1  import sys
2  import paho.mqtt.client as publish
3  import serial
```

Gambar 5.31 *Library* MQTT *publisher*

Dalam menjalankan fungsinya sebagai receiver dari *gateway*, MQTT *publisher* harus terhubung dengan *gateway* melalui komunikasi serial. MQTT *publisher* juga berperan sebagai *client* dari *broker* pada simulasi ini, sehingga dibutuhkan koneksi dengan *broker*. Implementasi kode program koneksi MQTT

publisher dengan *Gateway* dan koneksi MQTT *publisher* dengan *broker* akan digambarkan pada Gambar 5.32 dibawah ini.

```
mqttc = publish.Client("monitoring", clean_session=False)
mqttc.connect("localhost", 1883)
ser = serial.Serial('COM4', 9600)
```

Gambar 5.32 Deklarasi variabel komunikasi MQTT *publisher*

Implementasi selanjutnya dilakukan dengan membaca data yang dikirim oleh *gateway* secara serial, kemudian dilakukan pemisahan *topic* dengan data sensor. Setelah data berhasil dipisah masing-masing pada variabel yang berbeda, dilakukan proses *publish* sesuai dengan *topic*nya. Kode pada Gambar 5.33 merupakan implementasi dari program utama aplikasi MQTT *publisher*.

```
while True:
a = ser.readline()
print a
if '#' in a:
 b = a.split('#')
 c = b[0]
 d = b[1]
 if c == "suhu":
  mqttc.publish("monitoring"+"/lembab", payload=d)
 elif c == "lembab":
  mqttc.publish("monitoring"+"/suhu", payload=d)

mqttc.loop_forever()
```

Gambar 5.33 Program utama MQTT *publisher*

5.2.2.4 Implementasi Simulasi MQTT *Subscriber*

Dalam menjalankan fungsinya, MQTT *subscriber* berkomunikasi dengan *broker*. Komunikasi dengan *broker* dilakukan dengan cara melakukan pemanggilan *broker* dan websocket yang telah disediakan *broker*. Pemrograman aplikasi *subscriber* menggunakan bahasa pemrograman HTML dan Javascript. Dibawah ini merupakan kode program implementasi dari pemanggilan *broker*, port websocket dan *topic* pesan yang di*subscriber*.

```
var MQTTbroker = 'localhost';
var MQTTport = 61614;
var MQTTsubTopic = 'monitoring/+';
```

Gambar 5.34 Inisialisasi kebutuhan *subscriber*

Setelah inisialisasi variabel yang berfungsi melakukan pemanggilan *broker* dan port websocket, selanjutnya adalah pembentukan klien. Pembentukan klien dilakukan dengan melakukan inisialisasi variabel klien dengan melakukan pemanggilan fungsi Paho seperti *publisher.* Implementasi variabel klien terdapat pada Gambar 5.35.

```
var client = new Paho.MQTT.Client(MQTTbroker, MQTTport,
      "myclientid_" + parseInt(Math.random() * 100, 10));
```

Gambar 5.35 Inisialisasi klien *subscriber*

Setelah klien terbentuk dan telah dapat dihubungkan dengan *broker* serta websocket, maka klien telah dapat melakukan fungsinya yaitu *subscribe* pesan sesuai path *topic* yang ada pada Gambar 5.34.

BAB 6 PENGUJIAN DAN ANALISIS

Bagian ini merupakan pembahasan hasil dan analisis pengujian dari penelitian yang telah dilakukan.

6.1 Pengujian dan Analisis MQTT-SN *Client*

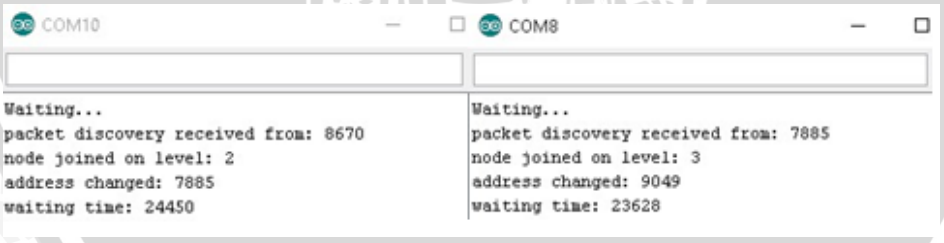
Pengujian dilakukan dengan mengaktifkan 3 node sebagai klien yang terhubung melalu jaringan nirkabel dengan satu node *gateway*. Adapun kebutuhan fungsionalitas yang harus dipenuhi oleh MQTT-SN *client* dijelaskan melalui sub bab 6.1.1 dan 6.1.2 dibawah ini beserta prosedur pengujiannya.

6.1.1 Protokol TPSN pada MQTT-SN *Client*

Protokol TPSN merupakan jenis protokol yang digunakan untuk melakukan penyetaraan waktu antar node pada penelitian ini. Pada proses penyetaraan waktu, TPSN melalui 2 fase yaitu fase *discovery* dan fase sinkronisasi.

6.1.1.1 Fase *Discovery*

Pengujian fase *discovery* bertujuan untuk mengamati keberhasilan setiap MQTT-SN *client* yang aktif dapat membentuk sebuah topologi hirarki yang tersusun pada level 1-3. Mulanya setiap klien diberi alamat "node" kemudian alamat tersebut berubah setelah node berhasil menduduki level pada topologi hirarki. Perubahan alamat perlu dilakukan agar data dari MQTT-SN *client* dapat dikirim secara individu menuju MQTT-SN *gateway*. Pemberian alamat baru dilakukan secara acak yang terdiri dari 4 karakter (bilangan bulat). Prosedur pengujian untuk fungsionalitas fase *discovery* dijelaskan pada Tabel 3.1. Gambar 6.1 dibawah ini merupakan tampilan pada saat proses *discovery*.



Gambar 6.1 Tampilan Fase *Discovery*

Serial monitor menampilkan asal paket *discovery*, juga level node dan perubahan alamat node. Pada kolom status keberhasilan terdapat tanda (v) yang menyatakan percobaan fase discovery berhasil dilakukan. Parameter keberhasilan fase discovery diukur dari keberhasilan suatu node membentuk topologi hirarki ditandai dengan level yang didapatkan dan perubahan alamat. Pengujian dilakukan selama 5 kali percobaan menghasilkan luaran pada Tabel 6.1 dibawah ini.

Tabel 6.1 Hasil Pengujian Fase *Discovery*

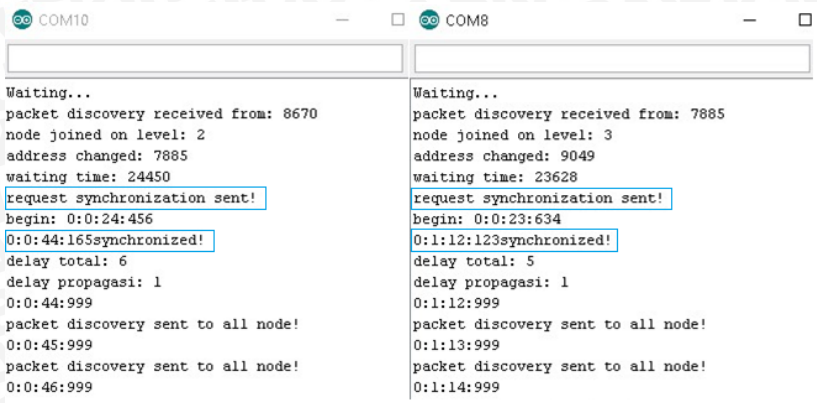
Percobaan Ke-	Level Node	Alamat Lama	Alamat Baru	Alamat *Parent*	Status Keberhasilan
1	1	node	1962	1111	v
	2	node	7976	1962	v
	3	node	1356	7976	v
2	1	node	8084	1111	v
	2	node	5674	8084	v
	3	node	6980	5674	v
3	1	node	1479	1111	v
	2	node	7433	1479	v
	3	node	8558	7433	v
4	1	node	8093	1111	v
	2	node	5472	8093	v
	3	node	9605	5472	v
5	1	node	1881	1111	v
	2	node	9644	1881	v
	3	node	6332	9644	v

Dari hasil pengujian yang dilakukan sebanyak 5 kali percobaan dengan prosentase keberhasilan 100% maka dapat disimpulkan, bahwa fase *discovery* pada MQTT-SN *client* dapat membentuk topologi hirarki terhitung dari level 0 yang diduduki oleh *gateway* dan level 1 sampe 3 diduduki oleh klien. Masing-masing klien dapat menerima paket *discovery* dari *root* dan mendapat level-level melalui proses aktivasi klien secara bergantian.

6.1.1.2 Fase Sinkronisasi

Pengujian fase sinkronisasi bertujuan untuk mengamati keberhasilan setiap MQTT-SN *client* yang aktif dapat melakukan penyetaraan waktu antar level yang berbeda sesuai dengan ketentuan protokol TPSN. Mulanya setiap klien dengan alamat baru dan telah menduduki level pada topologi hirarki akan melakukan *request* sinkronisasi kepada node *parent*. *Parent* atau *root* menerima dan membalas permintaan sinkronisasi dengan mengirimkan waktu *parent* yang sedang berjalan. Pengujian ini juga mencatat durasi waktu tunggu klien yang berbeda dengan klien lainnya. Pengaturan waktu tunggu yang berbeda dilakukan secara acak dan kurang dari 30 detik. Waktu tunggu yang acak dan berbeda antar klien berfungsi menghindari interferensi permintaan sinkronisasi. Prosedur pengujian untuk fungsionalitas fase *discovery* dijelaskan pada Tabel 3.2.

Gambar 6.2 dibawah ini merupakan tampilan pada saat proses *discovery*. Tanda berwarna biru menunjukkan klien melakukan permintaan sinkronisasi dan berhasil mendapat jawaban permintaan dari *parent*.



Gambar 6.2 Sinkronisasi waktu pada MQTT-SN *client*

Pada serial monitor menampilkan waktu tunggu node antara proses *discovery* sampai pengiriman permintaan sinkronisasi berbeda-beda. Hal tersebut disebabkan proses *discovery* dan pengiriman permintaan sinkronisasi antar klien yang juga berbeda. Pengujian dilakukan selama 5 kali percobaan didapatkan hasil seperti pada Tabel 6.2.

Kolom waktu tunggu berisi nilai rentang waktu node saat melakukan discovery sampai node dapat melakukan permintaan sinkronisasi. Kolom Tersinkron pada berisi waktu node saat dinyatakan telah tersinkron. Keberhasilan permintaan berisi status keberhasilan node melakukan permintaan sinkronisasi sedangkan keberhasilan respon berisi status keberhasilan node menerima ACK sinkronisasi.

Tabel 6.2 Hasil Pengujian Fase Sinkronisasi

Percobaan Ke-	Level Node	Waktu Tunggu (sekon)	Tersinkron Pada (MM:SS)	Keberhasilan Permintaan	Keberhasilan Respon
1	1	22,119	0:26	v	v
	2	16,817	0:48	v	v
	3	23,369	1:14	v	v
2	1	20,174	0:18	v	v
	2	29,008	0:29	v	v
	3	20,209	1:11	v	v
3	1	21,785	0:24	v	v
	2	16,817	0:27	v	v
	3	23,289	0:53	v	v
4	1	23,736	0:26	v	v
	2	22,709	0:34	v	v
	3	16,893	0:53	v	v
5	1	16,168	0:16	v	v
	2	16,848	0:28	v	v
	3	15,562	2:13	v	v

Dari hasil pengujian yang dilakukan sebanyak 5 kali percobaan dengan prosentase keberhasilan 100% maka dapat disimpulkan, proses sinkronisasi *child* dengan *parent* yang berada satu level diatasnya berjalan lancar dengan rata-rata waktu tunggu kurang dari 30 detik. Salah satu kelemahan dari protokol TPSN adalah kemungkinan terjadinya interferensi data pada proses permintaan sinkronisasi akibat waktu tunggu acak yang mendekati waktu permintaan node lainnya. Kemungkinan interferensi bisa diperkecil dengan menambah rentang waktu tunggu acak yang dihasilkan, namun dapat berakibat fase sinkronisasi yang berjalan lebih lama.

6.1.2 Protokol TDMA pada MQTT-SN *Client*

Protokol TDMA pada penelitian ini merupakan protokol yang digunakan untuk melakukan pengiriman data berdasarkan slot waktu yang telah ditentukan secara random. Pada pengiriman data menggunakan protokol TDMA, data atau pesan yang dikirim menggunakan format MQTT-SN dengan nilai *QoS* 0. Sehingga data dikirim tanpa adanya permintaan dan juga tanpa adanya pesan acknowledge.

6.1.2.1 Pengiriman Pesan dengan *Topic* yang sama

Pada penelitian ini pengiriman pesan menggunakan metode TDMA dengan pembagian rentang waktu 10 detik. Pengiriman dimulai dari menit pertama. Jadwal pengiriman pesan dikirim menuju klien melalui paket jawaban permintaan sinkronisasi. Pengujian fungsionalitas TDMA dengan *topic* yang sama bertujuan untuk membuktikan keberhasilan setiap klien dapat mengirimkan pesan dengan format MQTT-SN dan pada rentang waktu yang telah ditentukan secara acak. Untuk lebih jelasnya, prosedur pengujian diatur pada Tabel 3.3.

Tabel 6.3 merupakan tabel hasil pengujian pengiriman pesan dengan *topic* yang sama. Kolom time slot berisi pada detik ke berapa klien mengirimkan pesan. Kolom waktu kirim berisi waktu pengiriman data oleh klien. Kolom *Topic* berisi *topic* pesan yang dikirimkan oleh klien. Kolom status pengiriman berisi tanda (v) yang menandakan pengiriman pesan berhasil dilakukan oleh klien sedangkan tanda (-) menandakan tidak ada data yang dikirim. Kolom status penerimaan berisi tanda (v) yang menandakan penerimaan pesan berhasil dilakukan oleh gateway sedangkan tanda (-) menandakan tidak ada data yang diterima oleh node.

Tabel 6.3 Hasil Pengujian Pengiriman Pesan dengan *Topic* yang sama

Percobaan Ke-	Level Node	Time Slot	Waktu Kirim (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
1	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	Suhu & Lembab	v	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	Suhu & Lembab	v	v
	3	-	-	-	-	-

Percobaan Ke-	Level Node	Time Slot	Waktu Kirim (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
	1	10	3:10	Suhu & Lembab	v	v
	2	50	3:50	Suhu & Lembab	v	v
	3	-	-	-	-	-
2	1	10	1:10	temperature & humdity	v	v
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	2:10	temperature & humdity	v	v
	2	50	2:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	50	3:50	temperature & humdity	v	v
	3	10	3:10	temperature & humdity	v	v
	1	-	-	-	-	-
	2	50	4:50	temperature & humdity	v	v
	3	10	4:10	temperature & humdity	v	v
3	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	Suhu & Lembab	v	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	Suhu & Lembab	v	v
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	v	v
	2	50	3:50	Suhu & Lembab	v	v
	3	-	-	-	-	-
	1	10	4:10	Suhu & Lembab	v	v
	2	50	4:50	Suhu & Lembab	v	v
	3	-	-	-	-	-
4	1	10	1:10	Suhu & Lembab	v	V
	2	50	1:50	Suhu & Lembab	v	V
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	V	v
	2	50	2:50	Suhu & Lembab	V	v
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	V	V

Percobaan Ke-	Level Node	Time Slot	Waktu Kirim (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
	2	-	-	-	-	-
	3	-	-	-	-	-
	1	10	4:10	Suhu & Lembab	V	V
	2	-	-	-	-	-
	3	-	-	-	V	v
5	1	10	1:10	Suhu & Lembab	V	v
	2	50	1:50	Suhu & Lembab	V	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	V
	2	50	2:50	Suhu & Lembab	v	V
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	V	V
	2	-	-	-	-	-
	3	-	-	-	-	-

Dari hasil pengujian yang dilakukan sebanyak 5 kali percobaan dengan prosentase keberhasilan pengiriman pesan mencapai 100% dengan rincian keberhasilan pengiriman 44 dari total pengiriman 44. Sedangkan sebanyak 10 percobaan klien tidak dapat mengirim pesan disebabkan karena tidak mendapat slot waktu. Dari rincian tersebut dapat disimpulkan, protokol sinkronisasi waktu TPSN dapat menunjang pengiriman pesan menggunakan metode penjadwalan TDMA. Pengiriman paket oleh klien menggunakan format pesan MQTT-SN dan *topic* yang sama antar masing-masing klien dapat dikirim sesuai slot waktu menuju *gateway*. Pengiriman pesan dilakukan diatas menit pertama dengan pembagian slot pengiriman 10 detik dan waktu *idle* sebesar 40 detik.

6.1.2.2 Pengiriman Pesan dengan *Topic* yang berbeda

Pada penelitian ini, pengiriman pesan menggunakan metode TDMA dengan pembagian slot waktu 10 detik. Pengiriman dimulai dari menit pertama detik ke 10. Jadwal pengiriman data dikirim menuju klien melalui paket balasan permintaan sinkronisasi. Pengujian fungsionalitas TDMA dengan *topic* yang berbeda bertujuan untuk membuktikan keberhasilan setiap klien dapat mengirimkan pesan dengan format MQTT-SN dan pada slot waktu yang telah ditentukan. Prosedur pengujian telah diatur pada Tabel 3.4.

Tabel 6.4 Hasil Pengujian Pengiriman Pesan dengan 2 *Topic* berbeda

Percobaan Ke-	Level Node	Time Slot	Waktu (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
1	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	temperature & humdity	v	v

Percobaan Ke-	Level Node	Time Slot	Waktu (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	50	3:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	50	4:50	temperature & humdity	v	v
	3	-	-	-	-	-
2	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	v	v
	2	50	3:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	4:10	Suhu & Lembab	v	v
	2	50	4:50	temperature & humdity	v	v
	3	-	-	-	-	-
3	1	-	-	-	-	-
	2	-	-	-	-	-
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	50	3:50	temperature & humdity	v	v

Percobaan Ke-	Level Node	Time Slot	Waktu (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	50	4:50	temperature & humdity	v	v
	3	-	-	-	-	-
4	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	v	v
	2	50	3:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	4:10	Suhu & Lembab	v	v
	2	50	4:50	temperature & humdity	v	v
	3	-	-	-	-	-
5	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	-	-	-	-	-
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	v	v
	2	-	-	-	-	-
	3	-	-	-	-	-
	1	10	4:10	Suhu & Lembab	v	v
	2	-	-	-	-	-
	3	-	-	-	-	-

Tabel 6.5 Hasil Pengujian Pengiriman Pesan dengan 3 *Topic* berbeda

Percobaan Ke-	Level Node	Time Slot	Waktu (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
1	1	10	1:10	Suhu & Lembab	v	v
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	2:10	Suhu & Lembab	v	v
	2	50	2:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	3:10	Suhu & Lembab	v	v
	2	50	3:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	4:10	Suhu & Lembab	v	v
	2	50	4:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	5:10	Suhu & Lembab	v	v
	2	50	5:50	temperature & humdity	v	v
	3	-	-	-	-	-
2	1	10	1:10	temp & humi	v	v
	2	50	1:50	suhu & lembab	v	v
	3	-	-	-	-	-
	1	10	2:10	temp & humi	v	v
	2	50	2:50	suhu & lembab	v	v
	3	-	-	-	-	-
	1	10	3:10	temp & humi	v	v
	2	50	3:50	suhu & lembab	v	v
	3	-	-	-	-	-
	1	10	4:10	temp & humi	v	v
	2	50	4:50	suhu & lembab	v	v
	3	-	-	-	-	-
	1	10	5:10	temp & humi	v	v
	2	50	5:50	suhu & lembab	v	v
	3	-	-	-	-	-
3	1	10	1:10	temp & humi	v	v
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	-	-	-	-	-

Percobaan Ke-	Level Node	Time Slot	Waktu (MM:SS)	*Topic*	Status Pengiriman	Status Penerimaan
	3	10	2:10	Suhu & Lembab	v	v
	1	-	-	-	-	-
	2	-	-	-	-	-
	3	10	3:10	Suhu & Lembab	v	v
	1	-	-	-	-	-
	2	-	-	-	-	-
	3	10	4:10	Suhu & Lembab	v	v
	1	-	-	-	-	-
	2	-	-	-	-	-
	3	10	5:10	Suhu & Lembab	v	v
4	1	-	-	-	-	-
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	50	2:50	temperature & humdity	v	v
	3	10	2:10	Suhu & Lembab	v	v
	1	10	3:10	temp & humi	v	v
	2	50	3:50	temperature & humdity	v	v
	3	-	-	-	-	-
	1	10	4:10	temp & humi	v	v
	2	-	-	-	-	-
	3	-	-	-	-	-
	1	-	-	-	-	-
	2	-	-	-	-	-
	3	10	5:10	Suhu & Lembab	v	v
5	1	-	-	-	-	-
	2	50	1:50	temperature & humdity	v	v
	3	-	-	-	-	-

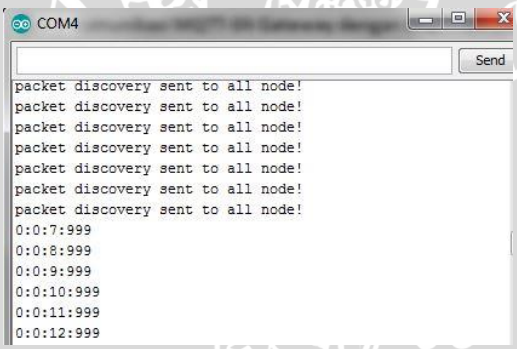
Dari hasil pengujian yang dilakukan sebanyak 5 kali percobaan dengan 2 *topic* berbeda menghasilkan prosentase keberhasilan sebesar 100% dengan rincian pengiriman pesan berhasil sebanyak 33 dari total pengiriman 33. Sedangkan sebanyak 27 percobaan klien tidak melakukan pengiriman pesan disababkan karena tidak mendapatkan slot pengiriman. Sedangkan percobaan dengan jumlah yang sama dengan 3 *topic* berbeda menghasilkan prosentase keberhasilan sebesar 100% dengan rincian pengiriman pesan berhasil sebanyak 34 dari total pengiriman 34. Sedangkan sebanyak 29 percobaan klien tidak melakukan pengiriman pesan disebabkan karena tidak mendapat slot pengiriman.

Dari rincian tersebut maka dapat disimpulkan, protokol sinkronisasi waktu TPSN dapat menunjang pengiriman pesan menggunakan metode penjadwalan TDMA. Pengiriman paket oleh klien menggunakan format pesan MQTT-SN dan *topic* yang berbeda antar masing-masing klien dapat dikirim sesuai slot waktu menuju *gateway*. Pengiriman pesan dilakukan diatas menit pertama dengan pembagian slot pengiriman 10 detik dan waktu *idle* sebesar 40 detik. Kemungkinan kegagalan pengiriman pesan dapat disebabkan oleh tidak ada/habisnya slot pengiriman atau slot transmit dalam satu menit.

Analisis penyebab klien tidak melakukan pengiriman pesan menuju gateway baik pada *topic* yang sama maupun *topic* yang berbeda dikarenakan klien tidak mendapat slot pengiriman. Slot pengiriman dibagi pada rentang 1 menit atau 60 detik dimana slot pengiriman sendiri dibagi menjadi 10 detik dan slot *idle* 30+slot pengiriman atau sama dengan 40 detik. Sehingga dalam satu menit terdapat 2 slot pengiriman dan 1 slot *idle*.

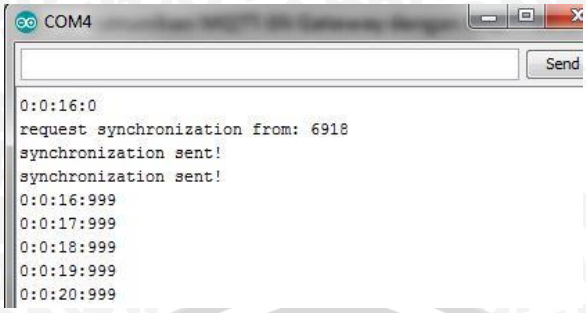
6.2 Pengujian dan Analisis MQTT-SN *Gateway*

MQTT-SN *gateway* berfungsi sebagai *root* yang secara *default* terletak pada level 0. MQTT-SN *gateway* berkomunikasi melalui media nirkabel dengan MQTT-SN *client*. Prosedur pengujian komunikasi MQTT-SN *gateway* dengan MQTT-SN *client* sebelumnya telah dijelaskan pada Tabel 3.5. Hasil pengujian keberhasilan komunikasi MQTT-SN *gateway* dengan MQTT-SN *client* dijelaskan pada Gambar 6.3, Gambar 6.4 dan Gambar 6.5. Gambar 6.3 Tampilan proses MQTT-SN *gateway* mengirim paket *discovery* seama 7 detik.



Gambar 6.3 Pengiriman paket *discovery* oleh *gateway*

Gambar 6.4 menunjukkan tampilan proses MQTT-SN *gateway* mengirim paket sinkronisasi menuju node dengan alamat 6918 berhasil dilakukan.



Gambar 6.4 Pengiriman balasan paket sinkronisasi oleh *gateway*

Gambar 6.5 menampilan proses MQTT-SN *gateway* menerima data pada slot waktu yang telah dijadwalkan.

Gambar 6.5 Penerimaan data sesuai slot waktu oleh *gateway*

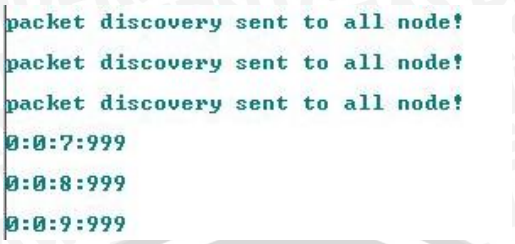
Dari hasil pengujian yang dilakukan sebanyak 5 kali percobaan maka disimpulkan, MQTT-SN *gateway* dapat menjalankan fungsinya yaitu; mengirimkan paket *discovery* selama 7 detik kemudian membalas permintaan sinkronisasi waktu oleh klien, mengirimkan slot waktu penjadwalan dibuktikan dengan dapat diterimanya data sensor sesuai dengan slot waktu yaitu 10 detik dan waktu *idle* sebesar 40 detik. Jumlah data yang diterima *gateway* tergantung dari besaran slot waktu yang ditentukan semakin kecil slot waktu semakin banyak data yang dapat diterima. Namun probabilitas tabrakan data juga dapat meningkat seiring kecilnya slot waktu yang diberikan.

6.3 Pengujian dan Analisis MQTT *Publisher*

MQTT-SN *gateway* berperan sebagai perantara antara lingkungan *sensor network* dan lingkangan simulasi MQTT pada jaringan lokal. Agar data sensor dapat diterima oleh MQTT *subscriber*, *gateway* meneruskan data yang diterima menuju MQTT *publisher*. Komunikasi MQTT-SN *gateway* dengan MQTT *publisher* menggunakan port serial. Port serial yang digunakan *gateway* dan MQTT *publisher* adalah port serial yang sama, sehingga untuk mengakses keduanya harus secara bergantian.

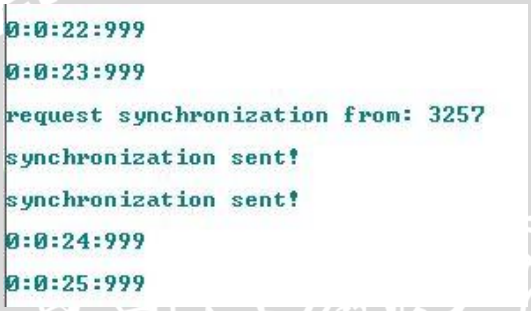
Luaran MQTT *publisher* sama dengan MQTT-SN *gateway*, hanya saja pada MQTT *publisher* terdapat fungsi yang menyeleksi data sensor sesuai dengan *topic*nya untuk di *publish* menuju *subscriber*. Keberhasilan komunikasi MQTT-SN *gateway* dengan MQTT *publisher* ditandai dengan luaran aplikasi *publisher* yang menyerupai MQTT-SN *gateway* dan dapat melakukan *publish* data menuju

subscriber. Keberhasilan MQTT *publisher* melakukan *publish* data dapat dilihat pada sub bab 6.4.

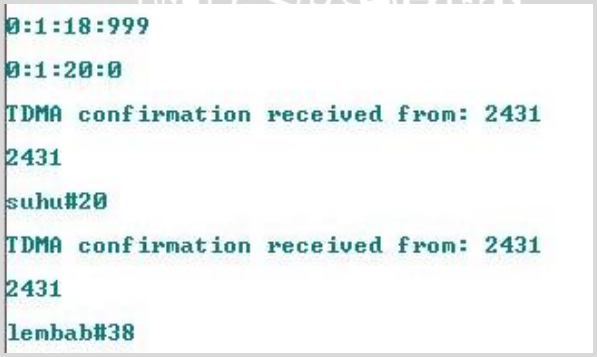


Gambar 6.6 Tampilan fase *discovery* oleh MQTT *publisher*

Gambar 6.6 menunjukkan bahwa MQTT *publisher* dapat mengasilkan luaran seperti MQTT-SN *gateway*, yaitu pengiriman paket *discovery*. Fungsionalitas lain seperti pengiriman balasan permintaan sinkronisasi dan penerimaan data sensor juga dapat dilakukan oleh MQTT *publisher*. Terubkti pada



Gambar 6.7 Tampilan pengiriman balasan paket sinkronisasi oleh MQTT *publisher*



Gambar 6.8 Tampilan Penerimaan data sesuai slot waktu oleh MQTT *publisher*

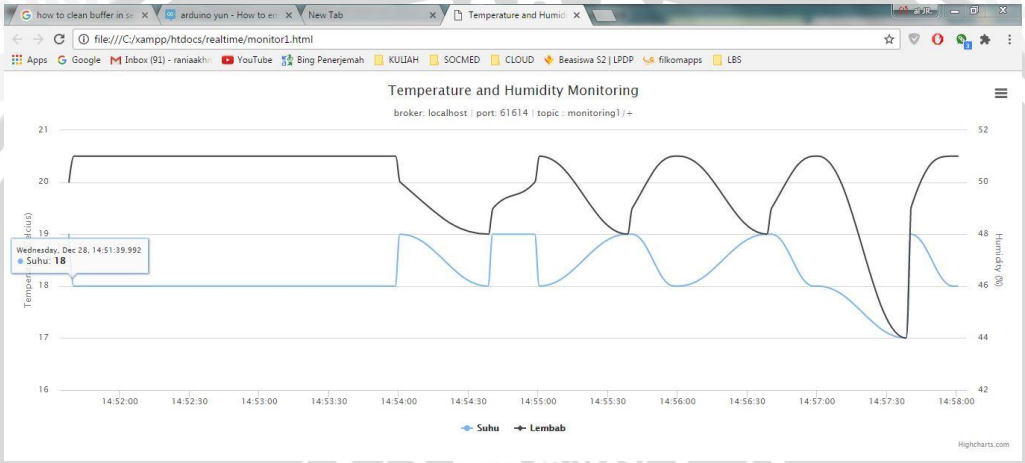
Dari hasil pengujian MQTT *publisher* yang dilakukan sebanyak 5 kali percobaan disimpulkan bahwa, MQTT *publisher* dan MQTT-SN *gateway* dapat berkomunikasi melalui port serial yang sama walaupun dalam pengaksesannya harus bergantian. Terbukti dengan keberhasilan MQTT *publisher* menampilkan luaran yang sama dengan MQTT-SN *gateway*. Hal tersebut disebabkan oleh port serial yang digunakan oleh *gateway* dan MQTT *publisher* adalah port yang sama. Dengan kata lain, MQTT *publisher* dapat menggantikan fungsi serial monitor

pada MQTT-SN *gateway*. MQTT *publisher* juga dapat mem*publish* data sesuai *topic* dibuktikan pada subbab 6.4.

6.4 Pengujian dan Analisis MQTT *Subscriber*

Pengujian MQTT *subscriber* adalah membuktikan bahwa *subscriber* dapat berlangganan data dari *publisher* sesuai dengan *topic*nya. Pada implementasinya aplikasi *subscriber* dibuat dengan menggunakan bahasa pemrograman HTML. Sehingga luaran yang dihasilkan berupa halaman web. Halaman web berisi grafik dengan satu sumbu x dan 2 sumbu y, sumbu x dengan label waktu dan sumbu y dengan label temperature dan kelembaban.

Gambar 6.9 merupakan hasil pengujian MQTT *publisher* dan juga MQTT *subscriber*.



Gambar 6.9 Tampilan luaran MQTT *subscriber*

Dari hasil pengujian MQTT *subscriber* disimpulkan bahwa, MQTT *subscriber* dapat berlangganan *topic* dari MQTT *publisher* yaitu suhu dan lembab dan menerima nilai suhu dan kelembaban seperti pada grafik Gambar 6.9. Waktu pada grafik menunjukkan waktu yang berjalan di sistem konfigurasi waktu pada laptop penguji tidak ada sinkronisasi waktu antara waktu pada lingkungan simulasi jaringan lokal dengan lingkungan sensor *network*.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil dari tahapan perancangan, implementasi, pengujian dan analisis hasil pengujian yang telah dilakukan, maka penulis dapat menyimpulkan sebagai berikut:

1. Implementasi protokol TPSN pada MQTT-SN *client* dapat dilakukan sebagai metode penyetaraan waktu antara klien dengan *gateway*. Proses penyetaraan waktu dengan protokol TPSN melalui 2 fase yaitu *discovery* dan sinkronisasi. Pada fase *discovery* jaringan *sensor network* berhasil membentuk topoogi hiraki dengan 4 level yaitu level 0, level 1, level 2 dan level 3. Fase sinkronisasi juga dapat dijalankan dengan rata-rata waktu tunggu sinkronisasi kurang dari 30 detik. Prosentase keberhasilan sinkronisasi waktu antara MQTT-SN *client* dengan MQTT-SN *gateway* mencapai 100%.
2. Implementasi protokol TDMA pada pengiriman pesan dengan format MQTT-SN *client* menuju *gateway* dapat dilakukan sebagai metode penjadwalan pengiriman data menggunakan pembagian slot waktu. slot waktu dibagi menjadi 10 detik dengan waktu *idle* sebesar 40 detik. Rentang waktu sekian dibuat untuk mengurangi kemungkinan interferensi data. Prosentase keberhasilan pengiriman pesan dengan *topic* yang sama sebesar 81,14% sedangkan untuk topik yang berbeda 55% dan 53,97%.
3. Komunikasi antara MQTT-SN *gateway* dengan MQTT *publisher* dilakukan agar data dari MQTT-SN *client* dapat tersalurkan menuju jaringan lokal. *Gateway* meneruskan data yang diterima dari klien menuju MQTT *publisher* melalui komunikasi port serial dengan cara mengaktifkan aplikasi broker terlebih dahulu. Forwarding data dari *gateway* menuju MQTT *publisher* berhasil dilakukan melalui port serial yang sama dengan pengaksesan serial monitor dilakukan secara bergantian.
4. MQTT *subscriber* dapat menampilkan pesan yang telah di*subscribe* dengan memanfaatkan port websocket pada aplikasi *broker Apache Activemq* yaitu, 61614*. Subscriber* dapat menerima data suhu dan kelembaban beserta nilainya.

7.2 Saran

Berdasarkan kesimpulan yang didapat oleh penulis, maka terdapat beberapa saran untuk peneliti selanjutnya dalam pengembangan sistem ini yaitu:

1. Disarankan untuk melakukan sinkronisasi waktu antara lingkungan *sensor network* dengan lingkungan simulasi pada jaringan lokal.

2. Menerapkan tipe arsitektur yang kedua yaitu, *transparent gateway* agar dapat dilakukan perbandingan performa tipe arsitektur mana yang lebih baik.
3. Melakukan penelitian lebih lanjut tentang analisis pembagian slot waktu pengiriman dan waktu *idle* agar didapatkan pembagian slot yang efektif dan efisien.

DAFTAR PUSTAKA

ActiveMQ, A. (2016). *ActiveMQ*. Retrieved September 2016, from The Apache Software Foundation: http://activemq.apache.org/

Azzahidin, M. A. (2016). *Implementasi Wireless Sensor Network Publisher Menggunakan Protokol Message Queue Telemetry Transport – Sensor Network (MQTT-SN).* Malang, Indonesia: Universitas Brawijaya.

Ball, B. (2008). *Everything You Need to Know about the nRF24L01 and MiRF-v2.* DIY Embedded.

Darmawan, A. A. (2016). Implementasi Refernce Broadcast Time Synchronization dengan Time Division Multiple Access pada Wireless Sensor Network.

D-Robotics UK. (2010). *DHT11 Humidity & Temperature Sensor.* UK: D-Robotics.

Elson, J. (n.d.). *Time Synchronization for Wireless Sensor Networks.* Los Angeles: University of California.

Erwanda, A. N. (2016). *Implementasi Time Synchronization Pada Wsn Untuk Metode TDMA Menggunakan Algoritma TPSN.* Malang, Indonesia: Universitas Brawijaya.

Govindan, K. (2015). *End-to-End Service Assurance in IoT MQTT-SN.* Denmark: IEEE Explorer.

Khalil, N. (2014). *Wireless Sensors Networks for Internet of Things.*

Kurose, & Ross. (2012). *Multiple Access Protocols Link Layer Addressing.* NYU.

Python, E. (2015). *Eclipse Paho*. Retrieved September 2016, from Python Eclipse Paho: https://eclipse.org/paho/

Stanford-Clark, A. (2013). *MQTT For Sensor Networks (MQTT-SN) Protocol Specification.* International Business Machines Corporation (IBM).

Sundararaman, B. (2005). *Clock Synchronization for Wireless Sensor Networks: A Survey.* Chicago: University of Illinois.

LAMPIRAN A KODE PROGRAM MQTT *SUBSCRIBER*

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
<title>Temperature and Humidity Monitoring</title>

<script type="text/javascript" src="jquery.min.js"></script>
<script src="mqttws31.js" type="text/javascript"></script>
<script type="text/javascript">

//settings BEGIN
 var color='';
 var MQTTbroker = 'localhost';
 var MQTTport = 61614;
 var MQTTsubTopic = 'monitoring/+';

//settings END
 var chart;
 var dataTopics = new Array();

//mqtt broker
 var client = new Paho.MQTT.Client(MQTTbroker, MQTTport,
"myclientid_" + parseInt(Math.random() * 100, 10));
 client.onMessageArrived = onMessageArrived;
 client.onConnectionLost = onConnectionLost;

//mqtt connecton options including the mqtt broker subscribe
 var options = {
  timeout: 3,
  onSuccess: function () {
    console.log("mqtt connected");
    // Connection succeeded; subscribe to our topics
    client.subscribe(MQTTsubTopic, {QoS: 1});
   },
  onFailure: function (message) {
    console.log("Connection failed, ERROR: " +
message.errorMessage);
   }
 };

 function onConnectionLost(responseObject) {
   console.log("connection lost: " +
responseObject.errorMessage);
 };

//what is done when a message arrives from the broker
 function onMessageArrived(message) {
   console.log(message.destinationName,
'',message.payloadString);

//check if it is a new topic, if not add it to the array
 if (dataTopics.indexOf(message.destinationName) < 0) {
  dataTopics.push(message.destinationName);
  var y = dataTopics.indexOf(message.destinationName);
 }
 var y = dataTopics.indexOf(message.destinationName);
 var myEpoch = new Date().getTime();
 var thenum = message.payloadString.replace( /^\D+/g, '');
 var plotMqtt = [myEpoch, Number(thenum)]; //create the array
   if (isNumber(thenum)) {
     console.log('is a propper number, will send to chart.')
                    plot(plotMqtt, y);
```

```
    };
  };

//check if a real number
  function isNumber(n) {
    return !isNaN(parseFloat(n)) && isFinite(n);
  };

//function that is called once the document has loaded
 function init() {
   Highcharts.setOptions({
      global: {
      useUTC: false
      }
   });
 // Connect to MQTT broker
 client.connect(options);
 };

//this adds the plots to the chart
 function plot(point, chartno) {
   console.log(point);
   var series = chart.series[0],
   shift = series.data.length > 30;
   chart.series[chartno].addPoint(point, true, shift);
      };

      $(document).ready(function() {
      client.onConnectionLost = onConnectionLost;
          chart = new Highcharts.Chart({
              chart: {
                  renderTo: 'container',
                  defaultSeriesType: 'spline'
              },
              title: {
                  text: 'Temperature and Humidity Monitoring'
              },
              subtitle: {
                                  text: 'broker: ' + MQTTbroker
+ ' | port: ' + MQTTport + ' | topic : ' + MQTTsubTopic
                          },
              xAxis: {
                  type: 'datetime',
                                  tickInterval: 24 * 1000,

                  tickPixelInterval: 150,
                  maxZoom: 20 * 1000
              },
              yAxis: [{
                  minPadding: 0.2,
                  maxPadding: 0.2,
                  title: {
                      text: 'Temperature (Celcius)',
                      //margin: 80
                  }
              },{
                                  title: {
                                  text: 'Humidity (%)'
                                  },
                                  opposite: true
                       }],
                       series: [{
                                  yAxis: 0,
                                  name: 'Suhu',
                                  },
```

```
                        {yAxis: 1,
                        name: 'Lembab',

                        }
                    ]
            });
        });

</script>

        <script src="highstock.js"></script>
        <script src="exporting.js"></script>

        </head>
            <body onload="init();"><!--Start the javascript
ball rolling and connect to the mqtt broker-->
                    <div id="container" style="height: 500px;
min-width: 500px"></div><!-- this the placeholder for the
chart-->
            </body>
</html>
```