IMPLEMENTASI WEBSOCKET UNTUK MONITORING DAN KONTROLING PERANGKAT BERBASIS ARDUINO

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yosia Rimbo Deantama
NIM: 125150207111092

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI WEBSOCKET UNTUK MONITORING DAN KONTROLING PERANGKAT BERBASIS ARDUINO

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yosia Rimbo Deantama
NIM: 125150207111092

Skripsi ini telah diuji dan dinyatakan lulus pada
23 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adhitya Bhawiyuga, S.Kom, M.S	Rakhmadhany Primananda, S.T, M.Kom
NIP: 201405 890720 1 001	NIP:

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, 14 Agustus 2016

Yosia Rimbo Deantama

NIM: 125150207111092

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan skripsi yang berjudul "Implementasi Websocket untuk Monitoring dan Kontroling Perangkat Berbasis Arduino" ini dapat terselesaikan.

Penulis sangat menyadari bahwa skripsi ini tidak dapat terselesaikan tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada:

1. Bapak Adhitya Bhawiyuga, S.Kom., M.S dan Bapak Rakhmadhany Primananda, S.T, M.Kom selaku dosen pembimbing skripsi yang telah membimbing dan mengarahkan penulis untuk dapat menyelesaikan skripsi ini.
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku ketua Program Studi Informatika.
3. Bapak Kasyful Amron, S.T, M.Sc dan Bapak Mahendra Data, S.Kom, M.Kom selaku penguji I dan penguji II yang telah memberikan pengarahan, saran dan masukan sehingga skripsi ini bisa terselesaikan dengan jauh lebih baik.
4. Keluarga penulis, Bapak Gunung Djoko Suharyo, Ibu Laksmiwati, Hana Ayu Petricia serta saudara yang telah memberi dukungan berupa nasehat, kasih sayang, semangat, dan doa untuk dapat menyelesaikan skripsi ini, serta kesabarannya dalam membesarkan dan mendidik penulis.
5. Felia Eliantara, Hanif Kuncahyo Adi, Akbar Nourma Putra, Hastian Bayu, Irsyad Fauzan, Agnes Cexariana, Ayu Permatasari, Astanessa Kezia, Vitara Nindya, serta teman – teman asisten sistem operasi 2015/2016 atas dukungan, saran dan kesediannya membantu pengadaan *smartphone android* sebagai sarana pengembangan aplikasi dan pengujian sistem dalam penelitian ini. Sehingga skripsi ini dapat terselesaikan dengan baik.
6. Seluruh civitas akademika Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.

Dalam penyusunan skripsi ini, penulis menyadari bahwa masih banyak kekurangan, sehingga penulis sangat mengharapkan adanya kritik dan saran yang membangun. Akhir kata penulis berharap skripsi ini dapat berguna dan bermanfaat bagi semua pihak yang membaca.

Malang, 14 Agustus 2016

Penulis

yosiarimbo@gmail.com

ABSTRAK

Internet of Things (*IoT*) adalah sebuah konsep dimana suatu objek mati memiliki kemampuan untuk menerima dan mengirimkan sebuah data melalui koneksi jaringan. Dengan konsep *IoT*, manusia memungkinkan untuk mengontrol dan memonitoring sebuah lingkungan. Untuk mengimplementasikan konsep *IoT* maka dibutuhkan peran serta teknologi, tidak tekecuali teknologi jaringan. Tetapi semakin banyaknya protokol jaringan yang muncul, membuat komunikasi antar objek dalam *IoT* menjadi rumit. Maka dari itu dibutuhkan sebuah protokol jaringan yang mudah dikembangkan dan mudah dipahami kebanyakan orang. Pada penelitian ini, penulis menggunakan sebuah protokol jaringan yang mudah dikembangkan untuk mengimplementasikan *IoT* yaitu *Websocket*. Selain *Websocket* yang digunakan untuk jalur komunikasi, penulis juga menggunakan Arduino Uno sebagai mikrokontroler yang digunakan untuk mendapatkan dan memproses data dari sebuah lingkungan dan Redis sebagai media penyimpanan data. Untuk mempraktekkan implementasi Websocket untuk monitoring dan kontroling perangkat berbasis Arduino, maka dilakukan dua implementasi. Implementasi yang pertama adalah implementasi *Websocket* untuk mengontrol empat lampu *LED* yang terpasang pada Arduino Uno. Implementasi kedua adalah implementasi *Websocket* untuk memonitor data berupa suhu dan kelembapan dari sensor DHT 11 yang terpasang pada Arduino Uno. Kedua pengimplementasian sama-sama menggunakan metodologi client-server. Untuk melakukan satu kali proses kontroling dan monitoring perangkat, waktu yang dibutuhkan tidak lebih dari 1 detik dengan persentase keberhasilan 100%. Terdapat penurunan performa pada saat dilakukan request sebanyak 350 per detiknya atau terdapat 300 client yang terkoneksi secara bersamaan. Dengan semakin banyaknya request dan jumlah client yang terkoneksi maka akan berpengaruh terhadap Round Trip Time dan *request* yang mampu dilayani oleh sistem.

Kata kunci: *Websocket, Arduino, Redis, Internet of Things*

ABSTRACT

Internet of Things (IoT) is a concept where an object has the ability to receive and transmit data through a network connection. With the concept of IOT, humans can do controlling and monitoring an environment. To implement the concept of IOT it takes the role of technology, include network technology. But the increasing number of network protocols that appear, making communication between objects in the IoT becomes complicated. Therefore it needs a network protocol that is easy to develop and easy to understand for most people. In this study, the authors use a network protocol which is easy to developed to implement IoT, the protocol is Websockets. Besides Websockets which used for communication lines, the author also uses Arduino Uno as a micro-controller that used to acquire and process data from an environmental and Redis as a data storage. To practice Websockets implementation for monitoring and controlling devices based on Arduino, there has the two implementations. The first is Websockets implementation to control four LED lights mounted on the Arduino Uno. The second is Websockets implementation for monitoring temperature and humidity data from DHT 11 which mounted on the Arduino Uno.To perform one process controlling and monitoring devices, time needed is not more than 1 second with a percentage of 100% success. There is a decrease in performance at the time of the request as much as 350 per second or 300 clients are connected simultaneously. With the increasing number of requests and the number of clients connected it will affect the Round Trip Time and requests are able to be serviced by the system.

*Key*words: *Websocket, Arduino, Redis, Internet of Things*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

BAB 1 PENDAHULUAN

1.1 Latar belakang

Internet of Things (*IoT*) adalah sebuah konsep dimana suatu objek mati memiliki kemampuan untuk menerima dan mengirimkan sebuah data melalui koneksi jaringan. Salah satu contoh *IoT* adalah interaksi manusia dengan sebuah perangkat, dimana perangkat tersebut digunakan untuk mengontrol sebuah lingkungan atau memonitoring kondisi sebuah lingkungan. Selain membutuhkan perangkat seperti sensor atau aktuator, untuk mengimplementasikan konsep *IoT* dibutuhkan peran serta teknologi lain, seperti teknologi komputer dan teknologi jaringan.

Perkembangan teknologi yang sangat pesat berdampak pada bertambah banyaknya protokol jaringan khususnya protokol jaringan untuk komunikasi client dengan suatu perangkat. Munculnya berbagai jenis protokol tersebut menimbulkan satu masalah yaitu membuat pengembangan sistem menjadi rumit. Untuk mengatasi masalah tersebut dibutuhkan protokol jaringan yang sederhana dan sering digunakan kebanyakan orang.

Hyper Text Transfer Protocol (*HTTP*) adalah salah satu protokol yang paling banyak digunakan dalam internet, dengan *HTTP* yang bekerja di layer *TCP* membuat *HTTP* menjadi protokol yang mampu mengirim berita, video, dan melayani banyak aplikasi web (Megyesi, Kramer & Molnar, 1997). Dalam sebuah situs, yaitu dzone.com membuat sebuah survei tentang penggunaan *IoT*. Di dalam artikel tersebut, sebanyak 265 responden dari 528 responden menjawab *HTTP* merupakan protokol yang paling banyak digunakan dalam *IoT*. Dari paparan di atas, maka dalam peneltian ini dibutuhkan sebuah protokol yang mempunyai karakteristik *HTTP* yang mampu mengimplementasikan kegiatan kontroling dan monitoring perangkat berbasis Arduino.

Websocket adalah protokol yang mampu menyediakan komunikasi full-duplex pada protokol HTTP dengan menggunakan satu TCP socket saja. Dengan websocket memungkinkan untuk dibuatnya sebuah aplikasi berbasis HTTP yang *real time*. (Srinivasan, Scharnagl & Schilling, 2013). Dari penjelasan websocket di atas, menunjukkan bahwa websocket merupakan sebuah teknologi yang berjalan pada protokol *HTTP.* Dikarenakan adanya kesinambungan pada pengertian Websocket dengan permasalahan yang dijelaskan sebelumnya, maka dalam penelitian ini penulis menggunakan protokol Websocket untuk memonitoring dan kontroling perangakat berbasis Websocket.

Alasan penulis memilih protokol *Websocket* sebagai jalur komunikasi diperkuat dengan penelitian sebelumnya, yaitu *Analysis of WebSockets as the New Age Protocol for Remote Robot Tele-operation* (Srinivasan, Scharnagl & Schilling, 2013). Selain itu, penelitian lain yang berkaitan dengan Websocket adalah *Web Display of Real time Wind Sensor Data* (Pimentel & Nickerson, 2011). Dalam dua penelitian tersebut, disebutkan *Websocket* adalah protokol yang mampu memberikan dampak baik yang signifikan pada sisi latency. Dimana latency yang didapatkan

dari dua penelitian tersebut sangat kecil, sehingga menyebabkan sistem dapat mengirim sebuah data kurang dari satu detik.

Selain membutuhkan teknologi jaringan, agar perangkat seperti aktuator atau sensor dapat bekerja maka dibutuhkan sebuah teknologi komputer berupa mikrokontroler yang berfungsi sebagai pengolah data baik itu data yang akan dikirimkan ke perangkat atau data yang diterima dari perangkat. Selain memberi dampak pada perkembangan teknologi jaringan, perkembangan teknologi yang sangat pesat juga berdampak pada teknologi mikrokontroler. Dampak yang bisa dirasakan adalah munculnya bermacam-macam mikrokontroler yang mempunyai fungsi dan bentuk fisik yang bervariasi, contohnya muncul mikrokontroler Arduino, Raspberry, BeagleBone dan lain-lain.

Dalam penelitian *Experiences on Using Arduino for Laboratory Experiments of Automatic Control and Robotics* (Candelas, dkk, 2015). Dilakukan analisis tentang mikrokontroler Arduino dimana disimpulkan Arduino adalah sebuah mikrokontroler yang mudah dipelajari, memiliki harga yang murah serta memiliki *free software*. Maka dari itu, selain menggunakan protokol *Websocket* penulis juga menggunakan Arduino sebagai mikrokontroler.

Sebuah perangkat seperti sensor dapat melakukan 2 hal, yaitu mengontrol sebuah lingkungan dan mendapatkan kondisi sebuah lingkungan. Agar dalam penelitian ini dapat mencakup penggunaan Websocket secara menyeluruh pada sebuah perangkat seperti sensor, maka dalam penelitian ini dilakukan 2 pengimplementasian berupa kontroling dan monitoring perangkat berbasis Arduino.

Mengacu pada permasalahan yang telah disampaikan, judul yang diambil dalam skripsi ini adalah "Implementasi Websocket untuk Monitoring dan Kontroling Perangkat Berbasis Arduino". Dimana untuk pengimplementasian monitoring menggunakan perangkat 4 lampu *LED* yang terpasang di Arduino. Dan untuk pengimplementasian kontroling menggunakan sensor DHT 11 yang terpasang di Arduino. Diharapkan tema skripsi yang diangkat akan menjadi salah satu solusi supaya interkasi manusia dengan dengan sebuah perangkat berbasis Arduino dapat terus dikembangkan dan dapat dirasakan manfaatnya bagi para penggunanya

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dijabarkan, maka dapat dirumuskan beberapa permasalahan sebagai berikut:

1. Bagaimana menerapkan protokol Websocket untuk komunikasi antara server dengan client?
2. Bagaiamana mekanisme mengirim dan menerima data dari server ke Arduino?
3. Bagaimana performa kecepatan aliran data dan kehandalan sistem dalam menangani request yang diberikan oleh sistem?

1.3 Tujuan

Tujuan umum penulisan skripsi ini adalah :

1. Dapat merancang dan menerapkan Websocket sebagai metode komunikasi client server
2. Dapat merancang komunikasi untuk mengirim dan menerima data dari server ke Arduino
3. Dapat mengetahui performa dan kehandalan sistem.

1.4 Manfaat

Penulisan skripsi ini diharapkan mempunyai manfaat yang baik dan berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan adalah sebagai berikut:

A. Bagi penulis
   1. Sebagai media untuk pengimplementasian ilmu pengetahuan teknologi yang telah didapat selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya.
   2. Mendapatkan pengetahuan dan wawasan tentang komunikasi jaringan menggunakan Websocket

B. Bagi pembaca
   1. Dapat menjadi acuan pengembangan protokol Websocket pada perangkat berbasis Arduino

1.5 Batasan masalah

Agar pembahasan tidak melebar dari latar belakang dan terfokus dengan apa yang berkaitan dengan sistem, maka dapat di jabarkan batasan dalam penelitian ini yaitu :

1. Menggunakan mikrokontroller Arduino Uno.
2. Pengimplementasian Websocket untuk mengontrol perangkat berbasis Arduino menggunakan lampu *LED* sebagai perangkat.
3. Pengimplemntasian Websocket untuk memonitoring perangkat berbasis Arduino menggunakan sensor DHT 11 untuk mendapatkan data berupa suhu dan kelembapan ruangan.
4. Database yang digunakan adalah Redis.
5. Pengguna mengakses sistem menggunakan aplikasi web.

1.6 Sistematika pembahasan

Sistematika pembahasan dari penyusunan penelitian yang direncanakan adalah sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan terdiri dari latar belakang, identifikasi dan pembatasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika pembahasan dari proyek akhir ini. Selain itu sebuah sistematika pembahasan digunakan sebagai acuan untuk melakukan pembahasan terhadap penelitian yang dilakukan secara sistematis.

BAB II LANDASAN KEPUSTAKAAN

Bab landasan kepustakaan menjelaskan tentang teori – teori yang di pakai dalam penelitian, temuan dan bahan penelitian yang mungkin sebelumnya diperoleh dari beberapa refrensi yang menunjang penelitian dalam penulisan skripsi. Ada juga teori – teori yang tercakup dalam bab ini adalah mengenai protokol Websocket, Arduino, serta Redis.

BAB III METODOLOGI PENELITIAN

Bab Metodologi tahapan-tahapan langkah-langkah dalam melakukan penelitian. Langkah-langkah seperti perancangan, implementasi, pengujian, serta analisa hasil dibahas secara umum. Selain itu pada bab metodologi juga dijelaskan secara singkat mengenai perangkat keras yang dibutuhkan, arsitektur sistem, pengolahan format data serta proses transmisi dalam penelitian.

BAB IV PERANCANGAN

Bab Perancangan menjelaskan tentang bagaimana arsitektur jaringan yang sesuai dengan objek penelitian. Dibuat pula rancangan sistem untuk melakukan kontroling dan monitoring terhadap pernagkat berbasis Arduino. Pada bab ini juga akan akan dijelaskan pula bagaimana proses alur data dari client ke perangkat dan sebaliknya. Skenario pengujian yang mencakup tujuan penelitian juga disampaikan dalam bab ini.

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab Implementasi dan Pengujian menjelaskan tentang bagaimana menerapkan sistem yang telah dibuat berdasarkan sistematika sebelumnya secara rinci beserta dengan langkah-langkah pengerjaan yang dilakukan serta menampilkan gambar-gambar implementasi yang dilakukan. Setelah implementasi selesai dilakukan, dilakukan pengujian terhadap sistem yang telah dibuat berdasarkan skenario yang telah dibuat pada bab sebelumnya.

BAB VI PEMBAHASAN

Bab Pembahasan menyajikan data hasil pengujian beserta analisis dari implementasi sistem yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang telah ditentukan. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem dapat mencapai tujuan.

BAB VII PENUTUP

Pada Bab Penutup dipaparkan kesimpulan dari pelaksanaan penelitian. Kesimpulan ini dibuat dengan hasil pengujian dan analisis terhadap perancangan dan implementasi arsitektur sistem sebagai dasarnya. Saran-saran juga diberikan agar hasil dari penelitian ini dapat diperbaiki dan disempurnakan apabila penelitian ini dikembangkan kemudian.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka membahas penelitian yang telah ada dan yang diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Kajian Pustaka

Kajian pustaka membahas penelitian yang telah ada dan yang diusulkan. Pada penelitian ini kajian pustaka diambil dari beberapa penelitian yang relevan yang pernah dilakukan. Dasar atau acuan yang berupa teori-teori atau temuan-temuan melalui hasil berbagai penelitian sebelumnya merupakan hal yang sangat perlu dan dapat dijadikan sebagai data pendukung. Salah satu data pendukung yang menurut peneliti perlu untuk dijadikan bagian tersendiri adalah penelitian terdahulu yang relevan dengan permasalah yang sedang dibahas dalam penelitian ini.

Untuk pengimplementasian Websocket dalam mengontrol atau memonitoring sebuah perangkat ini telah dilakukan beberapa penelitian terdahulu seperti halnya Zhang dan dkk (2013) yang melakukan pengimplementasian Websocket untuk memonitoring keadaan pasien menggunakan sensor yang terpasang pada badan pasien, juga Pimentel dan Nickerson (2011) menggunakan Websocket untuk memonitoring kecepatan angin. Sedangkan Salzman dkk (2013) menggunakan Websocket untuk mengontrol berbagai kegiatan di dalam lab. **Tabel 2.1** menjelaskan tentang kajian pustaka yang digunakan dalam penelitian.

Tabel 2. 1 Kajian Pustaka

NO	Judul	Penulis	Perbandingan	
			Kajian Pustaka	Tugas Akhir Penulis
1	A Secure and Scalable Telemonitoring System Using Ultra-Low-Energy Wireless Sensor *Interface* for Long-Term Monitoring in Life Science Applications	W. Zhang, P. Passow, E. Jovanov, R. Stoll dan K. Thurow	Menggunakan Websocket untuk memonitoring keadaan pasien berbasis aplikasi mobile	Menggunakan Websocket untuk memonitoring suhu dan kelembapan berbasis aplikasi web

2	The Smart Device Specification for Remote Labs	Chritophe Salzmann, Sten Govaerts, Wissam Halimi dan Denis Gillet	Menggunakn Websocket untuk kontroling kegiatan lab menggunakan sensor dan actuator seperti menyalakan alarm tanpa menggunakan database	Menggunakan Websocket untuk kontroling lampu LED dan monitoring suhu dengan menggunakan database Redis
3	Web Display of *Real time* Wid Sensor Data	Victoria Pimentel dan Bradford Nickerson	Menggunakn Websocket untuk monitoring kecepatan angin dimana pengiriman data dari sensor tidak melalui mikrokontroler dan sistem menggunakan MySQL	Menggunakan Websocket untuk monitoring suhu dan kelembapan dimana pengiriman data dari sensor melalui mikrokontroler Arduino dan sistem menggunakan database Redis

2.2 Dasar Teori

Berdasarkan beberapa informasi yang didapat dari beberapa kajian pustaka, maka dalam "Implementasi Websocket untuk Kontroling dan Monitoring Perangkat Berbasis Arduino" terdapat beberapa dasar teori, antara lain :

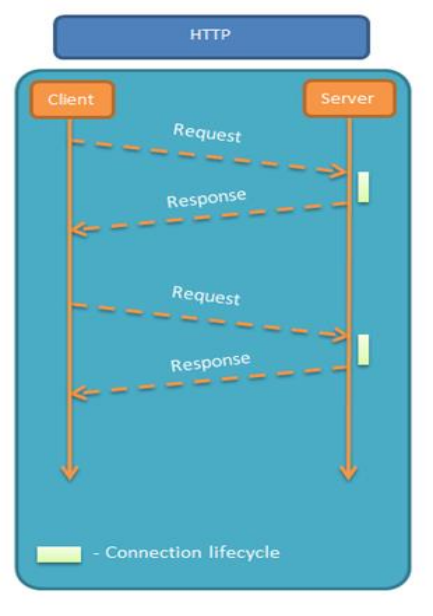
2.2.1 HTTP

HTTP adalah sebuah protokol jaringan lapisan aplikasi yang digunakan untuk sistem informasi terdistribusi. Penggunaannya banyak pada pengambilan sumber daya yang saling terhubung dengan sebuah dokumen *hiperteks* yang kemudian membentuk *World Wide Web*. *HTTP* merupakan protokol yang menyediakan perintah dalam komunikasi antar jaringan, yaitu komunikasi antara jaringan komputer *client* dengan web server. Dalam komunikasi ini,komputer *client* melakukan permintaan dengan mengetikkan alamat atau *website* yang ingindi akses. Sedangkan server mengolah permintaan tersebut berdasarkan kode protokol yang diinputkan.

HTTP berada pada *port* 80 dan biasanya didirikan di atas *TCP*/IP. Pada *HTTP* terdapat mekanisme koneksi *reuse* dimana header menggunakan fungsi *Keep-Alive*. Dengan adanya mekanisme tersebut diharapkan dapat mengurangi

pekerjaan CPU, mengurangi penggunaan memori, mengurangi kemacetan jaringan dan mengurangi *latency* pada saat terjadi proses *Hand Shaking*. (Lee, *Field*ing & Frystyk, 1996)

Karakteristik *HTTP* secara sederhana dapat dijelaskan dengan adanya metode request *response*, dimana client harus mengirimkan terlebih dahulu *HTTP* request untuk mendapatkan data dari server. Setelah itu data yang diminta client akan dikirimkan oleh server melewati *HTTP response*. Tetapi terdapat masalah pada karakteristik komunikasi *HTTP*, dengan karakteristik *HTTP* yang bersifat *half-duplex*. Jika client ingin mengambil lebih banyak data dari server maka client harus melewati satu siklus komunikasi dengan *HTTP* request dibalas *HTTP response*, dan akibatnya beban web server menjadi bertambah dan terjadi kemacetan pada jalur komunikasi. Masalah tersebut menjadi masalah besar jika sistem menggunakan konsep *IoT* dengan sejumlah perangkat saling berhubungan dan mengutamakan *real time*. Cara kerja secara umum dapat dilihat pada **Gambar 2.1**.



Gambar 2. 1 Cara Kerja HTTP

[sumber : Joo-Ho Lee 2013]

2.2.2 Web Socket

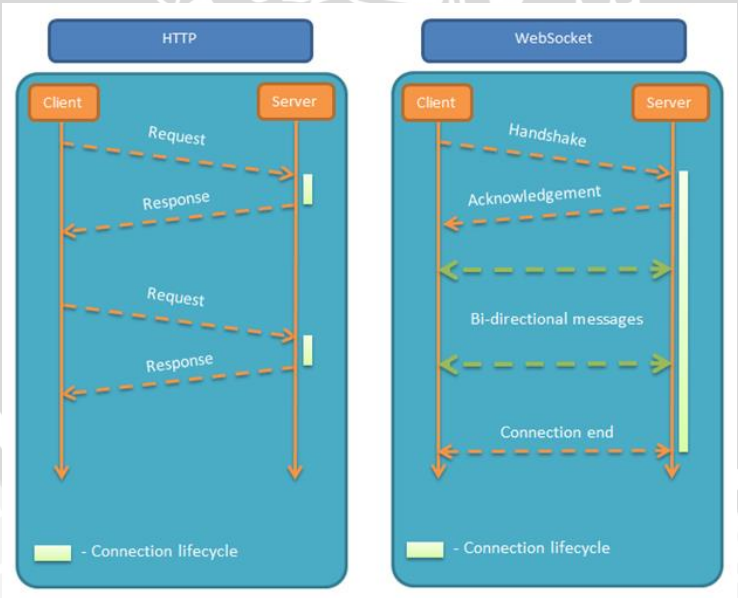
Pada awalnya, untuk menciptakan aplikasi web yang membutuhkan komunikasi secara dua arah antara klien dan server (misalnya, aplikasi *messenger* dan game) dapat dilakukan dengan melakukan manipulasi pesan *HTTP* untuk dapat terus menerus melakukan komunikasi atau biasa dikenal dengan istilah *polling* [RFC6202]. Hal ini bisa dilakukan namun menyebabkan beberapa masalah, antara lain:

1. Server dipaksa untuk menggunakan sejumlah komunikasi *TCP* dengan koneksi berbeda untuk setiap klien. Akan terbentuk komunikasi *TCP* baru untul setiap pesan yang masuk.

2. *Wire protocol* memiliki *overhead* yang tinggi, dengan masing-masing pesan *client-to- server* memiliki header *HTTP*.
3. script sisi klien dipaksa untuk mempertahankan pemetaan dari koneksi luar untuk melacak balasan dari koneksi masuk.

Sebuah solusi sederhana kemudian diciptakan dengan menggunakan koneksi *TCP* tunggal untuk komunikasi data di kedua arah. Layanan ini yang disediakan oleh Protocol WebSocket. Dikombinasikan dengan WebSocket API [WSAPI], websocket menyediakan alternatif pengganti *HTTP* polling melakukan sambungan dua arah dari halaman web ke server. Protokol WebSocket ini dirancang untuk menggantikan metode komunikasi dua arah yang sudah ada menggunakan *HTTP* sebagai *layer* trans*port*asi untuk mendapatkan keuntungan dari kemampuan *HTTP* berupa *proxy*, penyaringan, dan otentikasi. Websocket berjalan dalam *port* 80 dan 443 sehingga mudah untuk diimplementasikan karena mengakomodasi sifat-sifat *HTTP*.

Protokol Websocket mampu memberikan layanan komunikasi dua arah secara bersamaan (*bidirectional full duplex*) berbeda dengan HTTP yang hanya mampu memberikan layanan komunikasi *half-duplex*. Selain itu Websocket memiliki *header* yang kecil. Namun Websocket tidak bisa dipertimbangkan untuk menjadi alternatif pengganti *TCP*, karena Websocket dan *TCP* bekerja dalam dua lapisan jaringan yang berbeda dalam model protokol internet (Skvorc, dkk, 2014).Untuk dapat menggunakan protokol Websocket, diperlukan proses handshake agar client dapat terhubung dengan server (Fette & Melkinov, 2011). Untuk melihat perbedaan antara HTTP dan Websocket dapat dilihat pada **Gambar 2.2.**



Gambar 2. 2 Perbedaan HTTP dengan Websocket

[sumber : Joo-Ho Lee 2013]

Untuk membuka koneksi Websocket, client harus membuka meminta koneksi jaringan protokol *TCP* ke server terlebih dahulu. *Hypertext Transfer Protocol* (*HTTP*) merupakan protokol standar saat koneksi *TCP* terbuka. Setelah itu, client meminta server untuk mengubah protokol *HTTP* ke protokol Websocket. Setelah server merespon permintaan client, maka koneksi Websocket terbuka. Koneksi websocket ini akan terus terbuka hingga ada perintah untuk memutuskan koneksi. Pengiriman informasi dari server ke client dalam protokol ini tidak memerlukan lagi pengenalan client, sehingga jumlah ukuran paket yang dikirimkan lebih sedikit. Hal ini sangat efisien untuk mengirimkan informasi secara cepat. Berbeda dengan *HTTP* untuk melakukan pengiriman informasi dibutuhkan sebuah *reques*t dari client.

2.2.3 *Twisted*

Twisted adalah sebuah *framework* Python yang digunakan untuk pemrograman jaringan, dimana *twisted* menyediakan fungsi tingkat dasar maupun tingkat atas untuk membangun aplikasi berbasis jaringan. Sampai sekarang ini *twisted* sudah mampu mendukung macam-macam bentuk komunikasi seperti *TCP, UDP, SSL, HTTP,* dan *FTP*. *Framework* ini dibangun untuk implementasi client server, dimana dalam *twisted* tersedia fungsi *event-driven* yang bisa dipanggil ketika terdapat suatu *event* seperti *event* dalam menerima data dari client.

Framework twisted mempunyai beberapa kelebihan dibanding dengan *library-library* python yang lainnya yaitu :

1. Mendukung kinerja Python. *Twisted* dituliskan dalam bahasa Python dan berupa object oriented, dimana *twisted* dianggap cukup mudah untuk dibaca, ditulis dan dijalankan. Dan karena python merupakan cross-platform, maka aplikasi *twisted* mampu dijalankan pada beberapa macam sistem operasi seperti Linux, Windows, Mac OS, dan Unix.
2. Bersifat *asynchronous* dan *event-based*. Dengan sifat *twisted* yang *asynchronous* dan *event-based*, menyebabkan aplikasi yang dibangun menggunakan *twisted* menjadi responsif selama operasi jaringan dilakukan. Dengan begitu memungkinkan dibangunnya sebuah aplikasi yang mempunyai kompleksitas fungsi yang rumit dan mempunyai kebutuhan untuk membangun komunikasi dengan banyak client.
3. Mempunnyai fitur yang lengkap. Hampir semua fungsi yang digunakan untuk membuat suatu aplikasi sudah tersedia dalam *framework* seperti fungsi untuk chatting, SSH, Telnet, RPC, dan akses database.
4. Fleksibel. Di dalam *framework twisted*, pengguna dapat membuat cara kerja protokol, sehinga pengguna bisa memodifikasi kinerja protokol sesusai dengan kebutuhan.
5. *Framework twisted* bersifat *open source*, sehingga pengguna dapat menggunakannya secara bebas dan gratis.

6. Banyak komunitas yang sudah mendukung *twisted*, sehingga pengguna dapat mencari solusi dalam memecahkan masalah pada *twisted* dengan bertanya pada komunitas.
7. Dapat terintegrasi dengan beberapa layanan yang berbeda. Dengan menggunakan *twisted* maka memungkinkan pengguna untuk berkomunikasi atau bertukar data dengan sistem yang menggunakan protokol yang berbeda.

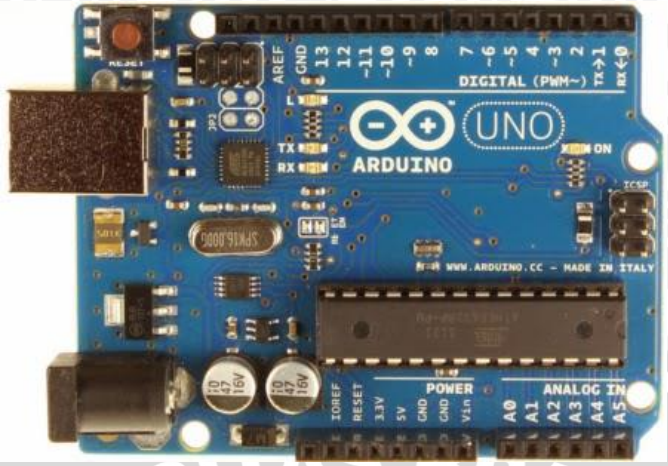
Dalam pengimplementasian *twisted*, terdapat 3 komponen utama yaitu *reactor*, *factory* dan *protocol*. Reactor adalah sebuah *event loop* yang berjalan sepanjang program berjalan atau dapat dilihat sebagai *infinite looping*. Dalam *reactor* juga terdapat method pengkoneksian dengan TCP, dengan menggunakan fungsi *listenTCP()*. Dan fungsi *run()* digunakan untuk memulai *event loop* yang berjalan terus-menerus.

Selain *reactor*, terdapat *protocol* yang merupakan komponen utama dari *twisted*. Pada bagian *protocol* akan diimplementasikan pemroses *event* jaringan, terdapat 4 fungsi utama dalam *protocol* yaitu *makeConnection* untuk membuat koneksi antara 2 *endpoint*, sedangakn *connectionMade* dipanggil ketika jalur komunikasi sudah berhasil terbentuk, *dataReceived* dipanggil ketika terjadi *event* penerimaan data dari titik lain, dan *connectionLost* dipanggil ketika jalur komunikasi ditutup. *Instance* dari protocol akan dibuat setiap ada koneksi yang terbentuk. Untuk menangani pembuatan *instance* protokol tersebut diperlukan sebuah class yang menanganinya, dan *class factory* adalah komponen utama yang menangani pembuatan *instance* tersebut. (McKellar & Fettig, 2012)

2.2.4 Arduino Uno

Arduino adalah kit elektronik atau papan rangkaian elektronik yang di dalamnya terdapat komponen utama yaitu sebuah *chip* mikrokontroler dengan jenis AVR dari perusahaan Atmel yang bersifat *open source*, fleksibel, dan mudah digunakan dalam hal perangkat keras dan perangkat lunak, dari sisi perangkat kerasnya arduino menggunakan prosesor Atmel AVR (Atmega8 atau Atmega168) dan memiliki pin yang dapat digunakan sebagai *input* maupun *output*. Sedangkan, dari sisi perangkat lunaknya, Arduino menggunakan bahasa pemrograman umum dan tersedia *boot-loader* yang berjalan pada modulnya.

Bahasa pemrograman adalah bahasa C tetapi bahasa ini sudah dipermudah menggunakan *library-library* sehingga pemula pun bisa mempelajarinya dengan cukup mudah. Papan Arduino Uno menggunakan mikro kontoller Atmega328P. Papan ini memiliki 14 pin *input output* digital (enam diantaranya dapat digunakan untuk *output* PWM), enam buah *input* analog, 16 MHz *crystal oscillator*, sambungan USB, ISCP *header*, dan tombol *reset*. Dalam penggunaannya cukup dengan menghubungkan ke komputer dengan menggunakan kabel USB atau dengan memberikan daya menggunakan adapter AC ke DC atau dengan baterai. Bentuk fisik dari Arduino uno ditunjukkan dalam **Gambar 2.3**



Gambar 2. 3 Arduino Uno

[sumber : Arduino.cc]

Arduino uno memiliki fitur-fitur sebagai berikut:

1. Serial : Pin yang digunakan untuk komunikasi Serial adalah Pin 0 sebagai TX dan pin 1 sebagai RX
2. I^2C : pin yang digunakan untuk komunikasi I^2C adalah pin SDA (A4) dan SDL (A5)
3. Pin Analog : Arduino uno memiliki 6 pin analog yaitu A0, A1, A2, A3, A4 dan A5
4. Pin Digital : Arduino Uno memiliki 14 pin digital yaitu pin 0 sampai pin 13
5. SPI : Pin yang digunakan adalah 10 (SS), 11(MOSI), 12 (MISO), 13 (SCK)
6. PWM : Arduino uno sudah memiliki fitur PWM yaitu menggunakan pin 3, 5, 6, 9, 10, dan 11 dengan format 8 bit
7. *Interrupts external* : pin yang digunakan adalah pin 2 dan pin 3

Spesifikasi Arduino UNO dapat dilihat di **Tabel 2.2**

Tabel 2. 2 Spesifikasi Arduino UNO

[sumber : Arduino.cc]

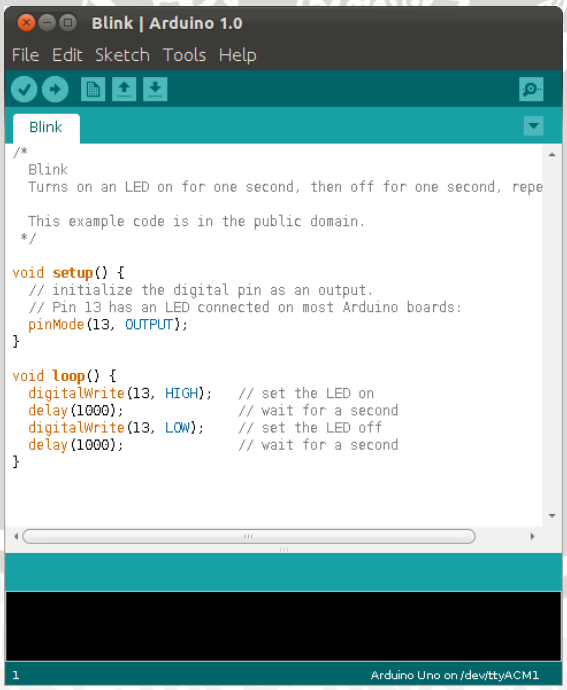
Microcontroller	Atmega 328
Operating Voltage	5V
Input Voltage	7 – 11 V
Digital I/O Pins	14
Analog Input Pins	6
DC Current per I/O Pin	50 Ma
Flash Memory	32KB
SRAM	2 KB

EEPROM	1 KB
Clock Speed	16 MHz (5V model)

Arduino Uno mempunyai beberapa fasilitas untuk dapat berkomunikasi dengan komputer, Arduino lain, atau dengan mikrokontroller lain. Mikrokontroler Atmega328P pada Arduino Uno menyediakan komunikasi UART TTL (5V), yang tersedia pada pin 0 (RX) dan 1 (TX). Papan Arduino Uno menyalurkan komunikasi serial ini melalui USB dan dilihat sebagai *com port* virtual pada software di komputer. Software Arduino *Integrated Development Environment* (IDE) memiliki serial monitor yang memungkinkan data teks sederhana dikirim ke dan dari Arduino (Durfee, 2011)

2.2.4.1 Arduino IDE

Arduino IDE adalah sebuah editor program yang digunakan untuk menulis program, mengcompile, dan mengunggah ke papan Arduino, Arduino *Integrated Development Environment* terdiri dari editor teks untuk menulis kode, area pesan, *toolbar* dengan tombol-tombol untuk fungsi umum, dan sederetan menu. Untuk memasukkan kode program ke papan Arduino, pastikan konfigurasi jenis papan dan *port* yang digunakan pada Arduino IDE sudah benar. Arduino dikatakan sebagai sebuah platform dari *physical computing* yang bersifat *open source*. Tampilan IDE Arduino dapat dilihat pada **Gambar 2.4** berikut.



Gambar 2. 4 Arduino IDE

[sumber : W. Durfee 2011]

Pada Arduino terdapat banyak projek dan alat-alat yang dikembangkan oleh akademisi dan profesional, selain itu juga ada banyak modul-modul pendukung seperti sensor, tampilan dan lain sebagainya. Pada Arduino IDE akan mempermudah dalam mengerjakan berbagai projek dengan bantuan beberapa *tools* dan *library* yang terdapat pada Arduino IDE.

Arduino IDE mempunyai *toolbar* dengan fungsi utama antara lain :

1. Editor Program, sebuah *window* yang memungkinkan pengguna menulis dan mengedit program dalam bahasa *processing*.
2. *Compile*, sebuah modul yang mengubah kode program menjadi biner. Sebuah mikrokontroller tidak akan bisa memahami bahasa *processing*.
3. *Uploader*, sebuah modul yang memuat kode biner dari komputer ke dalam memori didalam papan Arduino. (Durfee, 2011)

2.2.4.2 Pemrograman Arduino

Pada pemrograman Arduino terdapat tiga bagian utama dalam bahasa pemrograman Arduino, yaitu Struktur, *Variable*, dan Fungsi

Struktur dasar dari bahasa pemrograman Arduino itu sederhana dimana hanya terdiri dari dua bagian.

```
1.  void setup ( )
2.  {
3.  //statement
4.  }
5.  Void loop ( )
6.  {
7.  //statement
8.  }
```

Kode Program 2. 1 Kode Struktur Dasar Pemrograman Arduino

[sumber : Brian W. Evans 2007]

Dimana *setup()* bagian untuk inisialisasi yang hanya dijalankan sekali di awal program, sedangkan *loop*() untuk mengeksekusi bagian program yang akan dijalankan berulang-ulang untuk selamanya.

- *setup*()

  Fungsi *setup*() hanya di panggil satu kali ketika program pertama kali di jalankan. Ini digunakan untuk pendefinisian *mode* pin atau memulai komunikasi serial. Fungsi *setup*() harus diikutsertakan dalam program walaupun tidak ada *statement* yang dijalankan.

```
1.  void setup ( )
2.  {
3.  pinMode (13,OUTPUT);
4.  }
```

Kode Program 2. 2 Contoh Kode Method Setup()

[sumber : Brian W. Evans 2007]

- *loop()*

Setelah melakukan fungsi *setup()* maka secara langsung akan melakukan fungsi *loop()* secara berurutan dan melakukan instruksi-instruksi yang ada dalam fungsi *loop()*

```
1.  void loop ( )
2.  {
3.  digitalWrite(13, HIGH);
4.      delay(1000);
5.  }
```

Kode Program 2. 3 Contoh Kode Method Loop()

[sumber : Brian W. Evans 2007]

- *{ } curly braces*

Curly braces mendefinisikan awal dan akhir dari sebuah blok fungsi. Baik blok *void setup(), void loop()* maupun blok fungsi harus diberi tanda kurung kurawal buka "{" sebagai tanda awal program di blok tersebut dan kurung kurawal tutup "}" sebagai tanda akhir program. Tanda kurung kurawal buka dan tutup digunakan pada blok kontrol program, seperti *if, if-else, for-loop, while-loop dan do-while-loop*. Apabila ketika memprogram dan programmer lupa memberi *curly brace* tutup maak ketika di compile akan terdapat laporan error

- *; semicolon*

Semicolon harus di berikan pada setiap statement program yang kita buat, ini merupakan pembatas setiap statement program yang dibuat

- *// line comment*

Sama halnya dengan *block comment*, *line comments* pun sama hanya saja yang dijadikan komen adalah tiap baris

- *Variable*

Variable adalah sebuah penyimpan nilai yang dapat digunakan dalam program. *Variable* dapat dirubah sesuai dengan instruksi yang kita buat.

Ketika mendeklarisikan *Variable* harus diikutsertakan tipe *Variable* serta nilai awal variable

- *Variable scope*

Sebuah variabel dapat dideklarasikan pada awal program sebelum *void setup()*, secara local di dalam sebuah fungsi, dan terkadang di dalam sebuah block statement perulangan. Sebuah variabel global hanya satu dan dapat digunakan pada semua *blok function* dan statement di dalam program. Variabel global dideklarasikan pada awal program sebelum fungsi *setup()*. Sebuah variabel lokal dideklarassikan di setiap block function atau di setiap block statement perulangan dan hanya dapat digunakan pada block yang bersangkutan saja

- *Function*

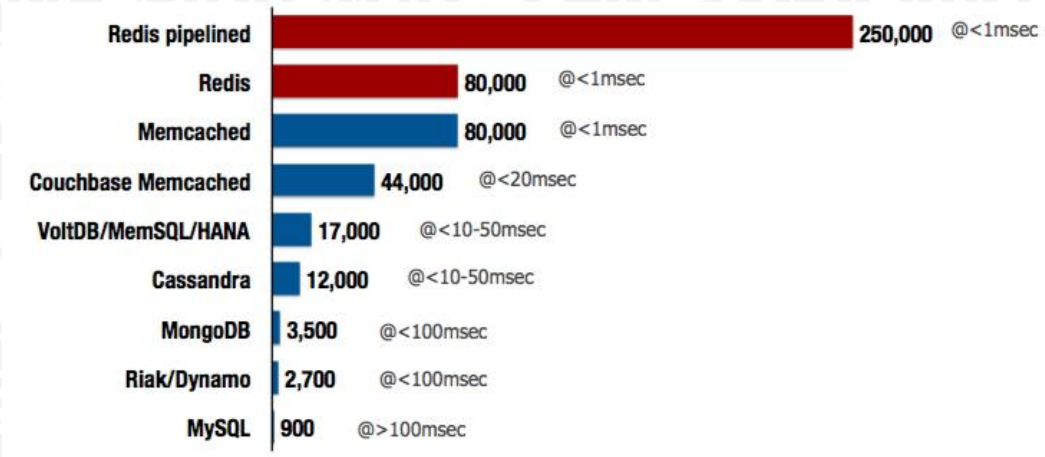
Function (fungsi) adalah blok pemrograman yang mempunyai nama dan mempunyai statement yang akan di eksekusi ketika function di panggil. Cara pendeklarasian function (Smith, 2011)

2.2.5 Redis

Redis adalah sebuah *open source No-sql* yang mempunyai penyimpanan data di dalam memori, Redis sering digunakan sebagai database, *cache*, dan *broker*. Redis mendukung beberapa struktur data yang bisa digunakan antara lain *string, hashes, lists, sets,* dan *sortedsets*.

Redis memiliki konsep dasar yang hampir sama dengan database seperti pada umumnya. Perbedaannya ada pada pengelompokkan data dimana bila dalam database pada umumnya data ditampilkan bersama-sama menjadi satu aplikasi. Sedangkan di dalam Redis database hanya diidentifikasikan dengan sebuah nomor, contohnya jika ingin mengubah sebuah database Redis harus menggunakan sebuah perintah di *command line interface*. Redis mendukung beberapa fitur yang tidak dimiliki database pada umumnya seperti fitur *pub-sub* dan *expire*. Fitur expire memungkinkan data akan hilang secara otomatis bila data telah di set *variable* expirenya.

Dikarenakan Redis mempunyai media penyimpanan di dalam memori, maka dapat dipastikan kemampuan Redis dalam mengeksekusi perintah sangat cepat. Redis mampu menjalankan seratus operasi perintah per detiknya. Maka dari itu Redis sangat cocok untuk sistem yang membutuhkan faktor *real time*. (Miftakhuddin, Suadi & Pratomo, 2011) Kecepatan Redis dapat dilihat dalam **Gambar 2.5** berikut :



Gambar 2. 5 Perbandingan Performa Redis dengan Database Lain

[sumber : Redis.io]

2.2.5.1 *Nosql*

Nosql adalah sebuah konsep mengenai penyimpanan data non-relasional. Teknologi dikembangkan pertama kali oleh Carlo Strozzi pada tahun 1998. Pada kebanyakan situs besar, *Nosql* digunakan bersamaan dengan RDBMS. Hal ini ditujukan untuk mengurangi beban server database. Metode yang digunakan *Nosql* bermacam-macam, tapi ada satu kesamaan yaitu tidak menggunakan struktur tabel yang tetap layaknya relational database. (Miftakhuddin, Suadi & Pratomo, 2011) Metode yang digunakannya, adalah sebagai berikut:

- *Tabel-oriented*, contoh: Google dengan Big Tabel, Facebook dengan Cassandra
- *Graph-oriented*
- *Document-oriented* database, contoh: MongoDB dan CouchDB.
- *Key-value store*, contoh: Redis

2.2.5.2 *Key-Value Store*

Database *Nosql* yang paling sederhana adalah *keyvalue store*. Sebuah *key* dapat menyimpan sebuah *value*, tanpa memperdulikan jenis *value*. Dengan kata lain, tidak ada skema tabel yang tetap, tetapi aplikasi client harus mendefinisikan semantik untuk mengetahui jenis data tersebut. Bentuk dari *key-value store* berupa *ordered tuple* yang berisi *key* dan *value*. Nama lain dari *key-value* ini adalah *associative array, map,* atau *dictionary*. Ketika menggunakan *key-value store*, sepasang data, *key* dan *value*, dimasukkan ke database berupa *dictionary* dan *value* dapat diambil kembali berdasarkan *key* yang dimiliki.

Penggunaan *Key-value store* sangat berguna ketika pengaksesan data dilakukan menggunakan sebuah *key*. Pengaksesan data menggunakan *key* sudah cukup umum digunakan. Pada beberapa bagian aplikasi seperti *user profiles, user sessions, shopping carts*, dan lain-lain, data/*value* disimpan dalam sebuah data *store* sehingga mudah ditangani. (Miftakhuddin, Suadi & Pratomo, 2011)

2.2.5.3 Struktur Data Redis

Redis memiliki keunggulan dalam penglompokkan struktur data yang tidak dimiliki oleh sistem database lainnya. Redis memiliki lima struktur data yaitu *String, Hashes, Lists, Sets dan Sorted sets*.

- *Strings*

String adalah struktur data yang paling sederhana dalam Redis. Struktur data *String* biasanya digunakan untuk mencocokkan antara *key* dengan *value* nya. Adapaun operasi perintah yang bisa digunakan untuk struktur data *String*, antara lain

- Set <*key*> <*value*>, digunakan untuk membuat struktur data *String*.
- Get <*key*>, digunakan untuk mendapatkan *value* paling akhir dari *key*
- Strlen <*key*>, digunakan untuk mendapatkan panjang *value* sesuai dengan *key*.
- Getrange <*key*> <start> <end>, digunakan untuk mendapatkan *value* dari *key* sesuai dengan jarak yang ditentukan.
- Append <*key*> <*value*>, digunakan untuk menambahkan *value* sesuai dengan *key* yang sudah dibuat sebelumnya.

- *Hashes*

Hashes mempunyai konsep hampir sama dengan struktur data *String*, perbedaannya terletak pada tambahan level *key*. Bila *string* mempunyai satu *key*, *hashes* memiliki dua level *key*. Adapun operasi perintah yang bisa digunakan untuk struktur data *hashes*, antara lain

- Hset <index> <*key*> <*value*>, digunakan untuk membuat struktur data *hashes*
- Hget <index> <*key*>, digunakan untuk mendapatkan *value* dari *key* yang telah ditentukan
- Hmset <index> <*key*> <*value*> <*key*> <*value*>, digunakan untuk membuat *hashes* dengan *key* lebih dari satu
- Hmset <index> <*key*> <*key*>, digunakan untuk mendapatkan *value* dari *hashes* yang memiliki *key* lebih dari satu.
- Hgetall <index>, digunakan untuk mendapatkan semua *value*.

- *Lists*

Struktur data *lists* memungkinkan untuk menyimpan *value* dalam bentuk array. Dengan menggunakan *list* pengguna dapat mendapatkan *value* secara lebih spesifik berdasarkan letaknya. Adapun perintah yang bisa digunakan dalam *lists*

- Lpush <*key*> <*value*>, digunakan untuk membuat struktur data *list*.

- Lrange <*key*> <start> <end>, digunakan untuk mendapatkan *value* dari *key* dengan jarak yang ditentukan.

➢ *Sets*

Konsep struktur data *sets* adalah menyimpan *value* dengan unik, sehingga menghindari adanya duplikat *value* yang sama. Adapun perintah yang bisa digunakan dalam *sets*

- Sadd <*key*> <*value*>, digunakan untuk membuat struktur data *sets*.
- Smembers <*key*s>, digunakan untuk mendapatkan semua *value* yang terdapat pada *key*.

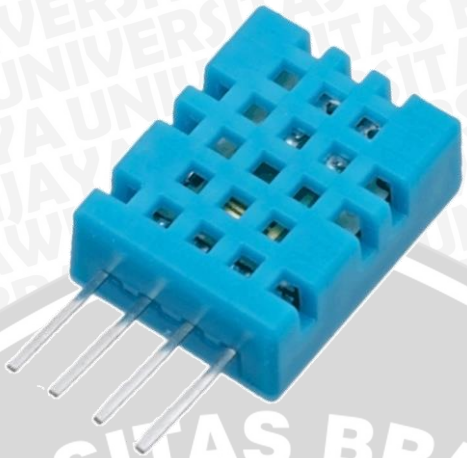
➢ *Sorted sets*

Sorted sets mempunyai konsep yang hampir sama dengan struktur data *sets*. Hanya saja *sorted sets* mempunyai variabel score, dimana score tersebut dapat diurutkan sesuai dengan keinginan. Adapun perintah yang bisa digunakan dalam *sorted sets*

- Zadd <*key*> <*value*> <score>, digunakan untuk membuat struktur data *sorted sets*.
- Zrange <*key*> <start> <end>, digunakan untuk mendapatkan *value* sesuai dengan jarak yang ditentukan.
- Zrange <*key*> <start> <end> withscores, digunakan untuk mendapatkan *value* dan score sesuai dengan jarak yang ditentukan dan hasilnya di urutkan berdasarkan *value* yang mempunyai score paling rendah terlebih dahulu. (Seguin, 2011)

2.2.6 Sensor DHT11

Sensor DHT 11 merupakan sensor suhu dan kelembapan udara yang memiliki jangkauan pengukuran suhu antara 0-50^{o}C dan jangkauan pengukuran kelembapan udara 20-95% RH. Dengan ukuran yang kecil dan memakan daya *list*rik yang kecil modul sensor ini mudah untuk diimplementasikan menggunakan mikrokontroller. DHT 11 cukup ekonomis namun memadai untuk aplikasi monitoring suhu dan kelembapan udara. Bentuk fisik dari DHT 11 dapat dilihat pada **Gambar 2.6**.

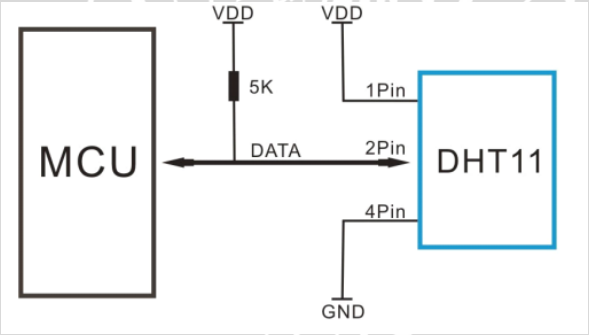


Gambar 2. 6 Bentuk Fisik DHT 11

[sumber : D-Robotics UK 2010]

2.2.6.1 Karakteristik DHT11

Sensor DHT 11 memiliki fitur pengukuran suhu dan kelembapan udara yang sudah dikalibrasi dengan keluaran sinyal digital. Sensor DHT11 memiliki komponen 4 pin namun yang digunakan hanya 3 yaitu *pin ground*, data, dan 5v. **Gambar 2.7** dijelaskan mengenai konfigurasi pin sensor DHT 11 serta **Tabel 2.3** menjelaskan tentang spesifikasi sensor DHT11



Gambar 2. 7 Gambaran Konfigurasi DHT 11

[sumber : D-Robotics UK 2010]

Tabel 2. 3 Spesifikasi Sensor DHT 11

[sumber : D-Robotics UK 2010]

Voltase	5 V
Temperature Range	0-50 °C
Humidity Range	20-90% RH
Interface	Digital

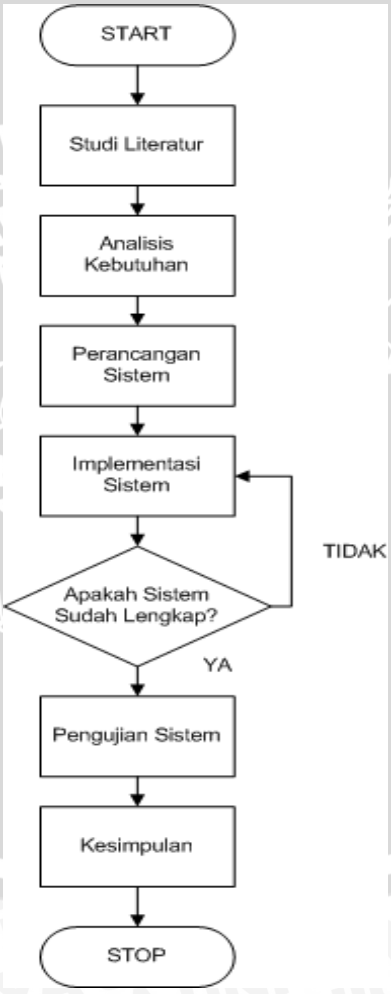
BAB 3 METODOLOGI

3.1 Jenis Penelitian

Penelitian yang dilakukan merupakan penelitian implementatif dan pengembangan. Penelitian jenis ini merupakan sebuah kegiatan penelitian dalam rangka membuat sebuah produk perangkat lunak dengan sistematis. Penelitian dimulai dari analisis kebutuhan, perancangan perangkat keras, perancangan perangkat lunak berdasarkan analisis kebutuhan, konstruksi atau pembuatan perangkat lunak, serta pengujian hasil menjadi produk perangkat lunak.

3.2 Metodologi Penelitian

Metodologi penelitian yang akan dilakukan pada penelitian ini secara umum ditunjukkan pada **Gambar 3.1** di bawah ini.



Gambar 3. 1 Metodologi Penelitian

3.2.1 Studi literatur

Studi literatur dilakukan bertujuan untuk mempelajari serta memahami konsep-konsep sistem agar ketika dilakukan perancangan tidak terlalu mengalami kendala. Jenis literatur yang digunakan adalah artikel jurnal. Adapun yang perlu menjadi bahan studi literatur pada penelitian ini meliputi :

a. Websocket
   - *Twisted framework*

b. HTTP

c. Arduino

d. Redis
   - *Lists*
   - *Sorted sets*

e. DHT 11

3.2.2 Analisis Kebutuhan

Pada tahap ini merupakan tahap menganalisa kebutuhan sistem yang akan digunakan. Aplikasi ini menggunakan dua komponen yaitu komponen hardware dan komponen software. Komponen hardware meliputi

a. Perangkat sensor
   - Arduino Uno
   - DHT 11
   - Lampu *LED*

b. Komputer / Laptop digunakan sebagai server sistem

Perangkat sensor diharapkan mampu digunakan sebagai actuator dan dapat mengirimkan data yang diperoleh dari sensor ke server. Kemudian server diharapkan mampu mengirimkan data ke client melalui koneksi Websocket. Selanjutnya komponen software meliputi

1. Program Perangkat Sensor

   Program perangkat sensor harus dapat mengeksekusi perintah dan dapat membaca serta mengirimkan data yang didapat dari sensor. Program perangkat sensor dituliskan dalam bahasa C dan dijalankan di Arduino IDE

2. Program server sistem

   Program server harus dapat menjadi penjembatan antara perangkat sensor dan client. Selain sebagai lalu lintas komunikasi, server juga diharapkan dapat mengatur database dalam bentuk Redis. Program server sistem dituliskan dalam bahasa python

3. Aplikasi web client

Aplikasi web client harus mampu melakukan koneksi dengan server menggunakan Websocket, serta mampu menerima dan mengirimkan data ke server. Bahasa yang digunakan adalah *javascript* dan *html*, dimana nantinya aplikasi bisa dijalankan pada *browser*.

4. Database Redis

Database Redis digunakan untuk menyimpan semua hasil data yang siap dikirimkan ke client atau ke mikrokontroler Arduino.

3.2.3 Perancangan

Perancangan dilakukan agar penelitian dapat berjalan dengan baik dan sistematis. Perancangan bertujuan untuk memberikan gambaran mengenai implementasi perangkat keras dan perangkat lunak dari penelitian yang akan dilakukan. Pada tahap ini dilakukan dua perancangan yaitu perancangan untuk mengontrol lampu *LED* dan perancangan untuk memonitor suhu dan kelembapan dari sensor DHT 11. Masing-masing perancangan meliputi perancangan perangkat keras, perancangan database, perancangan pengiriman pesan baik itu antara server dengan client maupun server dengan Arduino, perancangan pengolahan pesan, perancangan aplikasi web client, serta perancangan hasil akhir dari kedua implementasi .

Pada perancangan kontroling LED meliputi, perancangan perangkat keras akan dibuat gambar rancangan rangkaian antara perangkat lampu *LED* dengan mikrokontroler Arduino. Perancangan ini dibuat agar perangkat yang digunakan terintegrasi dengan Arduino Uno. Perancangan pengiriman pesan dari client ke server bertujuan untuk menjelaskan cara client mengirimkan pesan ke server menggunakan Websocket. Perancangan database Redis dibuat untuk menjelaskan struktur penyimpanan data yang diatur oleh server. Perancangan pengolahan pesan bertujuan untuk menjelaskan proses pembuatan pesan untuk dikirimkan ke Arduino. Perancangan aplikasi web client menjelaskan bagaimana client dapat berkomunikasi dengan server melewati *browser*. Perancangan eksekusi perintah oleh Arduino bertujuan untuk menjelaskan proses akhir dari implementasi yaitu bagaimana Arduino dapat mengkontrol lampu LED.

Pada perancangan monitoring suhu dan kelembapan meliputi, perancangan perangkat keras akan dibuat gambar rancangan rangkaian antara sensor DHT 11 dengan Arduino. Perancangan ini dibuat agar perangkat yang digunakan terintegrasi dengan Arduino Uno. Perancangan pengiriman data dari Arduino ke server bertujuan untuk menjelaskan proses pengambilan data suhu dan kelembapan oleh DHT 11 sampai data tersebut dikirimkan ke server. Perancangan database Redis dibuat untuk menjelaskan struktur penyimpanan data yang diatur oleh server. Perancangan pengolahan pesan bertujuan untuk menjelaskan proses pembuatan pesan untuk dikirimkan ke client. Perancangan aplikasi web client menjelaskan bagaimana client dapat berkomunikasi dengan server melewati

browser. Perancangan hasil monitoring pada aplikasi web bertujuan untuk menjelaskan proses akhir dari implementasi yaitu bagaimana aplikasi web menampilkan semua data yang didapat dari server.

3.2.4 Implementasi

Implementasi sistem dilaksanakan berdasarkan perancangan yang telah dibuat sebeblumnya. Untuk melakukan implementasi sistem, ada beberapa tahapan yaitu:

A. Implementasi Kontroling Lampu LED

1. Implementasi perangkat keras. Pada tahap ini akan dilakukan konfigurasi perangkat lampu *LED* dengan mikrokontroler Arduino Uno. Konfigurasi ini dilakukan untuk memastikan bahwa semua perangkat dapat terintegrasi dengan mikrokontroler Arduino Uno.
2. Implementasi database Redis. Tahap ini adalah membuat sebuah database yang tersimpan di dalam memori, dimana database menyimpan data-data yang didapatkan dari hasil komunikasi server dengan client.
3. Implementasi aplikasi web client.Aplikasi web client digunakan sebagai antar muka bagi pengguna untuk menjalankan sistem. Aplikasi web ini digunakan supaya mempermudah pengguna untuk mengontrol perangkat LED yang terpasang di Arduino Uno.
4. Implementasi pengiriman pesan dari client ke server. Untuk menjalankan fungsi untuk mengontrol LED maka diperlukan sebuah komunikasi antara client dengan server. Tujuannya supaya client tidak perlu melakukan komunikasi langsung dengan Arduino melainkan cukup dengan server saja.
5. Implementasi pengolahan pesan. Supaya Arduino dapat melakukan kontroling lampu LED diperlukan sebuah pesan dari server. Pengolahan pesan ini bertujuan untuk membuat sebuah pesan yang nantinya akan dikirmkan ke Arduino.
6. Implementasi eksekusi perintah oleh Arduino. Implementasi ini merupakan tahap akhir dari pengimplementasian kontroling LED. Setelah pesan dikirimkan oleh server dan diterima oleh Arduino, maka Arduino akan langsung menjalankan perintah sesuai dengan pesan yang diterima.

B. Implementasi Monitoring Suhu dan Kelembapan

1. Implementasi perangkat keras. Pada tahap ini akan dilakukan konfigurasi perangkat sensor DHT 11 dengan mikrokontroler Arduino Uno. Konfigurasi ini dilakukan untuk memastikan bahwa semua perangkat dapat terintegrasi dengan mikrokontroler Arduino Uno.

2. Implementasi database Redis. Tahap ini adalah membuat sebuah database yang tersimpan di dalam memori, dimana database menyimpan data-data yang didapatkan dari hasil komunikasi server dengan client.

3 Implementasi aplikasi web client.Aplikasi web client digunakan sebagai antar muka bagi pengguna untuk menjalankan sistem. Aplikasi web ini digunakan supaya mempermudah pengguna untuk memonitoring nilai suhu dan kelembapan yang didapatkan oleh sensor DHT 11.

4 Implementasi pengiriman pesan dari Arduino ke server. Dalam implementasi ini terdapat proses pengambilan data suhu dan kelembapan oleh sensor DHT 11. Setelah data didapatkan, data tersebut akan langsung dikirimkan ke server.

5 Implementasi pengolahan pesan. Supaya aplikasi web dapat menampilkan informasi yang didapatkan oleh sensor, perlu dilakukan pengolahan pesan. Pengolahan pesan bertujuan untuk membuat pesan yang nantinya dikirimkan ke client.

6 Implementasi hasil monitoring pada aplikasi web. Implementasi ini merupakan hasil akhir dari pengimplementasian monitoring suhu dan kelembapan. Hasil akhir dari pengimplementasian ini adalah menampilkan semua data yang didapatkan oleh sensor DHT 11, sehingga pengguna dapat membaca informasi dengan mudah.

3.2.5 Pengujian Sistem dan Analisa Hasil Pengujian

Pengujian sistem dilakukan untuk mengetahui keberhasilan dan kekurangan yang dimiliki oleh sistem yang telah dibuat. Pengujian sistem juga dilakukan untuk mengetahui tingkat keandalan sistem dalam mengirimkan data ke client, sehingga dapat diketahui juga keberhasilan pengiriman data dari sumber ke tujuan. Sementara analisa hasil pengujian dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan tujuan perancangan sistem tersebut. Terdapat dua jenis pengujian yaitu pengujian keandalan sistem dalam menangani request per edetiknya, dan pengujian Round Trip Time yang diberikan oleh sistem dalam menangani request. Pengujian ini dilakukan dengan beberapa skenario.

3.2.6 Pengambilan Kesimpulan dan Saran

Kesimpulan diambil berdasarkan semua tahapan yang telah dilalui dalam penelitian sehingga menghasilkan output yang sesuai dengan tujuan penelitian dan harapan peneliti. Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem telah selesai dilakukan serta didasarkan pada kesesuaian antara teori dan praktik. Kesimpulan diambil untuk menjawab rumusan permasalahan yang telah ditetapkan sebelumnya. Tahap terakhir adalah penulisan saran yang dimaksudkan untuk memperbaiki kekurangan-kekurangan yang terjadi dan menyempurnakan penulisan serta dapat dijadikan pertimbangan atas penelitian berikutnya.

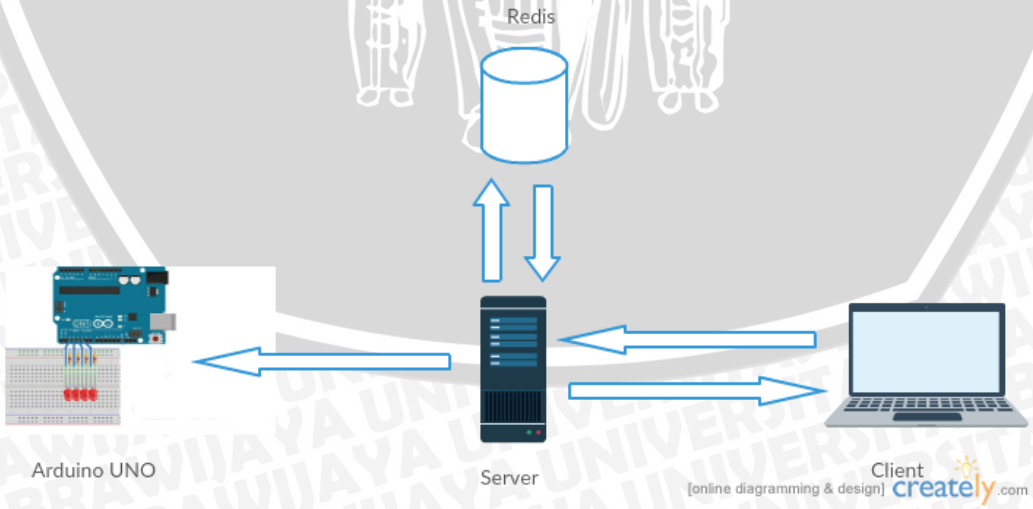
BAB 4 PERANCANGAN

Perancangan dilakukan agar implementasi dapat berjalan dengan baik dan sistematis. Perancangan menggambarkan bagaimana keterlibatan serta peranan antar elemen dalam sistem. Perancangan dibagi menjadi dua bagian yaitu perancangan untuk mengontrol lampu *LED* dan perancangan untuk memonitoring suhu dan kelembapan dari sensor DHT 11. Masing-masing perancangan terdiri dari perancangan perangkat keras, perancangan komunikasi server dengan Arduino Uno, perancangan database Redis, perancangan komunikasi server dengan client menggunakan protokol Websocket, perancangan aplikasi web client, serta perancangan keseluruhan.

4.1 Perancangan Kontroling Lampu *LED*

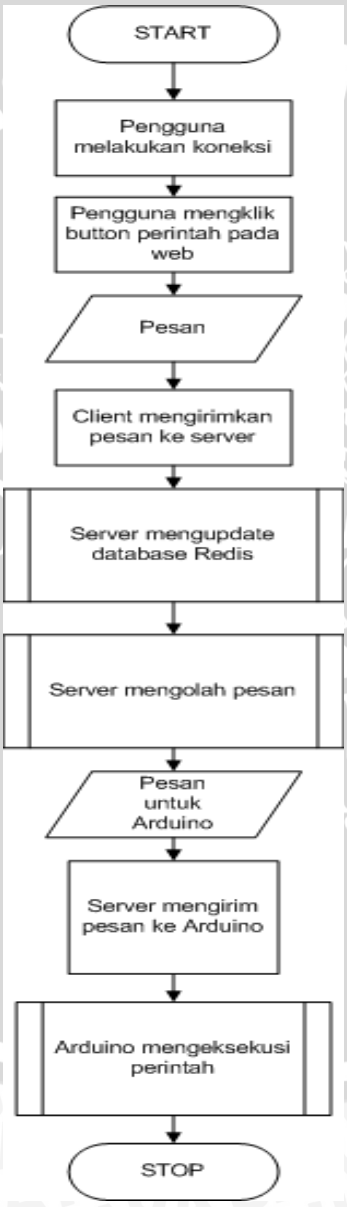
Perancangan untuk Mengontrol Lampu *LED* merupakan langkah awal untuk mengimplementasikan Websocket untuk mengontrol perangkat Arduino. Perancangan dilakukan secara berurutan supaya pada saat pengimplementasian berjalan dengan baik dan sistematis. Adapun perancangan yang dilakukan secara berurutan yaitu

1. Perancangan perangkat keras.
2. Perancangan database Redis.
3. Perancangan aplikasi web client.
4. Perancangan pengiriman pesan dari client ke server.
5. Perancangan proses pengolahan pesan.
6. Perancangan eksekusi perintah oleh Arduino .



Gambar 4. 1 Rancangan Arsitektur Implementasi Kontroling Lampu LED

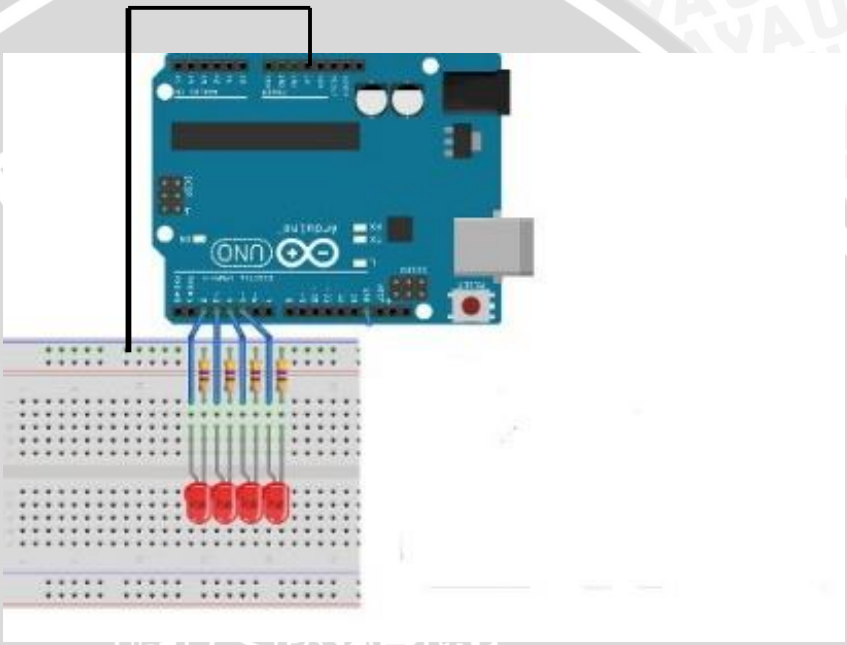
Gambar 4.1 menjelaskan secara umum arsitektur sistem yang akan dirancang. Terdapat 4 entitas yaitu Arduino Uno, server, database Redis, dan client. Dalam rancangan umum topologi terdapat 3 jenis komunikasi. Yang pertama adalah komunikasi server dengan Arduino Uno, komunikasi ini digunakan untuk mengirim pesan dari client kepada Arduino Uno. Kedua adalah komunikasi server dengan database server, komunikasi ini meliputi komunikasi untuk mendapatkan data yang tersimpan dalam Redis dan komunikasi untuk mengubah data yang terdapat dalam Redis. Dan yang ketiga dalah komunikasi antara server dengan client yang digunakan untuk mengirim dan menerima pesan baik dari server maupun client. **Gambar 4.2** menunjukkan diagram alur sistem untuk mengontrol lampu LED secara keseluruhan



Gambar 4. 2 Diagram Alur Sistem Kontroling Lampu LED Secara Keseluruhan

4.1.1 Perancangan Perangkat Keras

Perancangan perangkat keras menjelaskan bagaimana memasangkan perangkat ke mikrokontroler Arduino Uno. Pada penelitian ini diperlukan empat lampu *LED* yang mempunyai warna lampu yang berbeda-beda antara lain warna merah, putih hijau dan biru. Untuk menghubungkan lampu *LED* ke mikrokontroler Arduino. Diperlukan adanya resistor dan kabel jumper, yang berguna sebagai penyalur daya dari Arduino Uno ke lampu *LED*. Konfigurasi rancangan lampu *LED* dengan Arduino Uno adalah sebagai berikut :



Gambar 4. 3 Rancangan Konfigurasi Lampu LED pada Arduino UNO

Terlihat pada **Gambar 4.3**, empat lampu ditancapkan ke dalam board. Lalu dengan bantuan kabel jumper *LED* disambungkan ke digital IN/OUT dengan masing-masing lampu *LED* disambungkan ke pin 2, pin 3, pin 4, dan pin 5. Untuk mendapatkan daya supaya lampu *LED* dapat menyala dibutuhkan kabel jumper yang terhubung dari board dengan pin Power OUT yang dimiliki Arduino Uno. Dalam **Gambar 4.3**, terlihat kabel jumper dihubungkan pada power 5V. Setelah itu untuk mendapatkan daya *list*rik maka perlu menyambungkan Arduino Uno ke laptop menggunakan kabel USB.

4.1.2 Perancangan Database Redis

Sebuah database dibutuhkan dalam pengimplementasian sebagai media penyimpanan status empat lampu *LED*. Dimana status lampu diidentifikasikan dengan angka 1 dan 0. Angka 1 menunjukkan lampu *LED* tersebut dalam kondisi menyala dan angka 0 menunjukkan lampu *LED* tersebut dalam kondisi mati.

Redis sendiri memiliki lima struktur data dengan fungsinya yang berbeda-beda yaitu *string*, *sets*, *hashes*, *lists*, dan *sorted sets*. Pada perancangan database

Redis untuk mengontrol lampu *LED* menggunakan struktur data *sorted sets*. *Sorted sets* dipilih dalam perancangan untuk mengontrol lampu *LED* yang pertama karena *sorted sets* tidak memperbolehkan adanya elemen yang terduplikat, hal ini cocok dengan kebutuhan sistem dimana masing-masing empat lampu *LED* diidentifikasikan oleh satu elemen saja. Kedua dalam *sorted sets* terdapat *field* score, dimana score digunakan dalam sistem ini untuk menyimpan status lampu *LED* dalam bentuk *integer*. Ketiga dalam *sorted sets* terdapat fungsi pengurutan sehingga membuat posisi elemen menjadi statis.

Secara sederhana struktur data database dalam perancangan ini digambarkan oleh **Tabel 4.1** berikut

Tabel 4. 1 Rancangan Database LED Redis

Key	Elemen	Score
LED	*LED* 2	1 atau 0
LED	*LED* 3	1 atau 0
LED	*LED* 4	1 atau 0
LED	*LED* 5	1 atau 0

4.1.3 Perancangan Aplikasi Web Client

Perancangan aplikasi web client ini digunakan sebagai *interface* sistem. Perancangan web ini adalah hal yang sangat penting dalam perancangan sistem secara keseluruhan, karena aplikasi web in adalah bagian *interface* yang akan paling banyak digunakan oleh pengguna untuk menjalankan fungsi-fungsi dari sistem ini. **Gambar 4.4** menggambarkan rancangan aplikasi web client yang digunakan untuk mengontrol lampu LED.

Program aplikasi web berfungsi untuk mengontrol semua lampu *LED* yang terpasang pada Arduino Uno. Sistem ini dituliskan dalam bahasa *html* dan *javascript*. Selain itu program aplikasi web ini juga harus memiliki fitur-fitur sebagai berikut :

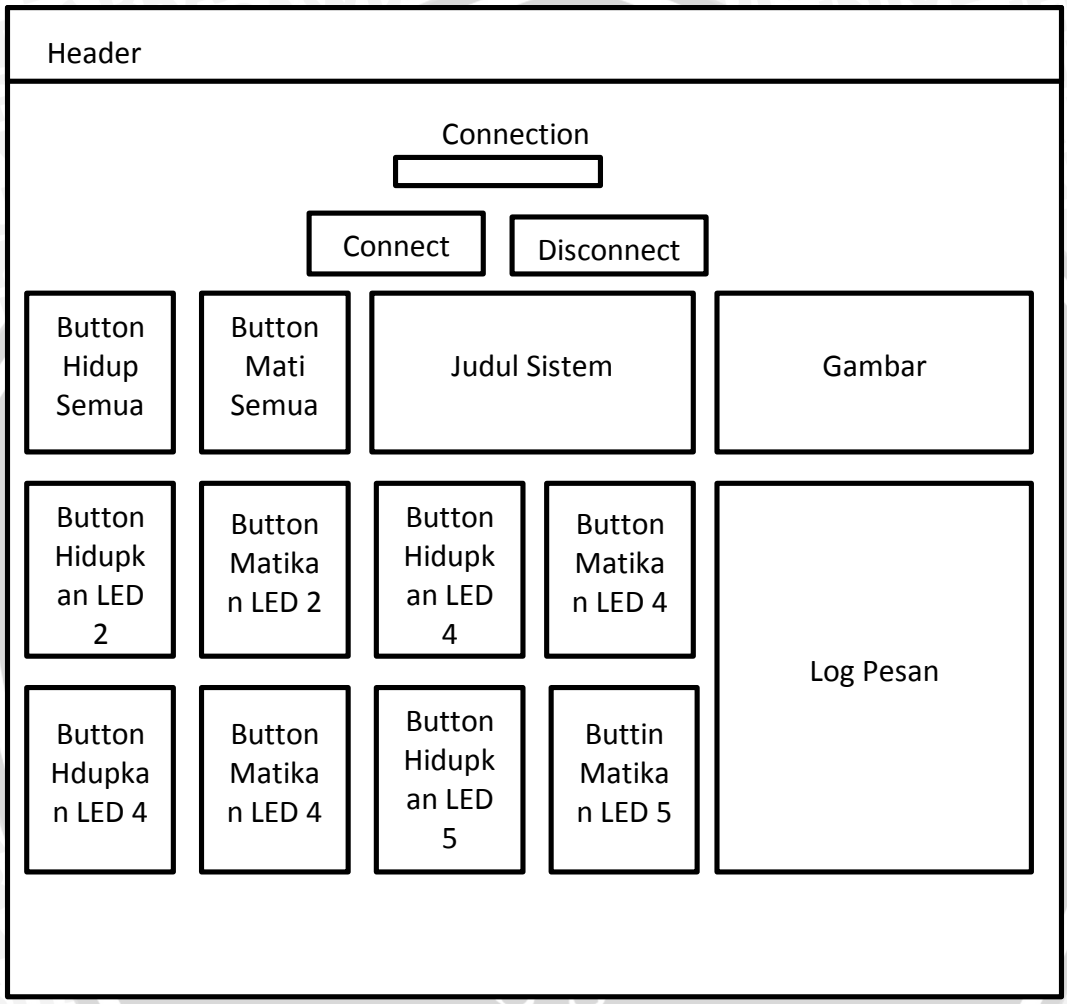
1. Sistem harus mampu untuk melakukan koneksi ke server menggunakan Websocket.
2. Sistem harus mampu untuk melakukan pemutusan koneksi ke server.
3. Sistem harus mampu mengirimkan pesan ke server untuk menghidupkan semua lampu *LED*.
4. Sistem harus mampu mengirimkan pesan ke server untuk mematikan semua lampu *LED*.
5. Sistem harus mampu mengirimkan pesan ke server untuk menghidupkan lampu *LED* yang berada di pin 2.
6. Sistem harus mampu mengirimkan pesan ke server untuk menghidupkan lampu *LED* yang berada di pin 3.

7. Sistem harus mampu mengirimkan pesan ke server untuk menghidupkan lampu *LED* yang berada di pin 4.
8. Sistem harus mampu mengirimkan pesan ke server untuk menghidupkan lampu *LED* yang berada di pin 5.
9. Sistem harus mampu mengirimkan pesan ke server untuk mematikan lampu *LED* yang berada di pin 2.
10. Sistem harus mampu mengirimkan pesan ke server untuk mematikan lampu *LED* yang berada di pin 3.
11. Sistem harus mampu mengirimkan pesan ke server untuk mematikan lampu *LED* yang berada di pin 4.
12. Sistem harus mampu mengirimkan pesan ke server untuk mematikan lampu *LED* yang berada di pin 5.
13. Sistem harus mampu menampilkan pesan yang dikirimkan oleh server ke client.

Setelah fungsi-fungsi didefinisikan, maka perlu dibuat sebuah *layout* untuk gambaran *interface* web seperti pada gambar.Adapun elemen-elemen yang terkandung pada *layout* yang memiliki fungsi masing-masing yaitu :

1. *Field* Connection : Digunakan untuk menuliskan alamat Websocket yang digunakan oleh server. Contohnya ws://ip_address:*port.*
2. Button Connect : Digunakan untuk melakukan koneksi dengan server dengan alamat yang sudah dituliskan pada *field* connection.
3. Button Disconnect : Digunakan untuk melakukan pemutusan koneksi dengan server.
4. *Field* Log Pesan : Digunakan untuk menampilkan pesan dari server.
5. Button Hidup Semua : Digunakan untuk mengirimkan pesan untuk menghidupkan semua *LED* kepada server.
6. Button Mati Semua : Digunakan untuk mengirimkan pesan untuk mematiakn semua *LED* kepada server.
7. Button Hidupkan *LED* 2 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 2 kepada server.
8. Button Matikan *LED* 2 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 2 kepada server.
9. Button Hidupkan *LED* 3 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 3 kepada server.
10. Button Matikan *LED* 3 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 3 kepada server.
11. Button Hidupkan *LED* 4 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 4 kepada server.

12. Button Matikan *LED* 4 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 4 kepada server.
13. Button Hidupkan *LED* 5 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 5 kepada server.
14. Button Matikan *LED* 5 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 5 kepada server.



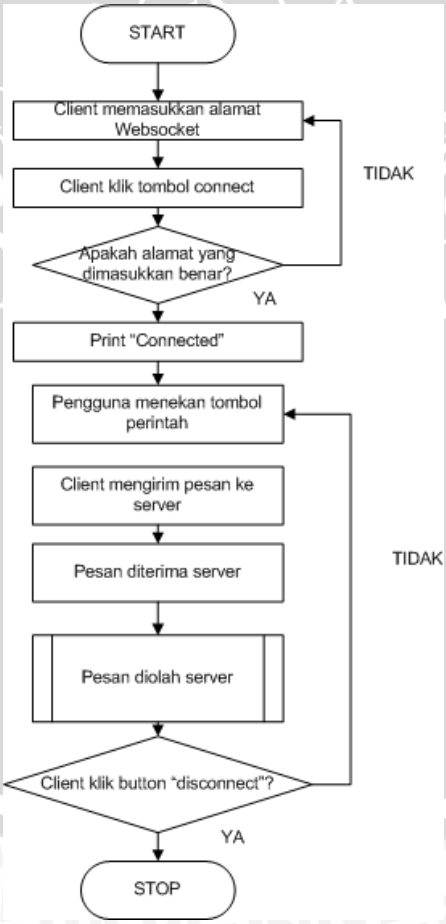
Gambar 4. 4 Rancangan Aplikasi Web Client untuk Implementasi Kontroling LED

Aplikasi web client dituliskan dalam bahasa *html* dan *javascript*. Untuk mendukung komunikasi Webcocket dibutuhkan juga sebuah client yang mampu membuat koneksi Websocket dengan server. Maka dari itu dalam studi kasus ini dalam perancangan aplikasi web client, web client membutuhkan sebuah Websocket API. Umumnya Websocket API pada web client memiliki 4 fungsi utama antara lain onOpen adalah sebuah fungsi yang digunakan untuk menbuat koneksi dengan server. OnClose adalah sebuah fungsi yang digunakan untuk menutup koneksi dengan server. OnMessage adalah sebuah fungsi untuk menerima pesan dari server dan mengirimkan pesan kepada server. Dan OnEror adalah sebuah fungsi untuk menampilkan pesan error saat client gagal melakukan koneksi dengan server.

4.1.4 Perancangan Pengiriman Pesan Dari Client ke Server

Untuk melakukan semua fungsi-fungsi yang sudah dijelaskan pada perancangan aplikasi web, diperlukan adanya komunikasi antara server dengan client. Dengan menggunakan protokol Websocket diharapkan sistem dapat terintegrasi antara server dengan client. **Gambar 4.5** menampilkan diagram alur proses pengiriman pesan dari client ke sever.

Dalam studi kasus ini untuk mengimplementasikan sebuah komunikasi Websocket dibutuhkan sebuah server yang mampu membuat koneksi jaringan komunikasi dengan client. Salah satu *library* python yang mendukung komunikasi Websocket antara server python dengan web client adalah *library twisted* txws. Umumnya *library* txws mempunyai 3 fungsi utama antara lain. Fungsi connectionMade digunakan untuk menerima koneksi dari client. connectionLost digunakan untuk memutuskan koneksi dengan client. Dan fungsi datareceived digunakan untuk menerima data yang dikirimkan oleh client dan mengirimkan data kepada client. Dengan adanya persamaan fungsi *library* txws dengan Websocket API client, dalam studi kasus ini digunakan sebuah *library* python txws.



Gambar 4. 5 Diagram Alur Proses Pengiriman Pesan dari Client ke Server

Gambar 4.5 merupakan rancangan proses pengiriman pesan dari client menuju server. Untuk mengirim pesan kepada server, dibutuhkan sebuah koneksi antara client dengan server. Dengan menggunakan *library* Websocket pada web client dan *library* txws pada server, maka client perlu measukkan alamat websocket yang telah disepakati oleh server. Diagram alur di atas dapat dijelaskan secara rinci.

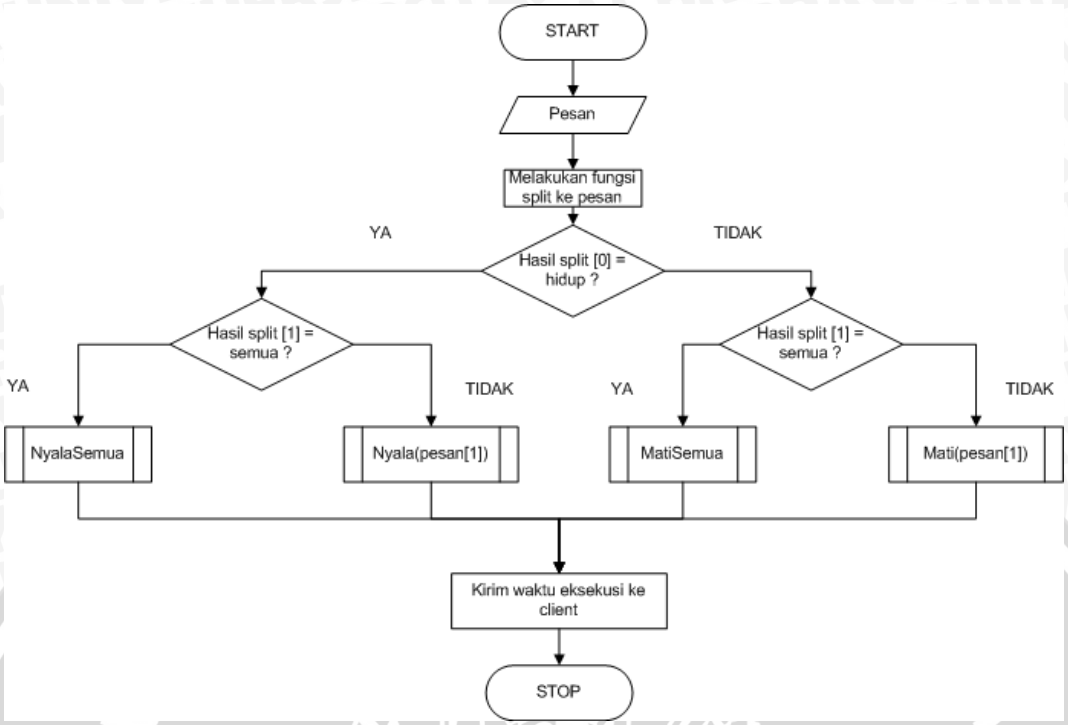
1. Pertama pengguna memasukkan alamat yang telah disepakati dengan server. Pengguna dapat memasukkan alamat tersebut pada *field* yang tersedia pada sistem sesuai dengan perancangan aplikasi web. Adapun format alamt yang dipakai adalah ws://(ip_address server):(*port*)
2. Pengguna mengklik button “Connect”
3. Jika alamat yang dimasukkan benar maka sistem akan mencetak pesan “CONNECTED” untuk menandakan client berhasil tersambung dengan server. Sebaliknya jika alamat yang dimasukkan salah maka sistem akan mencetak “ERROR” pada web.
4. Setelah client berhasil melakukan koneksi dengan server, pengguna diberi pilihan untuk menjalankan fungsi-fungsi yang ada dalam web sesuai dengan fungsi yang dijelaskan pada perancangan aplikasi web client.
5. Client akan mengirim pesan ke server sesuai dengan button apa yang diklik oleh pengguna.
6. Setelah itu pesan yang diterima server akan diolah oleh server. Dimana dalam proses ini terdapat proses penyeleksian pesan, mengubah status *LED* dalam Redis, dan pengiriman pesan yang telah diolah ke Arduino.

Setelah pesan berhasil diterima oleh server, server melakukan pengolahan pesan. Pengolahan pesan ini bertujuan untuk mengubah pesan menjadi sebuah *integer*, dimana *integer* tersebut akan dikirimkan ke Arduino Uno.

4.1.5 Perancangan Pengolahan Pesan

Dikarenakan Arduino hanya bisa membaca data dalam bentuk *byte* dalam komunikasi *Serial*, maka diperlukan pengolahan pesan yang dilakukan server. Selain itu pengolahan pesan juga digunakan untuk menyederhanakan pesan, yang semula berbentuk *string* akan diubah menjadi *integer*. Dengan begitu pesan yang akan dikirimkan ke Arduino berbentuk *integer* yang memiliki 1 *byte* atau 2 *byte*. Dalam proses pengolahan pesan yang dilakukan oleh server terdapat beberapa langkah antara lain :

1. Proses penyeleksian pesan.
2. Proses pembaruan data dalam database Redis
3. Proses membuat pesan untuk Arduino



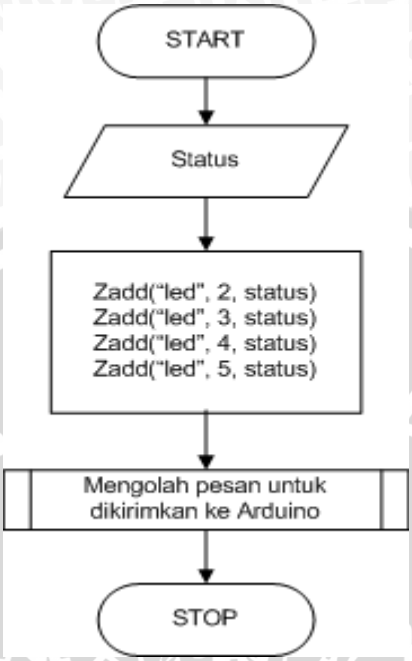
Gambar 4. 6 Diagram Alur Proses Penyeleksian Pesan

Dari gambar 4.6 ditunjukkan diagram alur proses penyeleksian pesan. Dalam diagram alur di atas dapat dijelaskan bahwa proses komunikasi antara client dengan server adalah sebagai berikut

1. Dari pesan yang telah dikirim oleh client, server akan melakukan fungsi split pada pesan tersebut. Fungsi split ini bertujuan untuk menyeleksi proses yang akan dikerjakan oleh server. Contoh proses split pesan, server menerima pesan “hidup semua”. Setelah itu pesan “hidup semua” akan di split, dimana hasil split tersebut adalah split[0] adalah “hidup” dan split[1] adalah “semua”
2. Setelah proses split, dilakukan penyeleksian proses mana yang harus dikerjakan oleh server.
3. Setalah server selesai melakukan proses tersebut, server akan mengirimkan waktu. Waktu tersebut dikirimkan ke client dengan tujuan untuk memberi informasi kepada pengguna kapan pengguna memberi perintah untuk menyalakan atau mematikan lampu *LED*.

Di dalam proses pengolahan pesan terdapat sub proses. Sub proses tersebut dikerjakan oleh server setelah server menyeleksi pesan. Terdapat empat sub proses diantaranya adalah proses NyalaSemua digunakan untuk menyalakan

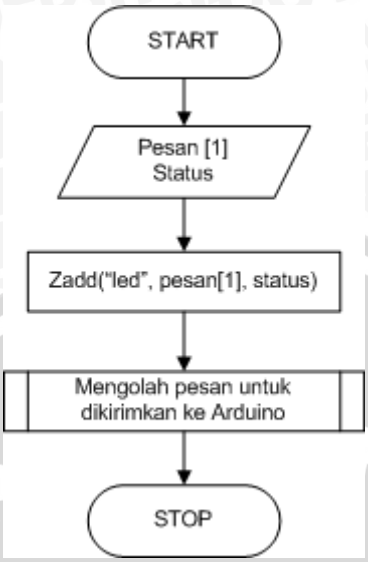
semua *LED*, Nyala(pesan[1]) untuk menyalakan salah satu *LED*, MatiSemua untuk mematikan semua *LED*, dan Mati(pesan[1]) untuk mematikan salah satu lampu.



Gambar 4. 7 Diagram Alur Fungsinyala Semua dan Mati Semua

Gambar 4.7 menunjukkan diagram alur untuk fungsinyalasemua dan matisemua. Bila proses NyalaSemua yang dilakukan oleh server maka nilai status adalah 1. Bila proses MatiSemua yang dilakukan oleh server maka nilai status adalah 0. Dari diagram alur di atas dapat dijelaskan secara rinci

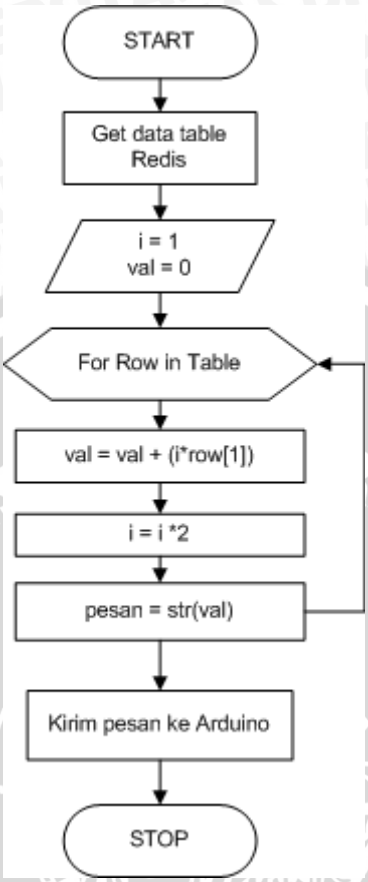
1. Inisialisasi nilai status. Nilai status dapat bernilai 1 atau 0.
2. Setelah itu server akan mengakses database Redis untuk memperbarui status semua lampu *LED*.
3. Server akan melakukan pengolahan pesan untuk dikirimkan ke Arduino, dimana proses pengolahan pesan yang nanti dikirimkan memerlukan data dari database Redis yang paling baru.



Gambar 4. 8 Diagram Alur Fungsi Mati dan Nyala Satu Lampu

Gambar 4.8 menunjukkan diagram alur untuk fungsi menyalakan atau mematikan salah satu lampu. Dimana proses keduanya digunakan untuk menyalakan atau mematikan satu lampu *LED* saja. Bila proses Nyala(pesan[1]) yang dilakukan oleh server maka nilai status adalah 1 dan pesan[1] menunjukkan ID dari lampu *LED*. Bila proses Mati(pesan[1]) yang dilakukan oleh server maka nilai status adalah 0 dan pesan[1] menunjukkan ID dari lampu *LED*. Dari diagram alur di atas dapat dijelaskan secara rinci

1. Inisialisasi nilai status dan pesan[1]. Nilai status dapat bernilai 1 atau 0 dimana nilai 1 berarti menyala sedangkan nilai 0 berarti mati. Pesan[1] menunjukkan ID lampu *LED* yang ingin dikontrol.
2. Setelah itu server akan mengakses database Redis untuk memperbarui status salah satu lampu *LED*.
3. Server akan melakukan pengolahan pesan untuk dikirimkan ke Arduino, dimana proses pengolahan pesan yang nanti dikirimkan memerlukan data dari database Redis yang paling baru.



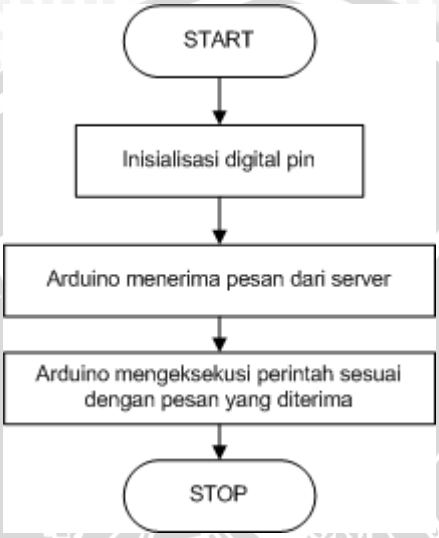
Gambar 4. 9 Diagram Alur Pengolahan Pesan Untuk Arduino

Gambar 4.9 menjelaskan proses pengolahan pesan untuk dikirimkan ke Arudino. Dimana pesan dibuat berdasarkan nilai status dari semua *LED* yang tersimpan dalam Redis. Proses tersebut nantinya akan menghasilkan output berupa pesan yang siap dikrimkan ke Arduino dalam bentuk angka atau *integer*. Dari diagram alur di atas dapat dijelaskan secara rinci

1. Server mendapatkan nilai ID dan status dari Redis. Data tersebut akan direpresentasikan dalam bentuk tabel.
2. Penginisiasian variabel i dan val. Dimana i bernilai 1 sedangkan val bernilai 0.
3. Server melakukan perulangan untuk setiap baris pada tabel. Dengan perulangan ini diharapkan server akan mendapatkan sebuah nilai dan nantinya nilai tersebut akan dikirimkan ke Arduino Uno. Nilai tersebut digunakan untuk merepresentasikan keadaan setiap lampu apakah menyala atau mati. Jika perulangan sudah mencapai baris terakhir dari tabel maka server melakukan langkah selanjutnya.
4. Server mengirimkan pesan ke Arduino Uno.

4.1.6 Perancangan Eksekusi Perintah oleh Arduino

Setelah pesan berhasil dikirimkan oleh server ke Arduino Uno, maka Arduino Uno akan melakukan tindakan sebagai actuator untuk mengontrol 4 perangkat *LED* yang terpasang. Tindakan yang akan dilakukan oleh Arduino Uno ditentukan oleh pesan yang dikirimkan oleh server. Diagram alur di bawah menunjukkan proses Arduino dalam mengontrol 4 lampu *LED*.



Gambar 4. 10 Diagram Alur Proses Kontroling LED oleh Arduino

Tabel 4. 2 Perintah Arduino Sesuai Pesan

Status Lampu	Pesan yang diterima Arduino
(2, 0), (3, 0), (4, 0), (5, 0)	0
(2, 1), (3, 0), (4, 0), (5, 0)	1
(2, 0), (3, 1), (4, 0), (5, 0)	2
(2, 1), (3, 1), (4, 0), (5, 0)	3
(2, 0), (3, 0), (4, 1), (5, 0)	4
(2, 1), (3, 0), (4, 1), (5, 0)	5
(2, 0), (3, 1), (4, 1), (5, 0)	6
(2, 1), (3, 1), (4, 1), (5, 0)	7
(2, 0), (3, 0), (4, 0), (5, 1)	8
(2, 1), (3, 0), (4, 0), (5, 1)	9
(2, 0), (3, 1), (4, 0), (5, 1)	10
(2, 1), (3, 1), (4, 0), (5, 1)	11
(2, 0), (3, 0), (4, 1), (5, 1)	12

(2, 1), (3, 0), (4, 1), (5, 1)	13
(2, 0), (3, 1), (4, 1), (5, 1)	14
(2, 1), (3, 1), (4, 1), (5, 1)	15

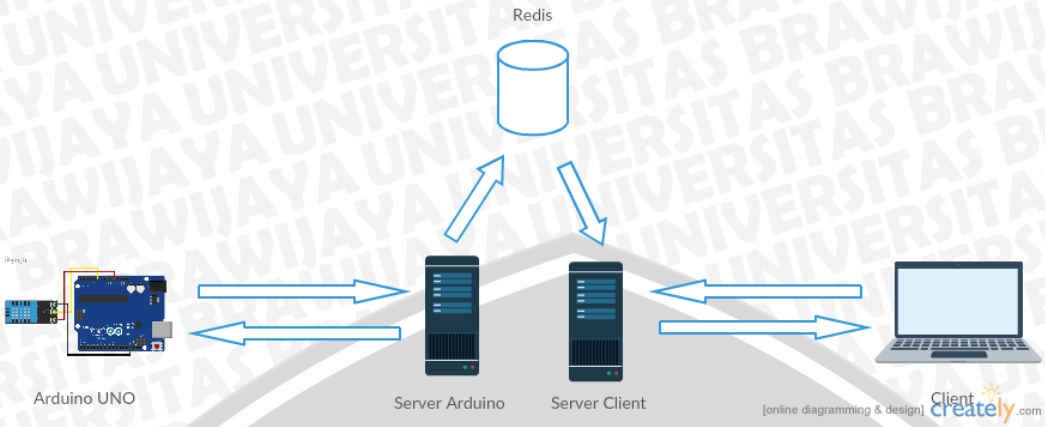
Gambar 4.10 di atas menjelaskan proses kontroling perangkat *LED* yang terpasang pada Arduino Uno. Dalam **Tabel 4.2** di atas *LED* 2 mempunyai arti sebagai perangkat *LED* yang tersambung pada pin nomor 2, *LED* 3 adalah perangkat *LED* yang tersambung pada pin nomor 3, *LED* 4 adalah perangkat *LED* yang tersambung pada pin nomor 4, dan *LED* 5 adalah perangkat *LED* yang tersambung pada pin nomor 5. Sedangkan nilai 0 berarti status lampu mati dan nilai 1 berarti status lampu hidup. Arduino Uno hanya akan melakukan sebuah proses jika Arduino Uno menerima pesan dari server. Dari diagram alur di atas dapat dijelaskan secara rinci.

1. Arduino Uno melakukan insialisasi pin. Pin yang digunakan adalah pin nomor 2, 3, 4, dan 5 dimana masing-masing pin terhubung dengan perangkat *LED* menggunakan kabel jumper.
2. Arduino menerima dan memproses pesan yang dikrimkan oleh server. Pesan yang dikirimkan oleh server berupa *integer*.
3. Arduino Uno menyeleksi pesan untuk menentukan tindakan yang harus dilakukan oleh Arduino Uno.Terdapat 16 kondisi yang nantinya akan di eksekusi oleh Arduino Uno.
4. Arduino mengeksekusi perintah dengan menyalakan atau mematikan lampu sesuai dengan pesan yang dikrimkan oleh server.

4.2 Perancangan Monitoring Suhu dan Kelembapan dari Sensor DHT 11

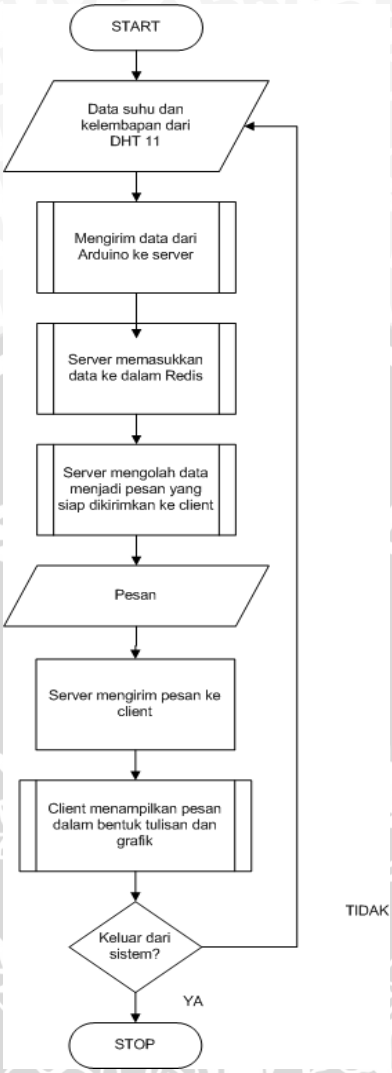
Perancangan untuk monitoring suhu dan kelembapan dari sensor DHT 11 merupakan langkah selanjtunya untuk mengimplementasikan Websocket untuk monitoring perangkat Arduino. Perancangan dilakukan secara berurutan supaya pada saat pengimplementasian berjalan dengan baik dan sistematis. Adapun perancangan yang dilakukan secara berurutan yaitu

1. Perancangan perangkat keras.
2. Perancangan database Redis.
3. Perancangan aplikasi web client .
4. Perancangan pengiriman data dari Arduino ke server.
5. Perancangan pengolahan pesan.
6. Perancangan hasil monitoring pada aplikasi web.



Gambar 4. 11 Rancangan Arsitektur Monitoring Suhu dan Kelembapan

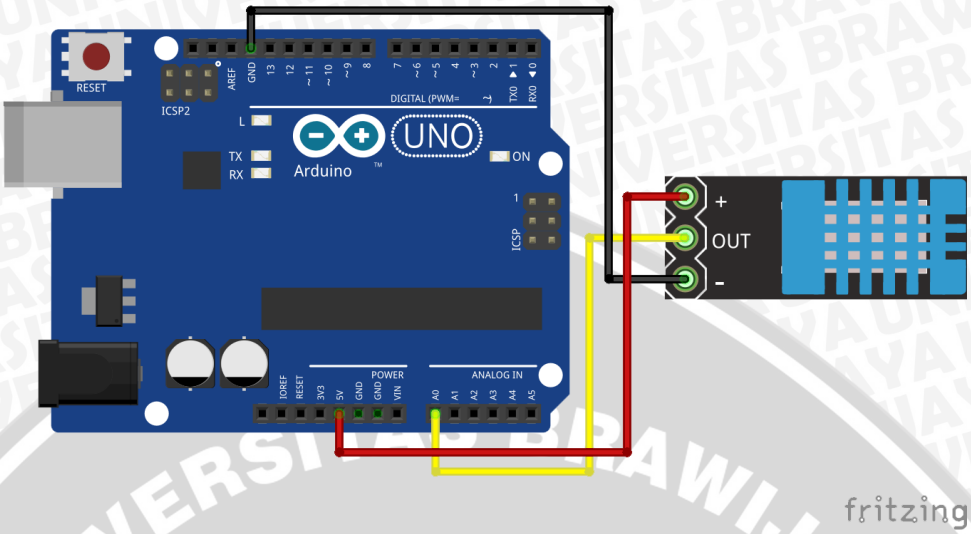
Gambar 4.11 menjelaskan secara umum arsitektur sistem yang akan dirancang. Terdapat 5 entitas yaitu Arduino Uno, server Arduino, server client, database Redis, dan client. Dalam rancangan umum topologi terdapat 4 jenis komunikasi. Yang pertama adalah komunikasi server dengan Arduino Uno, komunikasi ini digunakan untuk mengirim pesan dari server kepada Arduino Uno dan untuk mengirim data suhu dan kelembapan dari Arduino ke server Arduino. Kedua adalah komunikasi server Arudino dengan database server, komunikasi ini untuk menyimpan data suhu dan kelembapan yang didapatkan sensor DHT11 ke dalam database Redis. Ketiga adalah komunikasi Server client dengan database Redis, komunikasi digunakan untuk mendapatkan data suhu dan kelembapan yang tersimpan dalam database. Dan yang keempat dalah komunikasi antara server dengan client yang digunakan untuk mengirim dan menerima pesan baik dari server maupun client. Dalam studi kasus ini dibutuhkan 2 server supaya sistem mampu bekerja secara *real time*. **Gambar 4.12** menunjukkan diagram alur sistem untuk monitoring suhu dan kelembapan secara keseluruhan



Gambar 4. 12 Diagram Alur Sistem Monitoring Suhu dan Kelembapan Secara Keseluruhan

4.2.1 Perancangan Perangkat Keras

Untuk mendapatkan data berupa suhu dan kelembapan diperlukan adanya konfigurasi dimana DHT 11 terintegrasi dengan mikrokontroler Arduino. Untuk menyambungkan sensor DHT 11 dengan Arduino Uno dibutuhkan tiga kabel jumper. Adapun konfigurasi sensor DHT 11 dengan Arduino Uno adalah sebagai berikut :



Gambar 4. 13 Rancangan Konfigurasi DHT 11 pada Arduino UNO

Terlihat pada **Gambar 4.13**, DHT11 membutuhkan 3 komponen yaitu, vcc yang mempunyai peran sebagai sumber daya agar DHT11 dapat menyala, data agar data hasil pengamatan dapat ditransmisikan dalam bentuk binary, dan ground. Sumber daya yang dipilih sebesar 3V karena merupakan besaran voltase minimum yang dibutuhkan oleh DHT11.

4.2.2 Perancangan Database Redis

Sebuah database dibutuhkan dalam pengimplementasian monitoring suhu dan kelembapan sebagai media penyimpanan nilai suhu dan kelembapan dari sensor DHT 11. Dalam perancangan monitoring suhu dan kelembapan dari sensor DHT 11 ini membutuhkan 3 database Redis, dimana masing-masing database menyimpan nilai suhu, kelembapan, dan waktu. Waktu yang tersimpan dalam database digunakan sebagai informasi kapan nilai suhu dan kelembapan berhasil didapatkan oleh sensor DHT 11.

Redis sendiri memiliki lima struktur data dengan fungsinya yang berbeda-beda yaitu *string*, *sets*, *hashes*, *lists*, dan *sorted sets*. Pada perancangan database Redis untuk memonitoring suhu dan kelembapan dari sensor DHT 11 menggunakan struktur data *lists*. Alasan *lists* dipilih dalam perancangan monitorng suhu dan kelembapan sensor DHT 11 karena, pertama struktur data *lists* dapat menyimpan data yang memiliki nilai sama, hal ini cocok dengan kebutuhan sistem yang memungkinkan server dapat menyimpan nilai suhu atau kelembapan yang sama tetapi memiliki waktu yang berbeda. Kedua, *lists* memungkinkan menyimpan data secara berurutan, sehingga data yang didapatkan sensor DHT 11 dapat dimasukkan ke dalam database secara berurutan dengan waktu yang berurutan juga. Ketiga, struktur data *lists* dapat menyimpan data lebih dari 4 milyar data sehingga sistem tidak perlu untuk menghapus data yang ada sebelumnya untuk waktu yang lama.

Secara sederhana struktur data database dalam perancangan ini digambarkan oleh **Tabel 4.3, Tabel 4.4** dan **Tabel 4.5**.

Tabel 4. 3 Rancangan Database Suhu Redis

Lists	Elements
Temp	28
Temp	27
temp	28

Tabel 4. 4 Rancangan Database Kelembapan Redis

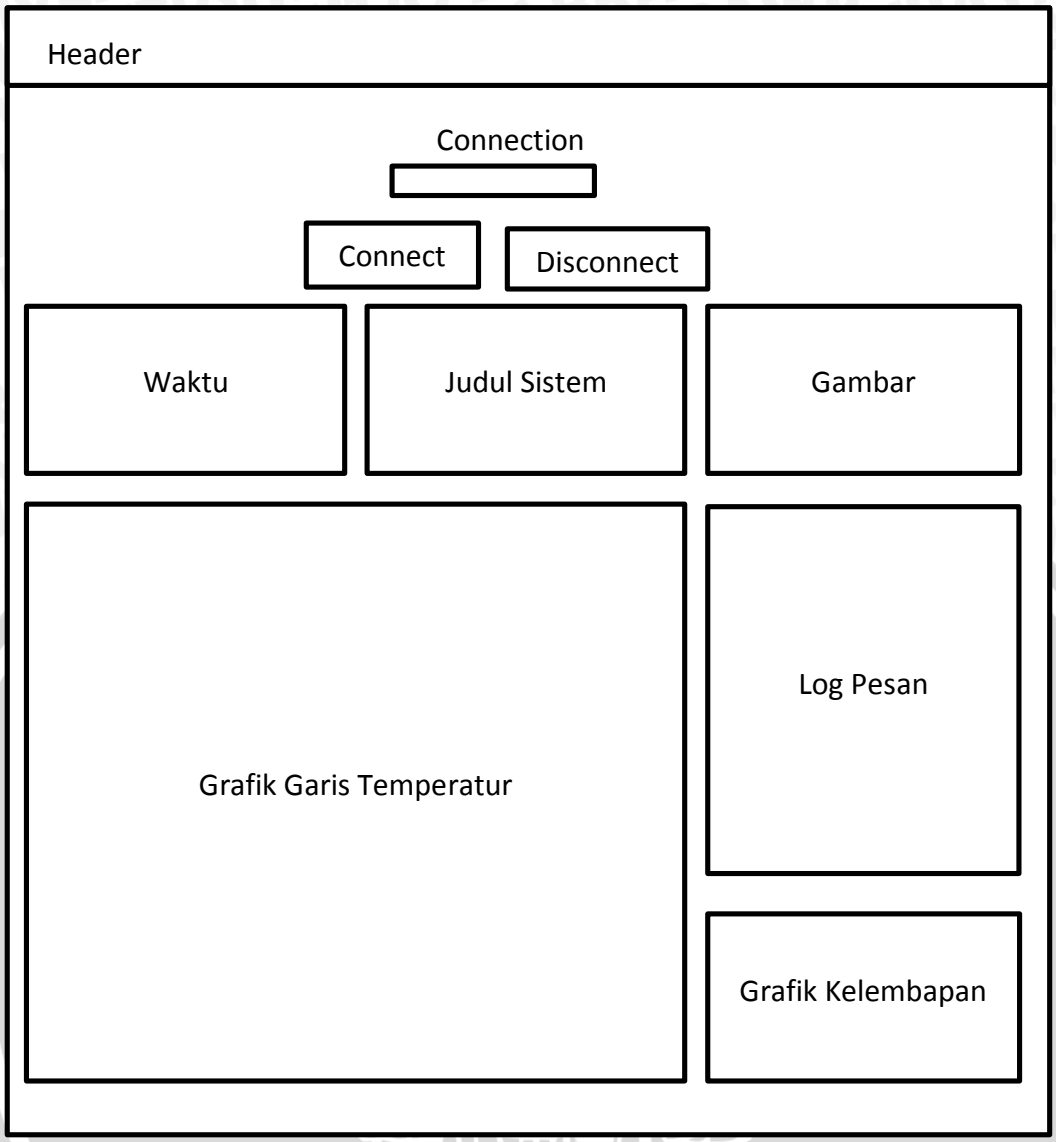
Lists	Elements
Humid	75
Humid	80
Humid	80

Tabel 4. 5 Rancangan Database Waktu Redis

Lists	Elements
Time	2016-6-2 16:30:00
Time	2016-6-2 16:31:00
time	2016-6-2 16:32:00

4.2.3 Perancangan Aplikasi Web Client

Perancangan aplikasi web client ini digunakan sebagai *interface* sistem. Perancangan web ini adalah hal yang sangat penting dalam perancangan sistem secara keseluruhan, karena aplikasi web in adalah bagian *interface* yang akan paling banyak digunakan oleh pengguna untuk menjalankan fungsi-fungsi dari sistem ini. **Gambar 4.14** menggambarkan rancangan aplikais web client yang digunakan untuk memonitoring suhu dan kelembapan.



Gambar 4. 14 Rancangan Aplikasi Web Client untuk Memonitor Suhu dan Kelembapan

Program aplikasi web berfungsi untuk mengontrol semua lampu *LED* yang terpasang pada Arduino Uno. Sistem ini dituliskan dalam bahasa *html* dan *javascript*. Selain itu program aplikasi web ini juga harus memiliki fitur-fitur sebagai berikut :

1. Sistem harus mampu untuk melakukan koneksi ke server menggunakan Websocket.
2. Sistem harus mampu untuk melakukan pemutusan koneksi ke server.
3. Sistem harus dapat menerima dan menampilkan pesan dari server pada log pesan.
4. Sistem harus bisa menerima data temperatur yang dikirim oleh server dan menampilkan data temperatur tersebut ke dalam grafik garis.

5. Sistem harus bisa menerima data kelembapan yang dikirim oleh server dan menampilkan data kelembapan tersebut ke dalam grafik speedometer.

Setelah fungsi-fungsi didefinisikan, maka perlu dibuat sebuah *layout* untuk gambaran *interface* web seperti pada gambar.Adapun elemen-elemen yang terkandung pada *layout* yang memiliki fungsi masing-masing yaitu :

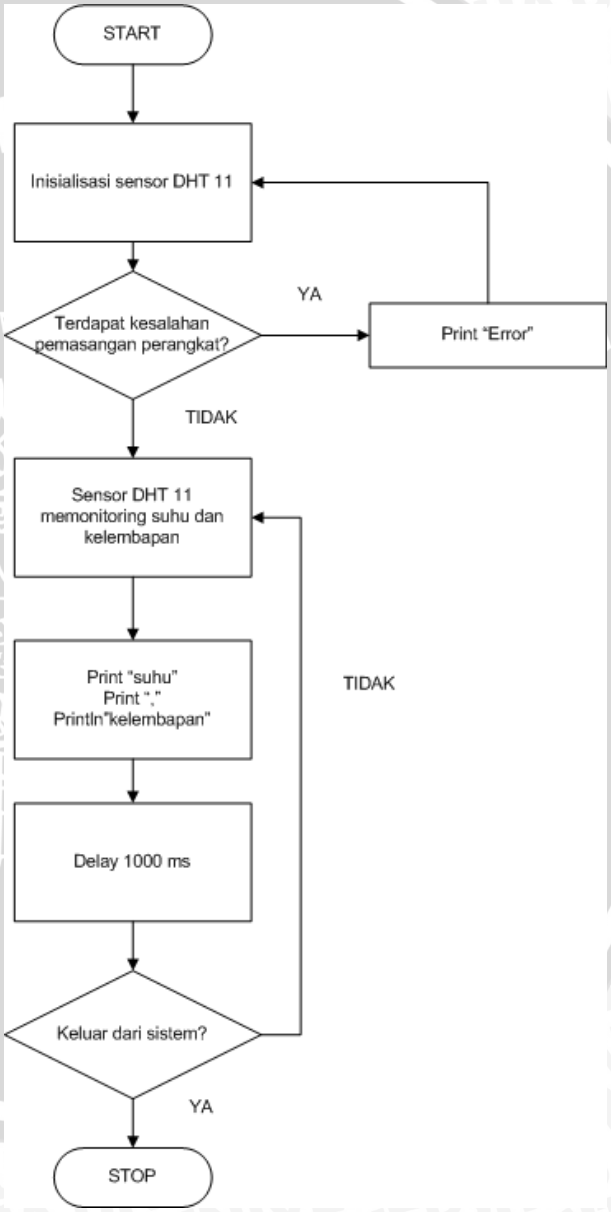
1. *Field* Connection : Digunakan untuk menuliskan alamat Websocket yang digunakan oleh server. Contohnya ws://ip_address:*port.*
2. Button Connect : Digunakan untuk melakukan koneksi dengan server dengan alamat yang sudah dituliskan pada *field* connection.
3. Button Disconnect : Digunakan untuk melakukan pemutusan koneksi dengan server.
4. *Field* Log Pesan : Digunakan untuk menampilkan pesan dari server.
5. *Field* Grafik Garis Temperatur : Digunakan untuk menampilkan data tempertaur yang dikirimkan oleh server dan direpresentasikan dalam bentuk grafik garis.
6. *Field* Grafik Kelembapan : Digunakan untuk menampilkan data kelembapan yang dikrimkan oleh server dan direpresentasikan dalam bentuk grafik speedometer.

Aplikasi web client dituliskan dalam bahasa *html* dan *javascript*. Untuk mendukung komunikasi Webcocket dibutuhkan juga sebuah client yang mampu membuat koneksi Websocket dengan server. Maka dari itu dalam studi kasus ini dalam perancangan aplikasi web client, web client membutuhkan sebuah Websocket API. Umumnya Websocket API pada web client memiliki 4 fungsi utama antara lain onOpen adalah sebuah fungsi yang digunakan untuk menbuat koneksi dengan server. OnClose adalah sebuah fungsi yang digunakan untuk menutup koneksi dengan server. OnMessage adalah sebuah fungsi untuk menerima pesan dari server. Dan OnEror adalah sebuah fungsi untuk menampilkan pesan error saat client gagal melakukan koneksi dengan server.

Sedangkan untuk menggambarkan grafik garis dan grafik speedometer, dibutuhkan sebuah API yang mampu mengubah sebuah pesan menjadi sebuah grafik. Dalam studi kasus ini web client menggunakan Google Chart API untuk menggamabarkan grafik garis dan speedometer. Dengan menggunakan Google Chart API web client hanya membutuhkan sebuah data JSON untuk dirubah menjadi sebuah grafik.

4.2.4 Perancangan Pengiriman Data dari Arduino ke Server

Supaya sensor DHT 11 mampu mendapatkan nilai suhu dan kelembapan, maka diperlukan perancangan sensor DHT 11 yang terpasang pada Arduino Uno. DHT 11 merupakan suatu variabel pasif dimana ketika Arduino Uno menginginkan data, mikrokontroler harus melakukan trigger untuk menyalakan sensor. Nilai yang didapatkan oleh sensor DHT 11 adalah nilai suhu dan kelembapan dan nantinya nilai tersebut akan dikirimakn oleh server untuk diolah. Proses rancangan pengambilan data suhu dan kelembapan oleh sensor DHT 11 digambarkan pada **Gambar 4.15.**

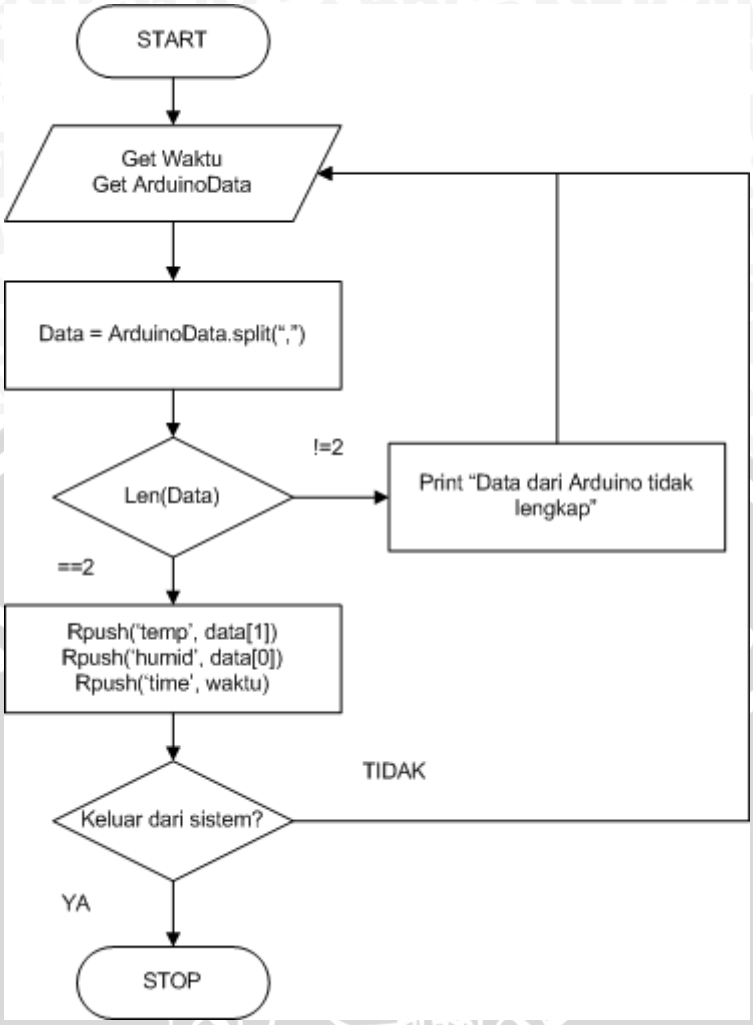


Gambar 4. 15 Diagram Alur Proses Pengambilan Data Suhu dan Kelembapan

Gambar 4.15 menjelaskan rancangan sensor DHT 11 untuk mendapatkan data suhu dan kelembapan. Data suhu dan kelembapan yang sudah berhasil didapatkan nantinya akan dipisahkan dengan tanda koma seperti contohnya 28,45. Penggunaan tanda koma nantinya akan digunakan untuk membedakan nilai suhu dan nilai kelembapan. Data yang sudah dicetak tersebut lah yang nantinya akan dikirimkan ke server. Dari diagram alur di atas dapat dijelaskan secara rinci.

1. Arduino Uno melakukan inisialisai pin untuk mengetahui apakah sensor DHT 11 sudah terpasang dengan benar pada Arduino Uno.
2. Arduino Uno melakukan fungsi membaca suhu dan kelembapan.
3. Setelah DHT 11 berhasil mendapatkan nilai suhu dan kelembapan, Arduino Uno akan mencetak nilai suhu dan kelembapan disertai dengan tanda koma pada serial monitor. Data yang telah dicetak tersebut lah yang nantinya akan dikirimkan ke server.
4. Arduino melakukan delay 1000 ms atau 1 detik. Hal ini perlu dilakukan supaya Arduino Uno mampu mendapatkan nilai suhu dan kelembapan selanjutnya secara presisi dan tepat.
5. Arduino akan mengulangi fungsi membaca suhu dan kelembapan sampai ada perintah untuk keluar dari sistem.

Setelah data suhu dan kelembapan berhasil didapatkan sensor DHT 11 dan diproses oleh Arduino Uno, data tersebut akan dikirimkan ke server untuk disimpan ke dalam database Redis. Data suhu akan disimpan ke dalam *lists* bernama 'temp', dan data kelembapan akan disimpan ke dalam *lists* bernama 'humid'. Selain adanya proses penyimpanan data suhu dan kelembapan, server juga menyimpan data waktu dimana data waktu didapatkan ketika data suhu dan kelembapan diterima oleh server. Data waktu tersebut digunakan sebagai informasi kapan data suhu dan kelembapan berhasil didapatkan oleh sensor DHT 11. Proses rancangan penyimpanan data oleh server dapat dijelaskan pada **Gambar 4.16**



Gambar 4. 16 Diagram Alur Penyimpana Data

Gambar 4.16 di atas menjelaskan rancangan penyimpanan data ke dalam database Redis yang dilakukan oleh server. Data yang didapatkan oleh server sama dengan apa yang dicetak oleh Arduino Uno pada Serial Monitor. Proses penyimpanan data dilakukan supaya sistem dapat memonitoring keadaan suhu dan kelembapan secara *real time* dan proses penyimpanan data ini dilakukan secara berulang-ulang. Server akan melakukan proses penyimpanan data setiap detik, hal ini dikarenakan pada proses rancangan mendapatkan data suhu dan kelembapan yang dilakukan oleh DHT 11 dilakukan pengambilan data setiap 1 detik. Dari diagram alur di atas dapat dijelaskan secara rinci.

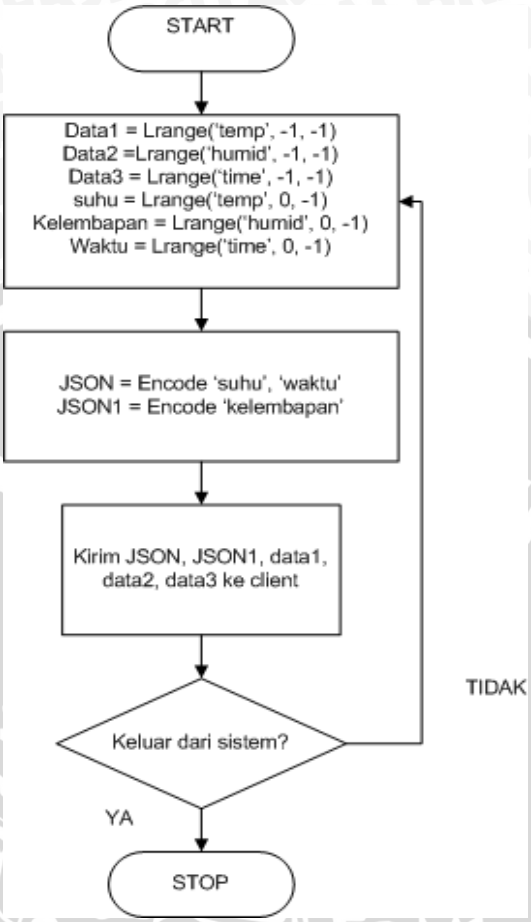
1. Server menerima data suhu dan kelembapan dari Arduino Uno. Server juga mendapatkan nilai waktu, tepat pada saat data dari Arduino diterima oleh server.
2. Data dari Arduino tersebut akan dipisahkan setiap terdapat tanda koma. Proses ini perlu dilakukan untuk membedakan data suhu dan data kelembapan.
3. Server melakukan pengkondisian terhadap hasil pemisahan data Arduino. Apabila jumlah elemen sama dengan 2 maka server akan menyimpan data suhu pada *list* "temp", data kelembapan pada *list*

"humid", dan data waktu pada *list* "time". Apabila jumlah elemen tidak sama dengan 2 maka server akan memberitahu bahwa data yang dikirimkan oleh Arduino tidak lengkap dan tidak melakukan proses penyimpanan data.

4.2.5 Perancangan Pengolahan Pesan

Dalam perancangan monitoring suhu dan kelembapan menggunakan sensor DHT 11 ini membutuhkan 2 server. Selain server untuk berkomunikasi dengan Arduino, dibutuhkan server untuk berkomunikasi dengan client. Tujuan dibuatnya 2 server ini supaya sistem dapat berjalan secara *real time*. Server yang digunakan untuk berkomunikasi dengan client ini melakukan proses pengolahan data menjadi pesan yang siap dikrimkan ke client menggunakan Websocket. Dimana data yang diolah adalah data yang didapatkan dari database Redis diantaranya adalah *list* "temp", "humid", dan "time". Salah satu proses pengolahan data yang dilakukan oleh server adalah mengubah *lists* "temp", "humid", dan "time" menjadi sebuah data bertipe JSON. Nantinya data JSON tersebut akan dikirimkan ke client dan akan diolah oleh proses yang ditulis dalam *javascript* menjadi sebuah grafik garis dan grafik speedometer. Grafik garis nantinya akan menunjukkan sebuah hasil monitoring suhu setiap detiknya dan akan terus-menerus diperbarui setiap detiknya. Sedangkan grafik speedometer akan menunjukkan nilai kelembapan setiap detiknya.

Dalam studi kasus ini untuk mengimplementasikan sebuah komunikasi Websocket dibutuhkan sebuah server yang mampu membuat koneksi jaringan komunikasi dengan client. Salah satu *library* python yang mendukung komunikasi Websocket antara server python dengan web client adalah *library twisted* txws. Umumnya *library* txws mempunyai 3 fungsi utama antara lain, fungsi connectionMade digunakan untuk menerima koneksi dari client. connectionLost digunakan untuk memutuskan koneksi dengan client. Dan fungsi datareceived digunakan untuk menerima data yang dikirimkan oleh client dan mengirimkan data kepada client. Dengan adanya persamaan fungsi *library* txws dengan Websocket API client, dalam studi kasus ini digunakan sebuah *library* python txws.



Gambar 4. 17 Diagram Alur Pengolahan Pesan dan Pengiriman Pesan

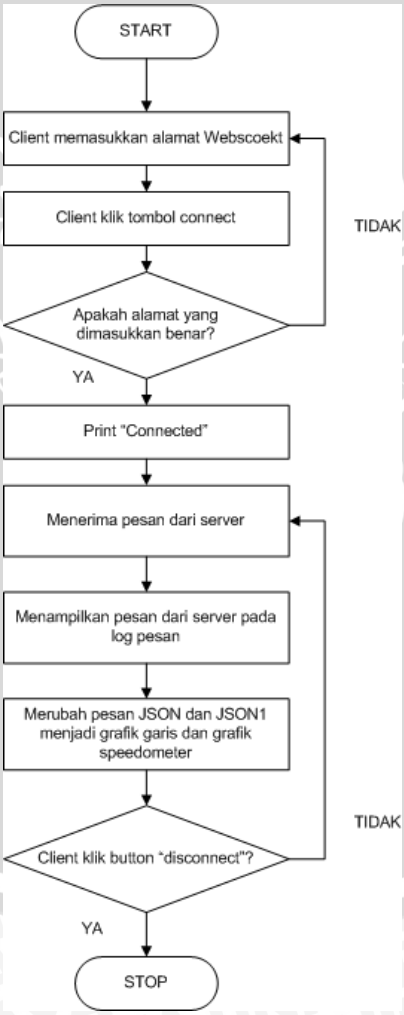
Gambar 4.17 menunjukkan diagram alur untuk proses pengolahan pesan dan pengiriman pesa. Untuk menampilkan data suhu dan kelembapan pada web client, diperlukan pesan dari server yang sudah terolah dan bisa ditampilkan di dalam *browser*. Untuk menampilkan grafik secara *real time* maka diperlukan adanya data bertipe JSON, dimana nantinya data JSON ini akan dirubah oleh web client menjadi sebuah grafik. Dikarenakan pada perancangan aplikasi web client akan menampilkan 2 grafik maka data JSON yang diperlukan berjumlah 2. Diantaranya adalah JSON yang berisi data suhu dan kelembapan untuk menampilkan grafik garis suhu, dan JSON yang berisi data kelembapan untuk menampilkan grafik speedometer. Data yang dirubah menjadi JSON tersebut adalah data yang tersimpan pada database Redis, dimana database tersebut akan diperbarui setiap 1 detik. Dari diagram alur dapat dijelaskan secara rinci.

1. Server mendapatkan data *lists* "temp", "humid", dan "time" dari database Redis. Server mengakses semua *lists* tersebut sebanyak 2 kali, dimana yang pertama server akan mendapatkan semua nilai dari *list* tersebut dari awal sampai akhir . Hasil pengambilan semua nilai akan diinisialisasikan menjadi variabel suhu, kelembapan, dan waktu. Yang kedua server akan mendapatkan nilai dari semua *lists* tetapi hanya mendapatkan nilai paling

akhir saja. Hasil pengambilan nilai paling akhir tersebut akan diinisialisasikan menjadi *Variable* data1, data2, dan data3. Tujuan pengambilan nilai akhir tersebut untuk menampilkan informasi suhu, kelembapan dan waktu sekarang kepada client dalam bentuk tulisan atau *string*, yang nantinya akan ditampilkan oleh web pada *field* Log Pesan.

2. Server membuat 2 data JSON. JSON yang pertama berisi semua nilai suhu dan waktu yang didapatkan dari *lists* "temp", dan "waktu". JSON kedua berisi semua nilai kelembapan yang didapatkan dari *lists* "humid".
3. Server mengirimkan sebanyak 5 pesan kepada client. Yaitu JSON, JSON1, data1, data2, dan data3
4. Untuk mendapatkan data suhu dan kelembapan secara *real time*, proses pengiriman pesan dilakukan berulang-ulang setiap 1 detik.

4.2.6 Perancangan Hasil Monitoring pada Aplikasi Web



Gambar 4. 18 Diagram Alur Menampilkan Pesan dan Grafik pada Aplikasi Web

Setelah pesan berhasil dikirimkan oleh server kepada client menggunakan protokol Websocket. Maka web client bertugas untuk menampilkan semua pesan yang diterima oleh server. Data yang bertipe JSON akan dirubah oleh web client menjadi grafik garis dan grafik speedometer. Dalam studi kasus ini pembuatan grafik dibuat menggunakan Google Chart API. Dengan mengikuti struktur data JSON yang dibutuhkan API Google Chart, data JSON yang diterima tersebut akan langsung dirubah menjadi sebuah grafik.

Gambar 4.18 menunjukkan proses untuk menampilkan semua pesan yang telah dikirimkan oleh server. Proses menampilkan pesan dari server dilakukan secara berulang-ulang, baik itu dalam menampilkan grafik maupun menampilkan pesan dari server yang akan ditampilkan di *field* log pesan. Web client akan berhenti memperbarui pesan yang ditampilkan ketika client melakukan *disconnect*. Setiap sensor DHT 11 mendapatkan data suhu dan kelembapan baru maka web client akan melakukan proses pembaruan pesan dan grafik. Sehingga membuat sistem menjadi sistem yang *real time*. Dari diagram alur di atas dapat dijelaskan secara rinci

1. Pertama pengguna memasukkan alamat yang sesuai dengan alamat yang dipakai server. Pengguna dapat memasukkan alamat tersebut pada *field* yang tersedia pada sistem sesuai dengan perancangan aplikasi web. Adapun format alamt yang dipakai adalah ws://(ip_address server):(*port*).
2. Pengguna mengklik button “Connect”.
3. Jika alamat yang dimasukkan benar maka sistem akan mencetak pesan “CONNECTED” untuk menandakan client berhasil tersambung dengan server. Sebaliknya jika alamat yang dimasukkan salah maka sistem akan mencetak “ERROR” pada web.
4. Setelah client berhasil melakukan koneksi dengan server, web client akan langsung menerima pesan yang dikrimkan oleh server.
5. Terdapat 2 tipe pesan yang dikirimkan oleh server yaitu tipe data *string* dan tipe data JSON. Untuk tipe data *string* web client akan langsung menampilkannya di *field* log pesan. Sedangkan tipe data JSON web client akan mengubah data tersebut menjadi grafik garis dan grafik speedometer.
6. Web client akan terus melakukan pembaruan pesan dan grafik secara otomatis setiap detiknya sampai pengguna mengklik tombol disconnect.

4.3 Perancangan Pengujian

Pengujian dilakukan untuk mengetahui seberapa handal suatu sistem. Terdapat 3 macam pengujian dalam studi kasus ini diantara nya adalah pengujian transmisi data, pengujian kehandalan sistem dalam melayani request, dan pengujian fungsional aplikasi web client. Dengan pengujian ini diharapkan akan menghasilkan data-data yang mampu merepresentasikan kehandalan sistem secara keseluruhan.

4.3.1 Perancangan Pengujian Proses Transmisi Data

Protokol Websocket dan komunikasi serial mempunyai peran penting dalam semua pengimplementasian yang dilakukan. Websocket lah yang mengatur mengalirnya data baik dari server ke client atau client menuju server. Sedangkan komunikasi serial yang mengatur aliran data dari server ke Arduino atau Arduino ke server. Untuk memastikan transimisi data berjalan sesuai kebutuhan, maka dilakukan pengujian untuk mengetahui seberapa cepat data dapat mengalir mulai dari client menuju ke Arduino sampai data kembali ke client per detiknya. Mekanisme yang dilakukan dalam pengujian ini adalah penghitungan Round Trip Time (RTT) sebuah data yang mengalir dari client ke Arduino hingga kembali ke client lagi.

Pengujian dilakukan dengan 2 keadaan,yaitu pengujian dengan menggunakan Redis dan pengujian dengan tidak menggunakan Redis. Masing-masing keadaan tersebut akan diuji dengan parameter yang berbeda-beda. Parameter dalam pengujian ini meliputi jumlah request yang dirubah setiap pengujiannya dan jumlah client yang dirubah setiap pengujiannya. Jumlah request dan jumlah client yang digunakan mula-mula sebanyak 50, pengujian selanjutnya sebanyak kelipatan 50 sampai pengujian mencapai 500. Setiap parameter akan diuji sebanyak 5 kali dan akan diambil rata-ratanya. Dalam studi kasus ini pengujian proses transmisi data akan dilakukan pada pengimplementasian monitoring suhu dan kelembapan dari sensor DHT11 saja, Dari pengujian tersebut dapat diketahui apakah terdapat kecocokan antara kemampuan Websocket dalam mengakses perangkat yang terpasang pada Arduino Uno.

4.3.2 Perancangan Pengujian Kehandalan Sistem Melayani Request

Dalam suatu sistem monitoring, server seharusnya dapat melayani request data dalam jumlah yang banyak. Misalnya dalam kasus monitoring suhu dan kelembapan server harus dapat melayani request data dari beberapa client. Untuk memastikan kehandalan sistem yang menggunakan protokol Websocket dan perangkat Arduino maka dilakukan pengujian kehandalan sistem melayani request yang ada. Dimana dalam pengujian akan dihitung berapa request yang mampu dilayani oleh server per detiknya.

Hampir sama dengan pengujian proses transmisi data, pengujian dilakukan dengan 2 keadaan,yaitu pengujian dengan menggunakan Redis dan pengujian dengan tidak menggunakan Redis. Masing-masing keadaan tersebut akan diuji dengan parameter yang berbeda-beda. Parameter dalam pengujian ini meliputi jumlah request yang dirubah setiap pengujiannya dan jumlah client yang dirubah setiap pengujiannya. Jumlah request dan jumlah client yang digunakan mula-mula sebanyak 50, pengujian selanjutnya sebanyak kelipatan 50 sampai pengujian mencapai 500. Setiap parameter akan diuji sebanyak 5 kali dan akan diambil rata-ratanya. Dalam studi kasus ini pengujian kehandalan sistem dalam melayani request akan dilakukan pada pengimplementasian monitoring suhu dan kelembapan dari sensor DHT11 saja. Dari pengujian tersebut dapat diketahui

apakah terdapat kecocokan sistem yang menggunakan Websocket dan Arduino sebagai perangkat sensor dalam melayani request yang banyak.

4.3.3 Perancangan Pengujian Fungsionalitas Aplikasi Web Client

Supaya dapat menjalankan semua fungsi yang tersedia dalam aplikasi web, diperlukan sebuah pengujian yang menguji apakah semua fitur yang tersedia dalam aplikasi web dapat berjalan dengan baik. Pengujian fungsionalitas ini dilakukan pada semua aplikasi web baik itu aplikasi web untuk monitoring suhu dan kelembapan atau aplikasi web untuk kontroling lampu *LED*. Pengujian dilakukan untuk setiap fungsi yang sudah dijabarkan pada bab pengimplemenasian.

1. Pengujian fungsionalitas apliaski web kontroling lampu *LED*

Tabel 4. 6 Skenario Pengujian Pembuatan Koneksi Aplikasi Web Kontroling LED

Test Case	Pengujian pembuatan koneksi antara client dengan server
Prosedur	1. Client mengisi alamat pada *field* connection 2. Client mengklik tombol connected
Keluaran yang Diharapkan	Sistem menampilkan pesan “connected” dan komunikasi antara client dengan server terbentuk

Tabel 4. 7 Skenario Pengujian Pemutusan Koneksi Aplikasi Web Kontroling LED

Test Case	Pengujian pemutusan koneksi antara client dengan server
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol disconnect
Keluaran yang Diharapkan	Sistem menampilkan pesan “disconnected” dan komunikasi antara client dengan server terputus.

Tabel 4. 8 Skenario Pengujian Pengiriman Pesan Hidup Semua

Test Case	Pengujian pengiriman pesan hidup semua
Prosedur	1. Client telah terkoneksi dengan server

	2. Client mengklik tombol hidup semua
Keluaran yang Diharapkan	Sistem mengirim pesan hidup semua kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 9 Skenario Pengujian Pengiriman Pesan Mati Semua

Test Case	Pengujian pengiriman pesan mati semua
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol mati semua
Keluaran yang Diharapkan	Sistem mengirim pesan mati semua kepada server dan menampilkan “sent : mati semua” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 10 Skenario Pengujian Pengiriman Pesan Hidup 2

Test Case	Pengujian pengiriman pesan hidup 2
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol hidup 2
Keluaran yang Diharapkan	Sistem mengirim pesan hidup 2 kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 11 Skenario Pengujian Pengiriman Pesan Mati 2

Test Case	Pengujian pengiriman pesan mati 2
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol mati 2

Keluaran yang Diharapkan	Sistem mengirim pesan mati 2 kepada server dan menampilkan “sent : mati 2” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 12 Skenario Pengujian Pengiriman Pesan Hidup 3

Test Case	Pengujian pengiriman pesan hidup 3
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol hidup 3
Keluaran yang Diharapkan	Sistem mengirim pesan hidup 3 kepada server dan menampilkan “sent : hidup 3” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 13 Skenario Pengujian Pengiriman Pesan Mati 3

Test Case	Pengujian pengiriman pesan mati 3
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol mati 3
Keluaran yang Diharapkan	Sistem mengirim pesan mati 3 kepada server dan menampilkan “sent : mati 3” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 14 Skenario Pengujian Pengiriman Pesan Hidup 4

Test Case	Pengujian pengiriman pesan hidup 4
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol hidup 4
Keluaran yang Diharapkan	Sistem mengirim pesan hidup 4 kepada server dan menampilkan “sent : hidup 4” dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 15 Skenario Pengujian Pengiriman Pesan Mati 4

Test Case	Pengujian pengiriman pesan mati 4
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol mati 4
Keluaran yang Diharapkan	Sistem mengirim pesan mati 4 kepada server dan menampilkan "sent : mati 4" dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 16 Skenario Pengujian Pengiriman Pesan Hidup 5

Test Case	Pengujian pengiriman pesan hidup 5
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol hidup 5
Keluaran yang Diharapkan	Sistem mengirim pesan hidup 5 kepada server dan menampilkan "sent : hidup 5" dan waktu eksekusi yang dikirim oleh server.

Tabel 4. 17 Skenario Pengujian Pengiriman Pesan Mati 5

Test Case	Pengujian pengiriman pesan mati 5
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol mati 5
Keluaran yang Diharapkan	Sistem mengirim pesan mati 5 kepada server dan menampilkan "sent : mati 5" dan waktu eksekusi yang dikirim oleh server.

2. Pengujian Fungsionalitas aplikasi web monitoring suhu dan kelembapan.

Tabel 4. 18 Skenario Pengujian Pembuatan Koneksi Aplikasi Web Monitoring Suhu dan Kelembapan

Test Case	Pengujian pembuatan koneksi antara client dengan server
Prosedur	1. Client mengisi alamat pada *field* connection 2. Client mengklik tombol connected
Keluaran yang Diharapkan	Sistem menampilkan pesan "connected" dan komunikasi antara client dengan server terbentuk

Tabel 4. 19 Skenario Pengujian Pemutusan Koneksi Aplikasi Web Monitoring Suhu dan Kelembapan

Test Case	Pengujian pemutusan koneksi antara client dengan server
Prosedur	1. Client telah terkoneksi dengan server 2. Client mengklik tombol disconnect
Keluaran yang Diharapkan	Sistem menampilkan pesan "disconnected" dan komunikasi antara client dengan server terputus.

Tabel 4. 20 Skenario Pengujian Menampilkan Pesan pada Log Pesan

Test Case	Pengujian menampilkan pesan dari server
Prosedur	1. Client telah terkoneksi dengan server 2. Client menunggu pesan dari server
Keluaran yang Diharapkan	Sistem menampilkan pesan dari server yaitu pesan waktu, suhu dalam celcius, suhu dalam fahrenheit, suhu dalam kelvin, dan kelembapan

Tabel 4. 21 Skenario Pengujian Menampilkan Grafik Garis

Test Case	Pengujian menampilkan grafik garis
Prosedur	1. Client telah terkoneksi dengan server 2. Client menunggu pesan JSON dari server
Keluaran yang Diharapkan	Sistem menampilkan grafik garis sesuai dengan isi dari JSON yang dikirimkan

Tabel 4. 22 Skenario Pengujian Menampilkan Grafik Speedometer

Test Case	Pengujian menampilkan grafik speedometer
Prosedur	1. Client telah terkoneksi dengan server 2. Client menunggu pesan JSON dari server
Keluaran yang Diharapkan	Sistem menampilkan grafik garis sesuai dengan isi dari JSON yang dikirimkan

BAB 5 IMPLEMENTASI

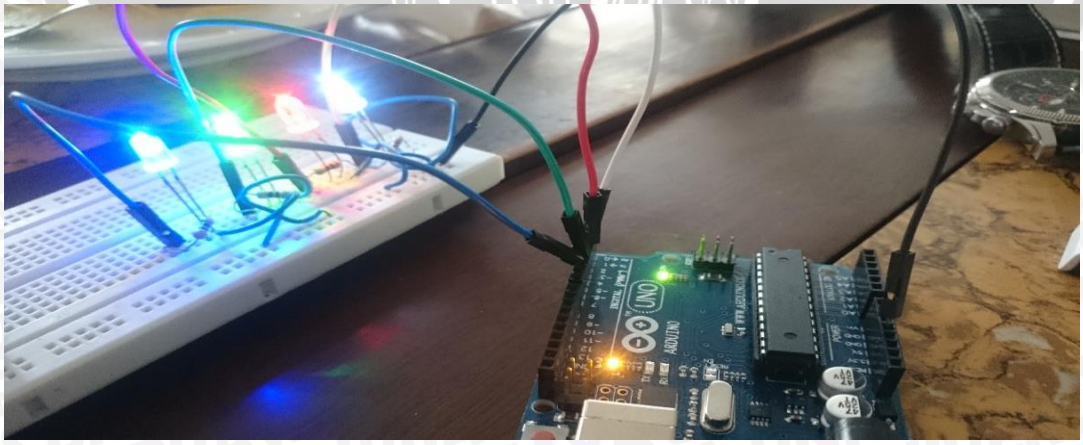
Implementasi merupakan bagian penting dalam penelitian. Tahap implementasi menetukan apakah penelitian sesuai dengan tujuan atau tidak. Agar mampu mencapai tujuan yang diinginkan, implementasi sistem dilaksanakan berdasarkan perancangan yang telah dibuat sebelumnya. Dalam bab ini terdapat 2 macam pengimplementasian yang disesuaikan dengan perancangan. Implementasi yang pertama adalah pengimplementasian kontroling lampu *LED*. Dan implementasi yang keduda adalah pengimplementasian monitoring suhu dan kelembapan menggunakan sensor DHT 11. Kedua pengimplementasian menggunakan *library* protokol Websocket yang sama yaitu txws.

5.1 Implementasi Kontroling Lampu *LED*

Implementasi kontroling lampu *LED* menjelaskan bagaimana untuk mengimplementasikan Websocket untuk mengontrol perangkat *LED* yang terpasang pada Arduino Uno. Sesuai dengan perancangan, dalam pengimplementasian kontroling lampu *LED* meliputi, implementasi perangkat keras, implementasi database redis, implementasi aplikasi web client, implementasi komunikasi server dengan client, dan implementasi komunikasi server dengan Arduino.

5.1.1 Implementasi Perangkat Keras

Berdasarkan perancangan perangkat keras yang telah dibuat sebelumnya, 4 lampu *LED* yang memiliki warna berbeda-beda yaitu putih, merah, hijau, dan biru dihubungkan pada mikrokontroler Arduino Uno dibantu dengan kabel jumper. Selain menggunakan kabel jumper dalam implementasi juga menggunakan 4 buah resistor yang dipasang pada setiap *LED*. Pemasangan resistor digunakan untuk menahan tegangan *list*rik untuk masing-masing *LED*. Jadi ketika Arduino Uno akan mematikan sebuah lampu *LED*, maka resistor yang terpasang pada *LED* tersebut akan menghambat tegangan *list*rik untuk *LED* tersebut.



Gambar 5. 1 Implementasi Konfigurasi LED pada Arduino

Dari **Gambar 5.1** dapat dilihat bahwa 4 lampu *LED* dihubungkan ke dalam pin yang tersedia pada Arduino Uno. Untuk memudahkan konfigurasi warna yang digunakan untuk menghubungkan *LED* ke dalam pin sama dengan warna *LED*. Dimana *LED* warna putih dihubungkan pada pin nomor 2, *LED* warna merah dihubungkan pada pin nomor 3, *LED* warna hijau dihubungkan pada pin nomor 4, dan *LED* warna biru dihubungkan pada pin nomor 5. 4 Resistor dipasang pada masing-masing *LED*, yang digunakan untuk menghambat tegangan *list*rik untuk lampu *LED*. Untuk mendapatkan tegangan *list*rik maka papan dihubungkan ke pin voltase 5V yang tersedia pada Arudino Uno menggunakan kabel jumper. Untuk mengintegrasiakn Arduino Uno dengan laptop digunakan 1 buah kabel USB yang dipasang pada Arduino dan dihubungkan pada *port* USB laptop.

5.1.2 Implementasi Database Redis

Untuk membuat media penyimpanan Redis, dibuatlah script menggunakan bahasa python. Sesuai dengan perancangan, database Redis yang dibuat bertipe data *sorted sets*. Pembuatan database Redis dilakukan oleh server, dimana server mempunyai akses untuk mendapatkan nilai dan mengubah nilai yang tersimpan dalam database Redis.

```
1.  r_server = redis.Redis('localhost')
2.  stat = 1
3.  r_server.zadd('LED', 2, stat)
4.  r_server.zadd('LED', 3, stat)
5.  r_server.zadd('LED', 4, stat)
6.  r_server.zadd('LED', 5, stat)
```

Kode Program 5. 1 Implementasi Database LED Redis

Dari **Kode Program 5.1** ditunjukkan script untuk implementasi database *LED* Redis. Script di atas akan menghasilkan sebuah database Redis yang bertipe data *sorted sets*. Berikut penjelasan dari **Kode Program 5.1** :

1. Baris 1 digunakan untuk membuat objek Redis dan untuk membuat koneksi dengan server Redis
2. Baris 3 sampai 6 digunakan untuk membuat database dalam bentuk *sorted sets* dimana sorted sets tersebut mempunyai *key* bernama '*LED*' dengan 4 elemen yaitu 2,3,4, dan 5. Selain itu dalam *sorted sets* juga mempunyai nilai scores dimana nilai scores tersebut dapat bernilai 1 atau 0.

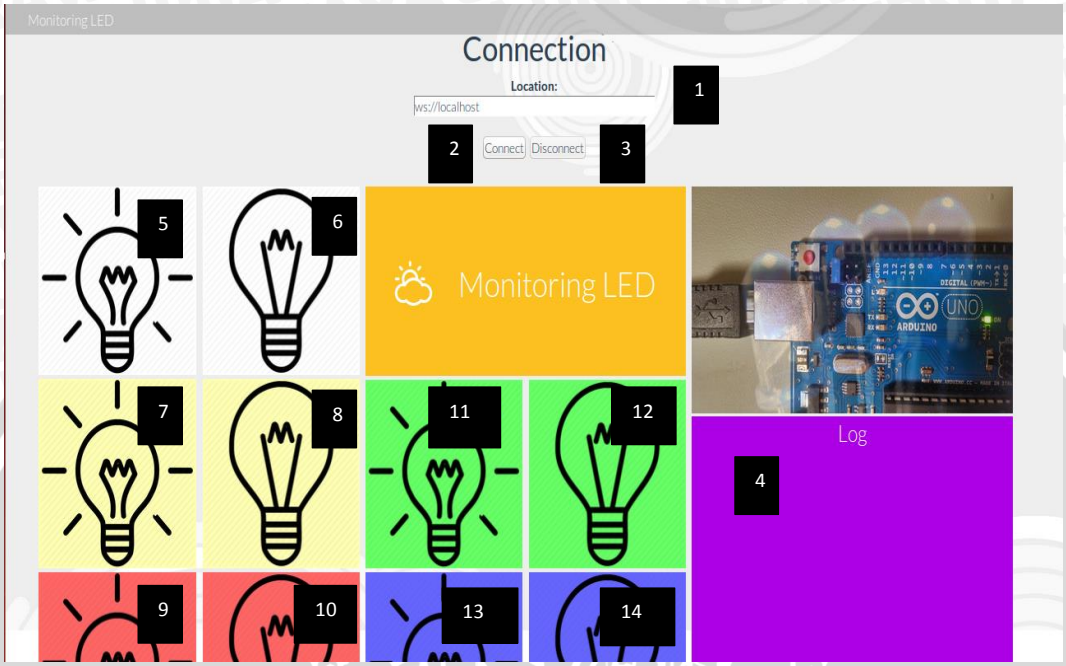
Gambar 5.2 menunjukkan bentuk database yang digunakan dalam kontrloing LED

```
[('3', 0.0), ('2', 1.0), ('4', 1.0), ('5', 1.0)]
```

Gambar 5. 2 Database Kontroling Lampu LED

5.1.3 Implementasi Aplikasi Web Client

Untuk membuat aplikasi web client, dibutuhkan sebuah script berbahasa *html* dan *javascript*. Dimana bahasa *html* digunakan untuk mengatur tampilan sistem yang nantinya digunakan oleh pengguna. Dan bahasa *javascript* digunakan untuk mengatur komunikasi dengan server menggunakan Websocket.



Gambar 5. 3 Implementasi Aplikasi Web Kontroling Lampu LED

Gambar 5.3 adalah tampilan web client yang nantinya digunakan pleh pengguna untuk menjalankan sistem. Sesuai dengan perancangan yang telah dibuat, maka dalam aplikasi web dibuat elemen yang mempunyai fungsi masing-masing yaitu

1. *Field* Connection : Digunakan untuk menuliskan alamat Websocket yang digunakan oleh server. Contohnya ws://ip_address:*port.*
2. Button Connect : Digunakan untuk melakukan koneksi dengan server dengan alamat yang sudah dituliskan pada *field* connection.
3. Button Disconnect : Digunakan untuk melakukan pemutusan koneksi dengan server.
4. *Field* Log Pesan : Digunakan untuk menampilkan pesan dari server.
5. Button Hidup Semua : Digunakan untuk mengirimkan pesan untuk menghidupkan semua *LED* kepada server.
6. Button Mati Semua : Digunakan untuk mengirimkan pesan untuk mematiakn semua *LED* kepada server.
7. Button Hidupkan *LED* 2 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 2 kepada server.

8. Button Matikan *LED* 2 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 2 kepada server.
9. Button Hidupkan *LED* 3 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 3 kepada server.
10. Button Matikan *LED* 3 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 3 kepada server.
11. Button Hidupkan *LED* 4 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 4 kepada server.
12. Button Matikan *LED* 4 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 4 kepada server.
13. Button Hidupkan *LED* 5 :Digunakan untuk mengirimkan pesan untuk menghidupakan *LED* pada pin 5 kepada server.
14. Button Matikan *LED* 5 : Digunakan untuk mengirimkan pesan untuk mematikan *LED* pada pin 5 kepada server.

Setelah tampilan aplikasi dibuat, untuk menjalankan fungsi-fungsi seperti melakukan koneksi dengan server, mengirimkan pesan ke server, menerima dan menampilkan pesan dari server, maka perlu dibuat sebuah script berbahasa *javascript* untuk mengatur fungsi-fungsi tersebut. Untuk medukung komunikasi Websocket dalam pengimplementasian Aplikasi Web menggunakan Webscoket API.

```
1.  function doConnect()
2.    {
3.      if (window.MozWebSocket)
4.      {
5.          logToConsole('<span style="color:
6.   red;"><strong>Info:</strong>
7.   Browser mendukung Websocket</span>');
8.          window.WebSocket = window.MozWebSocket;
9.      }
10.     else if (!window.WebSocket)
11.     {
12.         logToConsole('<span style="color:
13.   red;"><strong>Error:</strong>
14.   Browser tidak mendukung Websocket</span>');
15.         return;
16.     }
17.     var uri = wsUri.value;
18.     if (uri.indexOf("?") == -1) {
```

```
19.             uri += "?encoding=text";
20          } else {
21.             uri += "&encoding=text";
22.         }
23.         websocket = new WebSocket(uri);
24.         websocket.onopen = function(evt) { onOpen(evt) };
25.         websocket.onclose = function(evt) { onClose(evt) };
26.         websocket.onmessage = function(evt) { onMessage(evt) };
27.         websocket.onerror = function(evt) { onError(evt) };
28.       }
29.   function doDisconnect()
30.       {
31.         websocket.close()
32.       }
```

Kode Program 5. 2 Implementasi Websocket pada Aplikasi Web Kontroling LED

Penggalan script *javascript* dari **Kode Program 5.2** di atas menjelaskan pembuatan koneksi dan pemutusan koneksi dengan server. Berikut penjelasan dari **Kode Program 5.2**

1. Baris 3 sampai 16 menjelaskan jika *browser* yang digunakan tidak mendukung Websocket maka sistem akan menampilkan pesan "Browser tidak mendukung Websocket" pada *field* log pesan. Ketika pengguna sudah memasukkan alamat dengan benar dan pengguna mengklik tombol connect maka sistem akan menjalankan fungsi *doConnect*.
2. Baris 23 sampai 27 menjelaskan fungsi utama dari Websocket API, dimana dalam fungsi utama tersebut berguna untuk menangani *event* utama. *Event* utama tersebut terdiri dari 4 *event* yaitu *onopen* yang digunakan untuk membuat koneksi dengan server, *onclose* digunakan untuk memutuskan koneksi dengan server, *onmessage* digunakan untuk menangani sebuah *event* ketika ada pesan yang masuk ke client, dan *onerror* digunakan untuk menampilkan pesan *error* ketika terjadi kesalahan selama koneksi terbentuk. Apabila pengguna mengklik tombol disconnect maka sistem akan menjalankan fungsi *doDisconnect*, dimana fungsi tersebut digunakan untuk menutup koneksi dengan server.

5.1.4 Implementasi Pengiriman Pesan dari Client ke Server

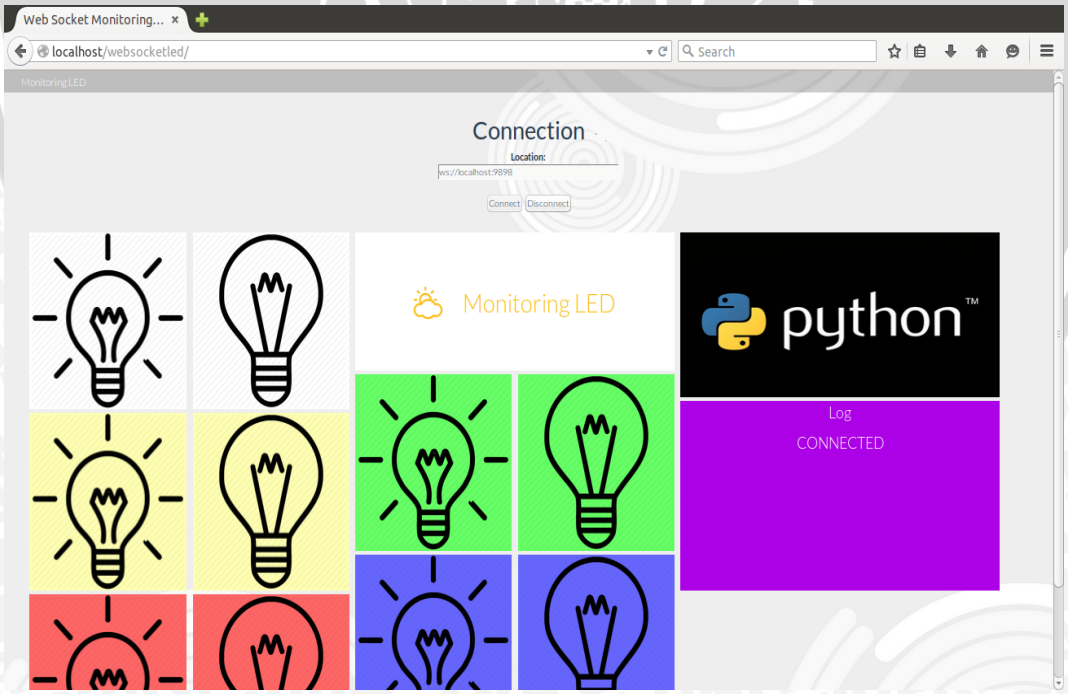
Dibutuhkan sebuah server sebagai penghubung antara client dengan Arduino Uno. Dalam pengimplementasian ini server dituliskan dalam bahasa python. Salah satu fungsi server dalam pengimplementasian kontroling lampu *LED* adalah menerima pesan dari client dan mengolah pesan tersebut. Supaya client dapat terhubung dengan server, pertama dilakukan pembuatan sebuah jalur komunikasi antara server dengan client.

```
1.  def main():
2.          factory = protocol.ServerFactory()
3.          factory.protocol = Echo
4.          reactor.listenTCP(9898,WebSocketFactory(factory))
5.          reactor.run()
```

Kode Program 5. 3 Pengimplementasian Websocket pada Server

Dalam **Kode Program 5.3** merupakan kode untuk pengimplementasian Websocket pada server. Berikut penjelasan dari kode di atas :

1. Pada baris 2 dan 3 adalah *script* yang digunakan untuk membuat *instance* dari kelas *protocol* ketika terdapat komunikasi baru.
2. Baris 4 adalah pembuatan jalur komunikasi yang berada pada *port* 9898. Dimana *port* ini wajib dituliskan oleh pengguna untuk melakukan koneksi dengan server. Pengguna dapat menuliskan alamat server pada *field* connection yang sudah dijelaskan pada implementasi aplikasi web client. Contoh alamat yang dapat dituliskan pengguna adalah ws://localhost:9898. Dimana localhost merupakan alamat IP dari server dan 9898 merupakan angka *port*. Setelah pengguna mengisikan alamat dengan benar dan mengklik tombol connect, maka sistem akan menampilkan pesan "Connected" pada log pesan seperti **Gambar 5.4**.



Gambar 5. 4 Pengimplementasian Websocket pada Server

```
1.  function doSend1()
2.    {
3.      logToConsole("SENT: mati semua");
4.      websocket.send("mati semua");
5.    }
6.  function doSend2()
7.    {
8.      logToConsole("SENT: nyala semua");
9.      websocket.send("nyala semua");
10.   }
11. function doSend3()
12.   {
13.     logToConsole("SENT: nyala 2");
14.     websocket.send("nyala 2");
15.   }
16. function doSend4()
17.   {
18.     logToConsole("SENT: mati 2");
19.     websocket.send("mati 2");
20.   }
21. function doSend5()
22.   {
23.     logToConsole("SENT: nyala 3");
24.     websocket.send("nyala 3");
25.   }
26. function doSend6()
27.   {
28.     logToConsole("SENT: mati 3");
29.     websocket.send("mati 3");
30.   }
31. function doSend7()
32.   {
33.     logToConsole("SENT: nyala 4");
34.     websocket.send("nyala 4");
35.   }
36. function doSend8()
37.   {
38.     logToConsole("SENT: mati 4");
39.     websocket.send("mati 4");
40.   }
41. function doSend9()
```

```
42.     {
43.       logToConsole("SENT: nyala 5");
44.       websocket.send("nyala 5");
45.     }
46.   function doSend10()
47.     {
48.       logToConsole("SENT: mati 5");
49.       websocket.send("mati 5");
50.     }
```

Kode Program 5. 4 Proses Pengiriman Pesan dari Client ke Server

Tujuan utama dari sistem adalah mampu mengontrol lampu *LED* menggunakan Websocket. Untuk itu diperlukan adanya pengiriman pesan yang dilakukan client kepada server. Pada **Kode Program 5.4** dijelaskan proses pengiriman pesan menggunakan Websocket. Fungsi pengiriman hanya bisa dilakukan jika pengguna sudah melakukan pembuatan koneksi dengan server dengan benar. Berikut penjelasan dari kode program di atas :

1. Baris 1 sampai 50 terdapat 10 macam pengiriman pesan kepada server, dimana setiap fungsi pengiriman mengirim pesan yang berbeda-beda. Fungsi pengiriman akan dilakukan ketika pengguna mengklik salah satu button bergambar lampu yang terdapat pada aplikasi web.

5.1.5 Implementasi Pengolahan Pesan

Setelah pengguna mengirimkan pesan dengan mengklik salah satu button, maka server perlu menyeleksi pesan yang dikirimkan oleh client. Fungsi penyeleksian dilakukan untuk mengolah pesan menjadi sebuah *integer*. Dimana pesan yang dikirimkan oleh client adalah *string* dan akan diubah menjadi nilai *integer*. Dan nilai *integer* tersebutlah yang akan dikirimkan oleh server ke Arduino. **Kode Program 5.5** menjelaskan kode untuk penyeleksian pesan.

```
1.   def dataReceived(self, data):
2.         msg = ''
3.         date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
4.         kata = data.split()
5.         print data
6.         if (kata[0]=="nyala"):
7.               if (kata[1]=="semua"):
8.                     nyalasemua()
9.                     self.transport.write('Date :' + date)
10.              else:
11.                    nyala(kata[1])
12.                    self.transport.write('Date :' + date)
13.                    print "nyala satu berhasil"
```

```
14.         elif (kata[0]=="mati"):
15.                 if (kata[1]=="semua"):
16.                         matisemua()
17.                         self.transport.write('Date :' + date)
18.                 else:
19.                         mati(kata[1])
20.                         self.transport.write('Date :' + date)
21.                         print "mati satu berhasil"
22.         else:
23.                 self.transport.write("command yang dimasukkan
24.  salah")
```

Kode Program 5. 5 Proses Penyeleksian Pesan

Dari **Kode Program 5.5**, dapat dijelaskan sebagai berikut :

1. Baris 6 sampai 13 dapat dijelaskan bila pesan yang diterima server “nyala semua” maka server akan mengerjakan fungsinyalasemua(). Jika “hidup (sebuah *integer*)” serperti contohnya “hidup 2” maka akan menjalankan fungsi hidup() dengan parameter kata kedua.
2. Baris 14 sampai 21 dapat dijelaskan jika “mati semua” maka server akan mengerjakan fungsi matisemua().Dan jika “mati (sebuah *integer*)” serperti contohnya “mati 2” maka akan menjalankan fungsi mati() dengan parameter kata kedua.

Semua fungsi tersebut digunakan untuk mengubah nilai status *LED* yang tersimpan dalam database Redis. Selain melakukan fungsi berikutnya, server juga mengirimkan pesan berupa waktu eksekusi kepada client untuk menunjukkan kapan waktu eksekusi.

```
1.  def nyala(id_lampu):
2.          stat = 1
3.          r_server.zadd('LED', id_lampu, stat)
4.          refresh()
5.          return stat
6.
7.  def nyalasemua():
8.          stat = 1
9.          r_server.zadd('LED', 2, stat)
10.         r_server.zadd('LED', 3, stat)
11.         r_server.zadd('LED', 4, stat)
12.         r_server.zadd('LED', 5, stat)
13.         refresh()
14.         return stat
15.
```

```
16.  def mati(id_lampu):
17.          stat = 0
18.          r_server.zadd('LED', id_lampu, stat)
19.          refresh()
20.          return stat
21.
22.  def matisemua():
23.          stat = 0
24.          r_server.zadd('LED', 2, stat)
25.          r_server.zadd('LED', 3, stat)
16.          r_server.zadd('LED', 4, stat)
27.          r_server.zadd('LED', 5, stat)
28.          refresh()
29.          return stat
```

Kode Program 5. 6 Proses Pembaharuan Database Redis

Pada **Kode Program 5.6** dapat dijelaskan sebagai berikut :

1. Baris 1 sampai 5 adalah *method* yang digunakan untuk mengubah salah satu status lampu *LED* menjadi bernilai 1. Dimana status lampu *LED* yang diubah sesuai dengan pesan yang diterima server.
2. Baris 7 sampai 14 adalah method yang digunakan untuk merubah semua status lampu *LED* menjadi bernilai 1.
3. Baris 16 sampai 20 adalah *method* yang digunakan untuk mengubah salah satu status lampu *LED* menjadi bernilai 0. Dimana status lampu *LED* yang diubah sesuai dengan pesan yang diterima server.
4. Baris 22 sampai 29 adalah method yang digunakan untuk merubah semua status lampu *LED* menjadi bernilai 0.

4 fungsi di atas digunakan untuk mengubah nilai yang terdapat pada database Redis. Kode dijalankan sesuai dengan pesan yang diterima oleh server dari client. Setelah server berhasil mengubah nilai dalam *sorted sets* "*LED*", server mengerjakan proses pengolahan pesan yang direpresentasikan pada fungsi refresh().

Untuk dapat mengontrol sebuah lampu *LED* yang terpasang pada Arduino Uno, Arduino memerlukan sebuah pesan dari server. Dimana pesan tersebut dapat menunjukkan lampu mana saja yang harus dimatikan atau dihidupkan. Maka dari itu setelah server menerima pesan berupa *string* dari client, server akan mengolah pesan tersebut menjadi sebuah bilangan atau *integer* dan mengirimkannya kepada Arduino.

```
1.  def refresh():
2.              global arduino
3.              found = r_server.zrange('LED', 0, -1,
```

```
4.    withscores=True)
5.                found.sort()
6.                x = PrettyTabel(["ID","Status"])
7.                x.align["ID"] = "l"
8.                x.padding_width =
9.                msg=""
10.               i = 1
11.               val = 0
12.               for row in found:
13.                       x.add_row([row[0],row[1]])
14.                       val = val + (i*row[1])
15.                       message = int(val)
16.                       i = i *2
17.                       msg = str(message)
18.               arduino.write(msg);
```

Kode Program 5. 7Proses Pengolahan Pesan

Dalam **Kode Program 5.7** menjelaskan tentang kode untuk pengolahan pesan. Dari kode di atas dapat dijelaskan sebagai berikut :

1. Baris 3 dan 4 digunakan untuk mengambil semua nilai yang terdapat pada *sorted sets* "LED".
2. Baris 5 sampai 7 digunakan untuk menampilkan semua data dalam database kedalam bentuk tabel. Untuk lebih jelasnya **Gambar 5.5** menunjukkan tabel yang berisi data dari database Redis.
3. Baris 11 sampai 16 adalah proses perulangan yang digunakan untuk menentukan nilai *integer* yang akan dikirimkan kepada Arduino. Setelah pesan berhasil terbentuk, server akan mengirimkan pesan tersebut ke Arduino menggunakan fungsi *write*.

Gambar 5. 5 Hasil Proses Pengolahan Pesan

Dalam **Gambar 5.5** dapat dijelaskan bahwa status semua *LED* bernilai 1 atau hidup. Untuk membuat pesan dalam bentuk bilangan, maka server melakukan perulangan untuk menentukan bilangan yang terbentuk dari data yang telah ditampilkan. Bila data yang didapatkan seperti pada **Gambar 5.5** maka pesan yang akan dikirimkan ke Arduino adalah 15. Dengan nilai status yang hanya bisa bernilai 1 atau 0, maka dapat dipastikan pesan yang terbentuk mempunyai jarak antara 0

sampai 15. **Tabel 5.1** menampilkan semua kemungkinan pesan yang dapat dikirimkan oleh server kepada Arduino Uno.

Tabel 5. 1 Daftar Pesan yang Dikirimkan Server ke Arduino

Data Redis (*LED*, status)	Pesan yang akan dikirimkan
(2, 0), (3, 0), (4, 0), (5, 0)	0
(2, 1), (3, 0), (4, 0), (5, 0)	1
(2, 0), (3, 1), (4, 0), (5, 0)	2
(2, 1), (3, 1), (4, 0), (5, 0)	3
(2, 0), (3, 0), (4, 1), (5, 0)	4
(2, 1), (3, 0), (4, 1), (5, 0)	5
(2, 0), (3, 1), (4, 1), (5, 0)	6
(2, 1), (3, 1), (4, 1), (5, 0)	7
(2, 0), (3, 0), (4, 0), (5, 1)	8
(2, 1), (3, 0), (4, 0), (5, 1)	9
(2, 0), (3, 1), (4, 0), (5, 1)	10
(2, 1), (3, 1), (4, 0), (5, 1)	11
(2, 0), (3, 0), (4, 1), (5, 1)	12
(2, 1), (3, 0), (4, 1), (5, 1)	13
(2, 0), (3, 1), (4, 1), (5, 1)	14
(2, 1), (3, 1), (4, 1), (5, 1)	15

5.1.6 Implementasi Eksekusi Perintah oleh Arduino

Untuk dapat mengirimkan pesan tersebut ke Arduino, diperlukan adanya komunikasi serial antara server dengan Arduino. Setelah komunikasi serial terbentuk server dapat mengirimkan pesan menggunakan fungsi write yang sudah dijelaskan sebelumnya.

```
1.  arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=.1)
2.  time.sleep(3)
3.  print ('Server Ready....')
4.  refresh()
```

Kode Program 5. 8 Pembentukan Komunikasi Serial pada Server

Kode Program 5.8 menjelaskan script yang digunakan pada sistem untuk membentuk komunikasi Serial dengan Arduino. Dari kode di atas dapat dijelaskan sebagai berikut :

1. Baris 1 adalah script pembuatan jalur komunikasi dengan Arduino. Untuk melakukan komunikasi serial dibutuhkan informasi tentang port USB yang dipakai Arduino dan baud rate yang dipakai Arduino. Dengan memasukkan port USB dan baud rate yang sama dengan yang dipakai Arduino, maka komunikasi serial dapat terbentuk.

```
1.  void loop() {
2.    if (Serial.available()>0)
3.    {
4.       char buffer[] = {' ',' '};
5.       while (!Serial.available());
6.       Serial.readBytesUntil('n', buffer, 2);
7.       int incomingByte = atoi(buffer);
8.       Serial.println(incomingByte);
9.       if (incomingByte == 0)
10.      {
11.        digitalWrite(2, HIGH);
12.        digitalWrite(3, HIGH);
13.        digitalWrite(4, HIGH);
14.        digitalWrite(5, HIGH);
15.      }
16.      if (incomingByte == 1)
17.      {
18.        digitalWrite(2, LOW);        digitalWrite(3, HIGH);
19.        digitalWrite(4, HIGH);
20.        digitalWrite(5, HIGH);
21.      }
```

Kode Program 5. 9 Proses Eksekusi Perintah oleh Arduoino

Kode Program 5.9 menjelakan kode yang dituliskan pada Arduino Uno. Kode dituliskan menggunakan bahasa C. Kode tersebut dituliskan dan di unggah menggunakan Arduino IDE. Dari penggalan kode di atas dapat dijelaskan sebagai berikut :

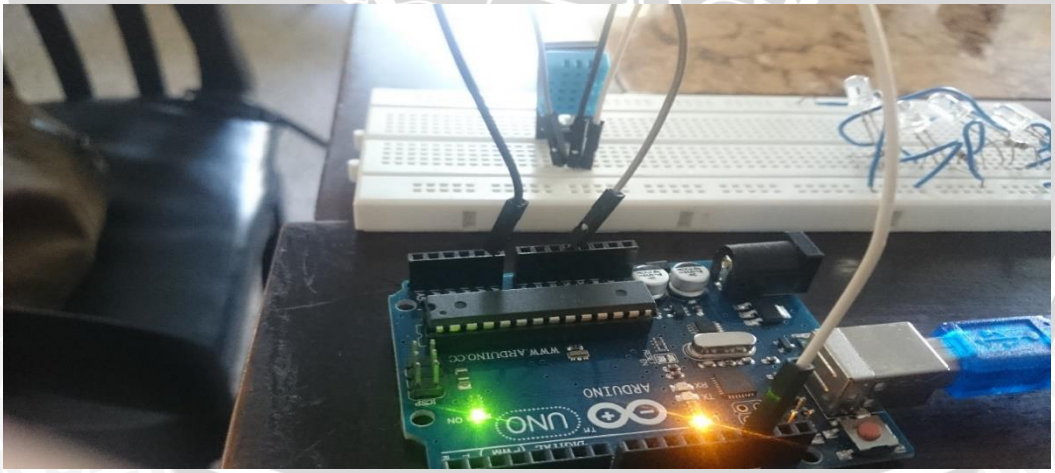
1. Baris 2 sampai 7 adalah proses bagaimana Arduino menerima pesan. Supaya Arduino dapat menerima pesan sampai 2 bilangan maka perlu dibentuk sebuah array untuk menampung 2 bilangan tersebut.
2. Baris 9 sampai 21 adalah penggalan dari script untuk mengeksekusi perintah. Terdapat 15 kondisi dimana masing-masing kondisi akan menampilkan output yang berbeda. Digitalwrite(2, HIGH) dalam kode program mempunyai arti bahwa *LED* yang terpasang pada pin nomor 2 tegangan *list*riknya akan diputuskan.

5.2 Implementasi Monitoring Suhu dan Kelembapan

Implementasi monitoring suhu dan kelembapan menjelaskan bagaimana untuk mengimplementasikan Websocket untuk memonitoring suhu dan kelembapan dari sensor DHT11 yang terpasang pada Arduino Uno. Sesuai dengan perancangan, dalam pengimplementasian kontroling lampu *LED* meliputi, implementasi perangkat keras, implementasi database redis, implementasi aplikasi web client, implementasi komunikasi server dengan Arduino, dan implementasi komunikasi server dengan client.

5.2.1 Implementasi Perangkat Keras

Berdasarkan perancangan perangkat keras yang telah dibuat sebelumnya, sebuah sensor DHT 11 akan dipasangkan pada board. Sensor DHT 11 membutuhkan 3 komponen yaitu vcc, data, dan ground. Dengan menggunakan bantuan dari kable jumper, sensor DHT 11 akan dihubungkan pada pin-pin yang terdapat pada Arduino Uno.



Gambar 5. 6 Konfigurasi Sensor DHT 11

Dari **Gambar 5.6** dapat dilihat bahwa pin VCC dihubungkan pada pin power bertegangan 5 V. Pin data terhubungkan pada pin Analog In A0, dan pin ground pada DHT 11 dihubungkan pada digital PMW GND. Dengan dihubungkannya DHT11 dengan power 5 V berarti tegangan *list*rik yang akan didapatkan oleh sensor akan sebesar 5V. Untuk dapat mengintegrasikan Arduino Uno dengan laptop maka digunakan 1 buah kabel USB yang dipasang pada Arduino dan dihubungkan pada *port* USB laptop.

5.2.2 Implementasi Database Redis

Untuk dapat membuat sistem monitoring suhu dan kelembapan yang *real time*, dibutuhkan sebuah media penyimpanan yang dapat menampung data-data yang didapatkan oleh sensor. Dalam studi kasus ini dibuat media penyimpanan Redis, dimana untuk membuat database tersebut dibutuhkan script yang dituliskan dalam bahasa python. Akses untuk mendapatkan data dan mengubah data yang tersimpan dalam Redis hanya dipunyai oleh server.

```
1.  r_server = redis.Redis('localhost')
2.  r_server.rpush('temp', int(processed_data[1]))
3.  r_server.rpush('humid', int(processed_data[0]))
4.  r_server.rpush('time', date)
```

Kode Program 5. 10 Implementasi Database Redis untuk Monitoring Suhu dan Kelembapan

Kode Program 5.10 merupakan *script* untuk pembuatan database Redis. Dari penggalan kode di atas dapat dijelaskan :

1. Baris 1 digunakan untuk membuat objek Redis dan melakukan koneksi dengan server Redis.
2. Baris 2 sampai 4 merupakan *script* untuk menghasilkan 3 database Redis yang bertipe data *lists*. Dalam studi kasus ini dibutuhkan 3 *lists* yaitu *lists* 'temp' yang berisi nilai suhu dari sensor DHT11, 'humid' yang berisi nilai kelembapan dari sensor DHT11, dan 'time' yang berisi informasi berupa waktu kapan data suhu dan kelembapan berhasil didapatkan oleh sensor.

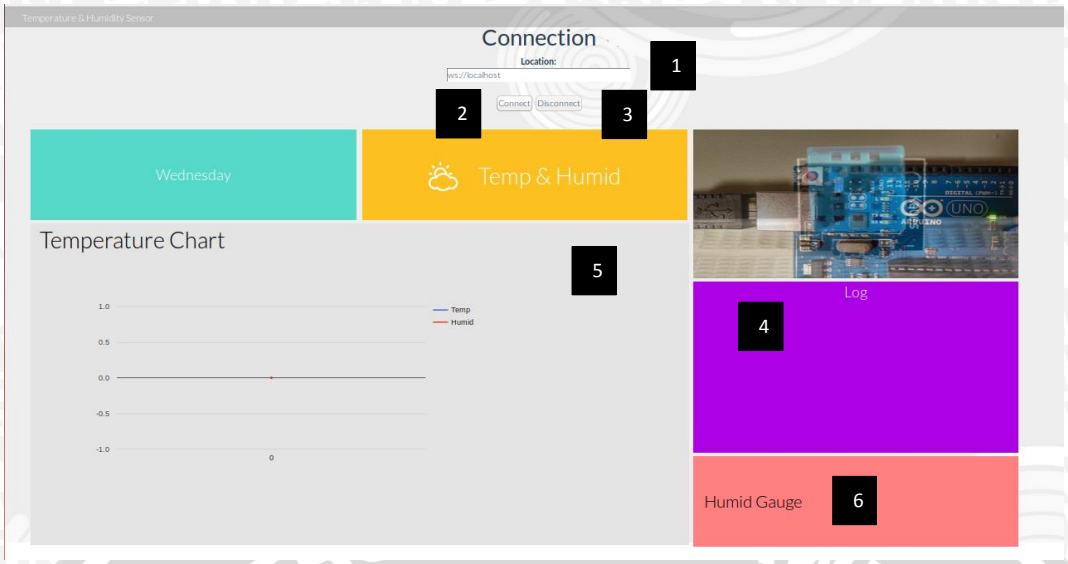
```
['28', '27', '27', '28', '27', '27']
['2016-02-29 15:05:05', '2016-02-29 15:06:08', '2016-02-29 15:30:12', '2016-02-29 15:05:14', '2016-02-29 15:06:16', '2016-02-29 15:30:18']
['32', '33', '32', '32', '33', '32']
```

Gambar 5. 7 Bentuk Database Redis 'temp', 'time', dan 'humid'

Pada **Gambar 5.7** ditunjukkan bentuk *list* 'temp' 'time' dan 'humid'. Baris pertama pada gambar menunjukkan *list* 'temp', baris kedua menunjukkan *list* 'time' dan baris ketiga menunjukkan *list* 'humid'.

5.2.3 Implementasi Aplikasi Web Client

Untuk membuat aplikasi web *client*, dibutuhkan sebuah script berbahasa *html* dan *javascript*. Dimana bahasa *html* digunakan untuk mengatur tampilan sistem yang nantinya digunakan oleh pengguna. Dan bahasa *javascript* digunakan untuk mengatur komunikasi dengan server menggunakan Websocket dan menampilkan 2 buah grafik.



Gambar 5. 8 Implementasi Aplikasi Web Monitoring Suhu dan Kelembapan

Gambar 5.8 adalah tampilan web client yang nantinya digunakan pleh pengguna untuk menjalankan sistem .Sesuai dengan perancangan yang telah dibuat, maka dalam aplikasi web dibuat elemen yang mempunyai fungsi masing-masing yaitu

1. *Field* Connection : Digunakan untuk menuliskan alamat Websocket yang digunakan oleh server. Contohnya ws://ip_address:*port*
2. Button Connect : Digunakan untuk melakukan koneksi dengan server dengan alamat yang sudah dituliskan pada *field* connection.
3. Button Disconnect : Digunakan untuk melakukan pemutusan koneksi dengan server
4. *Field* Log Pesan : Digunakan untuk menampilkan pesan dari server.
5. *Field* Grafik Garis Temperatur : Digunakan untuk menampilkan data tempertaur yang dikirimkan oleh server dan direpresentasikan dalam bentuk grafik garis.
6. *Field* Grafik Kelembapan : Digunakan untuk menampilkan data kelembapan yang dikrimkan oleh server dan direpresentasikan dalam bentuk grafik speedometer.

Setelah tampilan aplikasi dibuat, untuk menjalankan fungsi-fungsi seperti melakukan koneksi dengan server, menerima pesan dari server, menampilkan pesan dari server, dan menampilkan grafik, maka perlu dibuat sebuah script berbahasa *javascript* untuk mengatur fungsi-fungsi tersebut. Supaya client dapat berkomunikasi dengan server menggunakan Websocket, maka dalam aplikasi web perlu adanya pengimplementasian Websocket API.

```
1.  function doConnect()
2.    {
```

```
3.      if (window.MozWebSocket)
4.      {
5.          logToConsole('<span style="color:
6.    red;"><strong>Info:</strong> Browser mendukung
7.    Websocket</span>');
8.          window.WebSocket = window.MozWebSocket;
9.      }
10.     else if (!window.WebSocket)
11.     {
12.         logToConsole('<span style="color:
13.   red;"><strong>Error:</strong> Browser tidak mendukung
14.   Websocket</span>');
15.         return;
16.     }
17.     var uri = wsUri.value;
18.     if (uri.indexOf("?") == -1) {
19.         uri += "?encoding=text";
20      } else {
21.         uri += "&encoding=text";
22.     }
23.     websocket = new WebSocket(uri);
24.     websocket.onopen = function(evt) { onOpen(evt) };
25.     websocket.onclose = function(evt) { onClose(evt) };
26.     websocket.onmessage = function(evt) { onMessage(evt) };
27.     websocket.onerror = function(evt) { onError(evt) };
28.   }
29.  function doDisconnect()
30.   {
31.     websocket.close()
32.   }
```

Kode Program 5. 11 Pengimplementasian Websocket pada Aplikasi Web

Penggalan script *javascript* dari **Kode Program 5.11** di atas menjelaskan pembuatan koneksi dan pemutusan koneksi dengan server. Dari kode di atas dapat dijelaskan :

1. Baris 3 sampai 16 menjelaskan jika *browser* yang digunakan tidak mendukung Websocket maka sistem akan menampilkan pesan “Browser tidak mendukung Websocket” pada *field* log pesan. Ketika pengguna sudah memasukkan alamat dengan benar dan pengguna mengklik tombol connect maka sistem akan menjalankan fungsi *doConnect*.
2. Baris 23 sampai 27 menjelaskan fungsi utama dari Websocket API, dimana dalam fungsi utama tersebut berguna untuk menangani *event* utama.

Event utama tersebut terdiri dari 4 *event* yaitu *onopen* yang digunakan untuk membuat koneksi dengan server, *onclose* digunakan untuk memutuskan koneksi dengan server, *onmessage* digunakan untuk menangani sebuah *event* ketika ada pesan yang masuk ke client, dan *onerror* digunakan untuk menampilkan pesan *error* ketika terjadi kesalahan selama koneksi terbentuk.

3. Baris 29 sampai 31 digunakan apabila pengguna mengklik tombol disconnect maka sistem akan menjalankan fungsi *doDisconnect*, dimana fungsi tersebut digunakan untuk menutup koneksi dengan server.

```
1.  google.load('visualization', '1', {'packages':['corechart',
2.   'gauge']});
3.  google.charts.load('current', {'packages':['corechart', 'gauge']});
4.  function drawChart(evt) {
5.        logToConsole('<span style="color:
6.   white;">'+evt.data+'</span>');
7.        var data = new google.visualization.DataTabel(evt.data);
8.        var options = {
9.        'width' :800,
10.       'backgroundColor': '#E4E4E4',
11.       'height':400
12.         };
13.       Var chart = new
14. google.visualization.LineChart(document.getElemenById('chart_div'));
15.       chart.draw(data, options);
16.       var options1 = {
17.           width: 200, height: 150,
18.           redFrom: 90, redTo: 100,
19.           yellowFrom:75, yellowTo: 90,
20.           minorTicks: 5
21.         };.
22.       var chart1 = new
23. google.visualization.Gauge(document.getElemenById('gauge_div'));
24.       chart1.draw(data, options1);
25.    }
```

Kode Program 5. 12 Pengimplementasian Google API pada Aplikasi Web

Kode Program 5.12 merupakan kode untuk menampilkan data ke dalam bentuk grafik. Untuk menampilkannya, dalam pengimplementasian menggunakan Google API. Grafik akan ditampilkan jika pengguna sudah terkoneksi dengan server dengan benar. Dari kode di atas dapat dijelaskan :

1. Baris 3 sampai 13 adalah proses pembentukan grafik garis.

2. Baris 14 sampai 22 adalah proses pembentukan grafik speedometer.
3. Baris 5 dapat dijelaskan bahwa untuk membuat grafik dibutuhkan sebuah data yang berasal dari server, dimana Google API membutuhkan data bertipe JSON. Grafik garis akan menampilkan data suhu beserta waktu sedangkan grafik speedometer akan menampilkan data kelembapan.

5.2.4 Implementasi Pengiriman Data dari Arduino ke Server

Dalam gambaran topologi yang sudah dibuat dalam perancangan, dalam implementasi ini dibutuhkan 2 buah server. Server yang pertama berfungsi untuk menerima data suhu dan kelembapan dari Arduino dan menyimpannya ke dalam Redis. Dan server yang kedua berfungsi untuk mengolah data suhu dan kelembapan yang sudah tersimpan dalam Redis dan mengirimkannya kepada client. Untuk dapat menjalankan sistem dengan benar, pertama data suhu dan kelembapan perlu didapatkan.

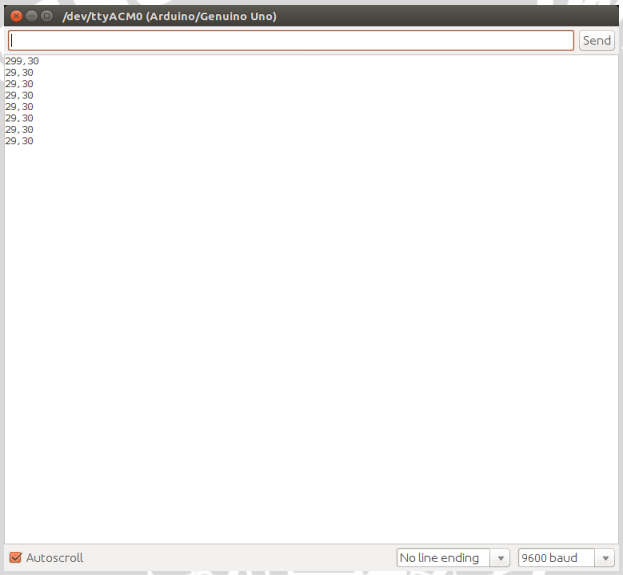
```
1.  void loop(){
2.    ReadDHT();
3.    switch (bGlobalErr){
4.      case 0:
5.       Serial.print(dht_dat[0], DEC);
6.       Serial.print(",");
7.       Serial.println(dht_dat[2], DEC);
8.           break;
9.        case 1:
10.          Serial.println("Error 1: DHT start condition 1 not met.");
11.          break;
12.       case 2:
13.          Serial.println("Error 2: DHT start condition 2 not met.");
14.          break;
15.       case 3:
16.          Serial.println("Error 3: DHT checksum error.");
17.          break;
18.       default:
19.          Serial.println("Error: Unrecognized code encountered.");
20.          break;
21.        }
22.   delay(1000);
23. }
```

Kode Program 5. 13 Proses Pengambilan Nilai Suhu dan Kelembapan oleh Sensor DHT 11

Kode Program 5.13 menjelaskan kode untuk mendapatkan data suhu dan kelembapan melalui DHT 11. Kode program dituliskan menggunakan bahasa C,

dimana kode tersebut dituliskan dan di unggah menggunakan Arduino IDE. Script yang telah dituliskan menggunakan sebuah *library* dari DHT, dimana *library* tersebut digunakan untuk membaca keadaan seuhu dan kelembapan dari lingkungan. Dari kode di atas dapat dijelaskan :

1. Baris nomor 2 adalah script untuk memanggil fungsi yang ada dalam *library* DHT untuk mendapatkan nilai suhu dan kelembapan.
2. Baris 4 sampai 20 adalah pengkondisian dimana proses mendapatkan nilai suhu dan kelembapan dilakukan setelah pemasangan DHT 11 pada Arduino sudah benar. Apablia masih terjadi kesalahan pemasangan Arduino tidak akan mendapatkan nilai suhu dan kelembapan, melainkan akan menampilkan pesan error pada Serial Monitor. Data yang telah didapatkan sensor DHT 11 akan ditampilkan pada serial monitor, dimana data suhu dan kelembapan akan dipisahakan dengan tanda koma.



Gambar 5. 9 Hasil Proses Pengambilan Nilai Suhu dan Kelembapan oleh Sensor DHT11

Gambar 5.9 di atas menunjukkan contoh keluaran yang dihasilkan oleh Arduino. Angka sebelum koma menunjukkan nilai kelembapan, dan angka sesudah koma menunjukkan nilai suhu. Supaya sistem dapat berjalan secara *real time*, maka proses mendapatkan nilai suhu dan kelembapan dilakukan Arduino setiap detiknya sampai sistem berhenti.

Setelah Arduino mencetak nilai suhu dan kelembapan, maka nilai suhu tersebut akan dikirimkan kepada server. Data yang dikirimkan ke server akan sama dengan apa yang dicetak oleh Arduino pada serial monitor. Untuk dapat melakukan koneksi dengan server, terlenbih dahulu dibutuhkan sebuah komunikasi serial antara server dengan Arduino. Selain terdapat proses penerimaan data dari Arduino, server juga melakukan proses penyimpan data tersebut ke dalam Redis.

```
1.  arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
2.  arduino.flush()
3.  arduino.flushInput()
4.  time.sleep(3)
```

Kode Program 5. 14 Pembentukan Komunikasi Serial pada Server

Kode Program 5.14 menjelaskan script yang digunakan pada sistem untuk membentuk komunikasi Serial dengan Arduino.

1. Baris 1 adalah script pembuatan jalur komunikasi dengan Arduino. Untuk melakukan komunikasi serial dibutuhkan informasi tentang port USB yang dipakai Arduino dan baud rate yang dipakai Arduino. Dengan memasukkan port USB dan baud rate yang sama dengan yang dipakai Arduino, maka komunikasi serial dapat terbentuk.
2. Baris 2 dan 3 digunakan untuk membersihkan *buffer* supaya mengurangi tingkat kesalahan pengiriman data dari Arduino.

```
1.  def getData(x):
2.          import time
3.          time.sleep(2)
4.          print('Server Ready....')
5.          i = 0
6.          while (i<4):
7.                  import time
8.                 global arduino
9.                  print arduino.readline()
10.                 date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
11.                 list = arduino.readline()
12.                 line = list.strip()
13.                 processed_data = line.split(",")
14.                 print date
15.                 status= len(processed_data)
16.                 print len(processed_data), processed_data
17.                 if (status == 2):
18.                         r_server.rpush('temp',
19. int(processed_data[1]))
20.                         r_server.rpush('humid',
21. int(processed_data[0]))
22.                         r_server.rpush('time', date)
23.                 else:
24.                         print 'Data dari Arduino salah'
                            arduino.readline()
```

```
25.                     continue
```

Kode Program 5. 15 Proses Penyimpanan Data ke Dalam Database Redis

Kode pada **Kode Program 5.15** menjelaskan tentang proses penerimaan data dari Arduino Uno dan proses penyimpanan data ke dalam Redis. Dari kode di atas dapat dijelaskan :

1. Baris 8 dan 9 digunakan untuk menerima pesan dan menampilkan pesan dari Arduino.
2. Baris 12 dan 13 digunakan untuk memisahkan data yang masuk berdasarkan tanda koma.
3. Baris 17 sampai 24 merupakan pengkondisian yang digunakan untuk menghindari sebuah data yang tidak lengkap dari Arduino, server melakukan pengecekan terlebih dahulu apakah data yang diterima berjumlah 2 nilai atau tidak. Jika benar maka server akan melakukan proses penyimpanan, dimana nilai sesudah koma akan disimpan ke dalam *lists* 'temp', nilai sebelum koma akan disimpan ke dalam *lists* 'humid', dan nilai wwaktu akan disimpan dalam *lists* 'date'. Apablia data yang diterima tidak berjumlah 2 nilai maka server tidak akan melakukan proses penyimpanan dan langsung mengulangi proses penerimaan data.

5.2.5 Implementasi Pengolahan Pesan

Selain membutuhkan server untuk menangani komunikasi dengan Arduino, diperlukan juga sebuah server untuk menangani server dengan client. Fungsi server ini adalah untuk mendapatkan niali dari Redis dan mengolahnya data tersebut menjadi sebuah data bertipe JSON. Dimana nantinya data JSON tersebut akan dirubah aplikasi web client menjadi grafik.

```
1.  def sendData(self):
2.                  i = 0
3.                  print clients
4.                  while (i <4) :
5.                        import time
6.                        time.sleep(1.2)
7.                        data = r_server.lrange('temp', -1, -1)
8.                        jam = r_server.lrange('time', -1, -1)
9.                        tekan = r_server.lrange('humid', -1, -1)
10.                       temp = r_server.lrange('temp', 0, -1)
11.                       time = r_server.lrange('time', 0, -1)
12.                       tekanan = r_server.lrange('humid', 0, -1)
13.                       temperature = map(int, temp)
14.                       humid = map(int, tekanan)
15.                       hum = map(int, tekan)
16.                       tabel = Tabel()
```

```
17.                     tabel.add_column('date', str, "Date")
18.                     tabel.add_column('temp', int, "Temp")
19.                     #tabel.add_column('humidity', int, "Humid")
20.                     for temp, waktu in zip(time, temperature):
21.                             tabel.append([temp, waktu])
22.                     json = encode(tabel)
23.                     tabel1 = Tabel()
24.                     tabel1.add_column('humidity', str, 'Humid')
25.                     tabel1.add_column('value', int, 'Value')
26.                     for value in zip(hum):
27.                             tabel1.append(['humid', value])
28.                     json1 = encode(tabel1)
29.                     hasil = ', '.join(data)
30.                     hasil1 = ', '.join(jam)
31.                     hasil2 = ', '.join(tekan)
32.                     x = int (hasil)
33.                     fahrenheit = x * 1.8 + 32
34.                     a = int(fahrenheit)
35.                     fah = str(a)
36.                     kelvin = x + 273
37.                     kel = str(kelvin)
38.                     for cl in clients :
39.                     #while (i < 4 ):
40.                             import time
41.                             #print(data)
42.                             cl.transport.write(json1)
43.                             cl.transport.write(json)
44.                             cl.transport.write('Date          :' +
45.  hasil1)
46.                             cl.transport.write('Temp          :' +
47.   hasil + "C")
48.                             cl.transport.write('Temp (F)      :' +
49.   fah + "F")
50.                             cl.transport.write('Temp (K)      :' +
51.   kel + "K")
52.                             cl.transport.write('Humid         : ' +
53.   hasil2 + "%")
```

Kode Program 5. 16 Proses Pengolahan Pesan dan Pengiriman Pesan ke Client

Pada **Kode Program 5.16** dijelaskan tentang proses pengolahan pesan dan pengiriman pesan dari server ke client. Dari kode di atas dapat dijelaskan :

1. Baris 7 sampai 9 menunjukkan kode yang digunakan untuk mendapatkan nilai terkahir dari *list* 'temp', 'humid', dan 'time'.
2. Baris 10 sampai 12 digunakan untuk mendapatkan semua nilai yang terdapat pada list 'temp', 'humid', dan 'time'.
3. Baris 16 sampai 22 digunakan untuk membentuk pesan JSON, dimana JSON tersebut berisikan semua data yang tersimpan dalam *lists* 'temp' dan 'date'. Data dari *lists* 'temp' dan 'date' nantinya akan digabungkan menjadi satu data JSON, dimana satu data 'date' akan mewakili satu data dari 'temp'. Contoh hasil JSON yang dihasilkan dapat dilihat pada **Gambar 5.10.**
4. Baris 23 sampai 28 digunakan untuk membentuk pesan JSON yang kedua, dimana JSON tersebut berisikan semua data yang tersimpan dalam *lists* 'humid'. Contoh hasil JSON kedua yang dihasilkan dapat dilihat pada **Gambar 5.11.**
5. Baris 42 sampai 53 digunakan untuk mengirim pesan yang telah dibuat ke client. Jumlah pesan yang akan dikrimkan oleh server kepada client berjumlah 7 pesan, dimana 2 pesan bertipe JSON dan 5 pesan bertipe *string*. Pesan JSON tersebut nanti akan dirubah menjadi grafik garis dan grafik speedometer, sedangkan pesan bertipe *string* akan langsung ditampilkan pada *field* log pesan yang terdapat pada web client.

{"cols":[{"label":"Datetime","type":"string"},{"label":"Temp","type":"number"}],"rows":[{"c":[{"v":"2016-02-29 15:05:05"},{"v":28}]},{"c":[{"v":"2016-02-29 15:06:08"},{"v":27}]},{"c":[{"v":"2016-02-29 15:30:12"},{"v":27}]},{"c":[{"v":"2016-02-29 15:05:14"},{"v":28}]},{"c":[{"v":"2016-02-29 15:06:16"},{"v":27}]},{"c":[{"v":"2016-02-29 15:30:18"},{"v":27}]}]}

Gambar 5. 10 JSON untuk Grafik Garis

{"cols":[{"label":"Humid","type":"string"},{"label":"Value","type":"number"}],"rows":[{"c":[{"v":"humid"},{"v":32}]}]}

Gambar 5. 11 JSON untuk Grafik Speedometer

5.2.6 Implementasi Hasil Monitoring pada Aplikasi Web

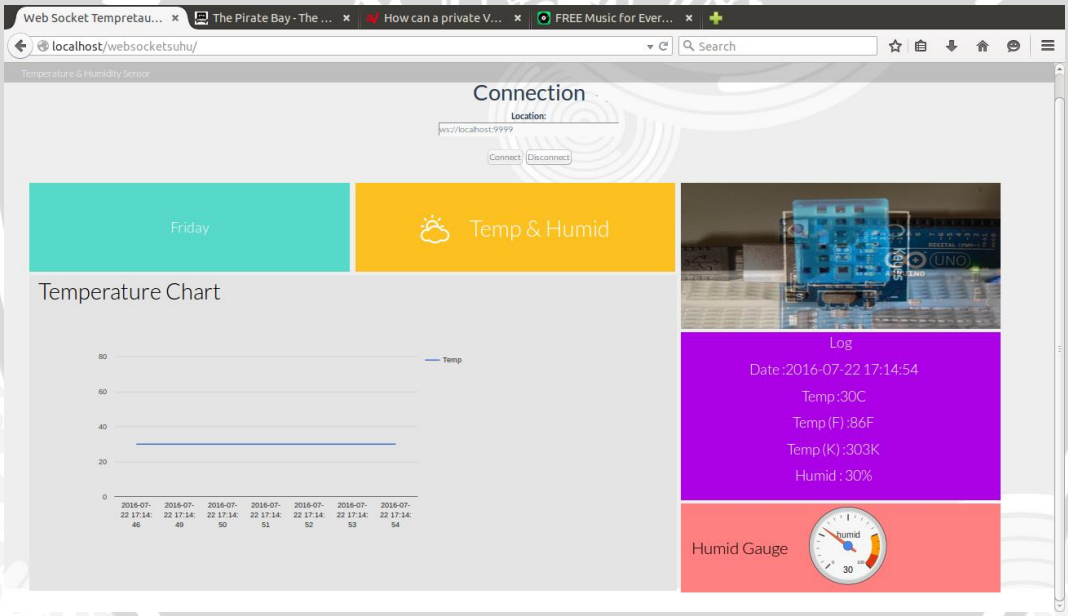
Sesudah pesan telah berhasil dibentuk, maka langkah terakhir yang dilakukan adalah proses pengiriman pesan tersebut ke aplikasi web client. Jalur komunikasi yang dibuat antara client dengan server mengimplementasikan protokol Websocket dengan server menggunakan library *twisted* txws dan client menggunakan Websocket API. Aplikasi web akan memperbaharui pesan yang didapat dari server setiap detik, sehingga sistem dapat bersifat real time.

```
1.  def main():
2.          factory = protocol.ServerFactory()
3.          factory.protocol = Echo
4.          reactor.listenTCP(9999,WebSocketFactory(factory))
5.          reactor.run()
```

Kode Program 5. 17 Pengimplementasian Websocket pada Server

Kode Program 5.17 menjelaskan pembuatan jalur komunikasi antara server dengan client. Dari kode di atas dapat dijelaskan :

1. Pada baris 2 dan 3 adalah *script* yang digunakan untuk membuat *instance* dari kelas *protocol* ketika terdapat komunikasi baru.
2. Baris 4 ditunjukkan bahwa jalur komunikasi berada pada *port* 9999. Sehingga client harus menuliskan alamat server beserta *port* yang telah dituliskan dengan benar. Dalam studi kasusu ini alamat yang dipakai server adalah ws://localhost:9999. Setelah client berhasil melakukan koneksi, server akan langsung mengirimkan data yang sudah diolah.



Gambar 5. 12 Hasil Pengriman Data dari Server ke Client

Gambar 5.12 adalah hasil dari pengiriman data dari server kepada client. Pesan bertipe JSON yang berisikan data suhu dan waktu akan direpresentasikan pada web client dalam bentuk grafik garis. Terlihat data waktu akan mewakli sumbu x sedangkan data suhu akan mewakili sumbu y. Pesan JSON kedua yang berisikan data kelembapan akan dirubah menjadi grafik speedometer. Sedangkan pesan berbentuk *string* akan langsung ditampilkan oleh aplikasi web pada *field* log pesan. Semua pesan yang ditampilkan oleh aplikasi web baik grafik maupun pesan dalam log pesan akan diperbarui setiap detiknya.

BAB 6 HASIL PENGUJIAN DAN ANALISIS

6.1 Hasil Pengujian

Pengujian proses transmisi data, kehandalan sistem melayani request dan fungsionalitas aplikasi web client dilakukan pada penelitian ini. Proses transmisi data menguji seberapa cepta data dapat mengalir dari client ke Arduino sampai data tersebut kembali ke client per detiknya. Pengujian kehandalan sistem melayani request menguji seberapa handal sistem mampu melayani request dengan paramater yang berbeda-beda. Dan pengujian fungsionalitas menguji semua fitur yang tersedia dalam aplikasi web apakah sudah berjalan dengan benar atau belum.

6.1.1 Hasil Pengujian Proses Transmisi Data

Pengujian proses transmisi data dilakukan pada pengimplementasian monitoring suhu dan kelembapan. Dalam pengujian ini aktor client dituliskan dalam bahasa python. Client akan mengirimkan request dalam bentuk angka, angka tersebut akan dikirim dari client sampai ke Arduino, lalu pada Arduino angka tersebut nilainya akan ditambahkan 1 dan akan dikirim lagi ke client melalui server beserta nilai suhu dan kelembapan. Pengujian proses transmisi data dilakukan total sebanyak 4 kali dengan kondisi dan parameter yang berbeda-beda, pengujian tersebut adalah pengujian proses transmisi data tanpa menggunakan Redis per detik dengan jumlah request sebagai parameternya, pengujian proses transmisi data menggunakan Redis per detik dengan jumlah request sebagai parameternya. pengujian proses transmisi data tanpa menggunakan Redis per detik dengan jumlah client sebagai parameternya dan pengujian proses transmisi data menggunakan Redis per detik dengan jumlah client sebagai parameternya.

1. Pengujian Proses Transmisi Data Sistem yang Tidak Menggunakan Redis dengan Jumlah Request yang Berbeda.

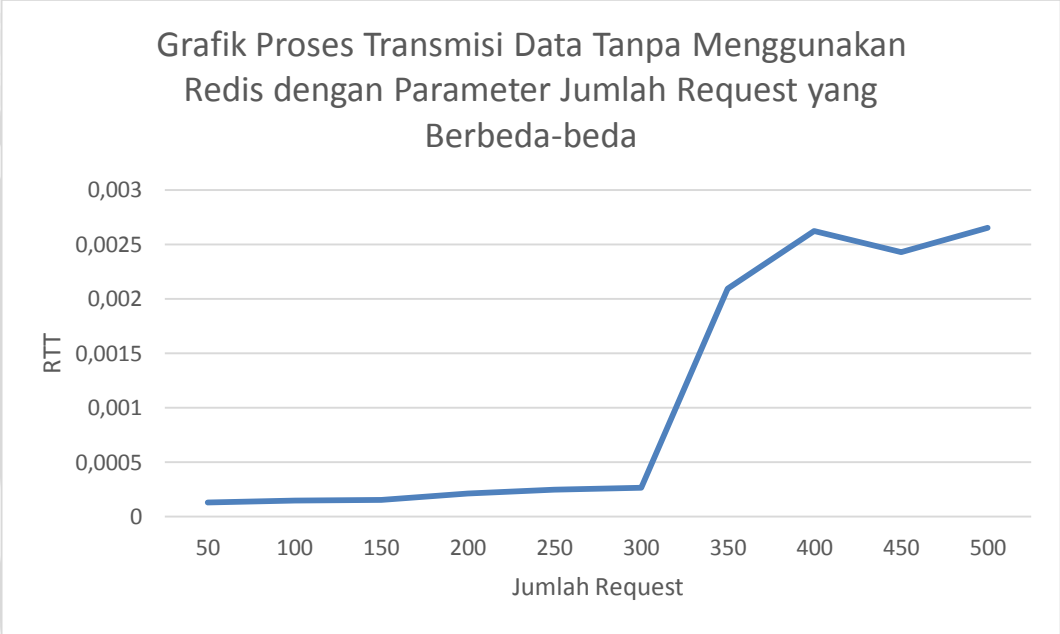
Pengujian yang pertama adalah pengujian proses transmisi data tanpa menggunakan Redis per detik dengan jumlah request sebagai parameternya. Hasil dari pengujian proses transmisi data yang pertama dapat dilihat pada **Tabel 6.1.**

Tabel 6. 1 Hasil Pengujian RTT Sistem yang Tidak Menggunakan Redis dengan Jumlah Request yang Berbeda

No.	Jumlah Request	Round Trip Time (second)
1.	50	0,000128703 detik
2.	100	0,000149128 detik
3.	150	0,000152434 detik
4.	200	0,000211575 detik
5.	250	0,000248356 detik

6.	300	0,000265567 detik
7.	350	0,00209046 detik
8.	400	0,00262454 detik
9.	450	0,002430684 detik
10.	500	0,002651984 detik

Dari data yang telah didapat dari pengujian, hasil dari pengujian proses transmisi data sistem yang tidak menggunakan Redis dengan jumlah request yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.1.**



Gambar 6. 1 Grafik Pengujian RTT Sistem yang Tidak Menggunakan Redis dengan Jumlah Request yang Berbeda

Gambar 6.1 menunjukkan grafik proses transmisi data tanpa menggunakan Redis dengan parameter jumlah request yang berbeda. Dari grafik yang didapatkan dapat diketahui bahwa dalam proses transmisi data tanpa menggunakan Redis memiliki Round Trip Time di bawah 1 detik. Terjadi penurunan performa pada saat client mengirimkan 350 request secara bersamaan. Hasil RTT yang didapatkan setelah mengirimkan 350 request semakin tinggi, walaupun pada saat pengiriman 450 request terdapat peningkatan performa tetapi RTT yang didapatkan tidak jauh berbeda dengan hasil RTT sebelumnya.

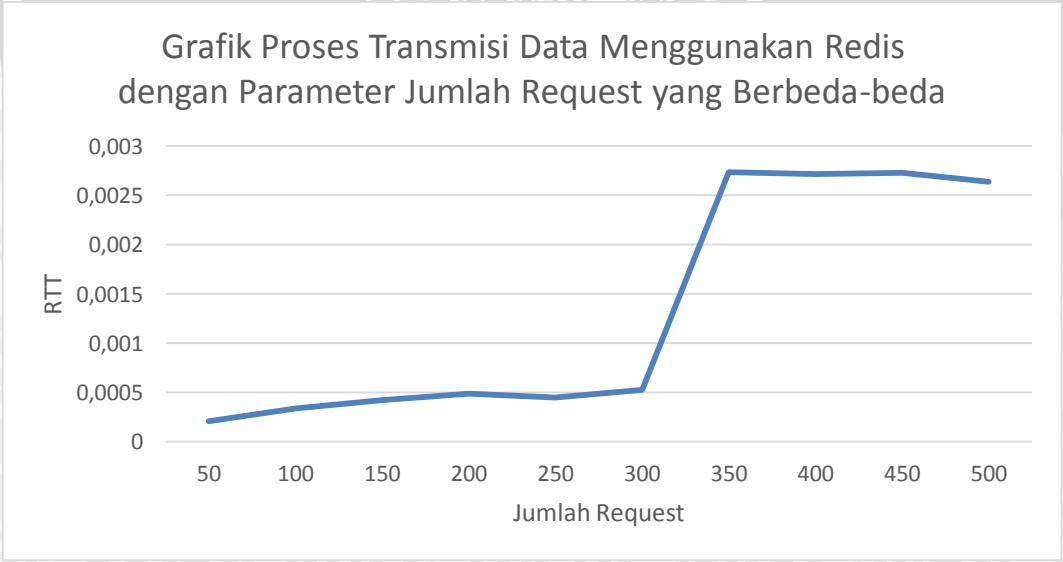
2. Pengujian Proses Transmisi Data Sistem yang Menggunakan Redis dengan Jumlah Request yang Berbeda

Pengujian yang kedua adalah pengujian proses transmisi data menggunakan Redis per detik dengan jumlah request sebagai parameternya. Hasil dari pengujian proses transmisi data yang kedua dapat dilihat pada **Tabel 6.2.**

Tabel 6. 2 Hasil Pengujian RTT Sistem yang Menggunakan Redis dengan Jumlah Request yang Berbeda

No.	Jumlah Request	Round Trip Time (second)
1.	50	0,000208836 detik
2.	100	0,00033766 detik
3.	150	0,000423888 detik
4.	200	0,00048631 detik
5.	250	0,000452888 detik
6.	300	0,000529603 detik
7.	350	0,002733974 detik
8.	400	0,002714516 detik
9.	450	0,002731662 detik
10.	500	0,002639258 detik

Dari data yang telah didapat dari pengujian, hasil dari pengujian proses transmisi data sistem yang menggunakan Redis dengan jumlah request yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.2.**



Gambar 6. 2 Grafik Pengujian RTT Sistem yang Menggunakan Redis dengan Jumlah Request yang Berbeda

Gambar 6.2 menunjukkan grafik proses transmisi data dengan menggunakan Redis dengan parameter jumlah request yang berbeda. Dari grafik yang didapatkan dapat diketahui bahwa dalam proses transmisi data tanpa menggunakan Redis memiliki Round Trip Time di bawah 1 detik. Terjadi penurunan performa yang signifikan pada saat client mengirimkan 350 request. Setelah itu pada saat pengiriman 400, 450 dan 500 request, RTT yang didapatkan tidak jauh berbeda dengan RRT yang didapatkan ketika terdapat 350 request yang dikirimkan.

3. Pengujian Proses Transmisi Data Sistem yang Tidak Menggunakan Redis dengan Jumlah Client yang Berbeda

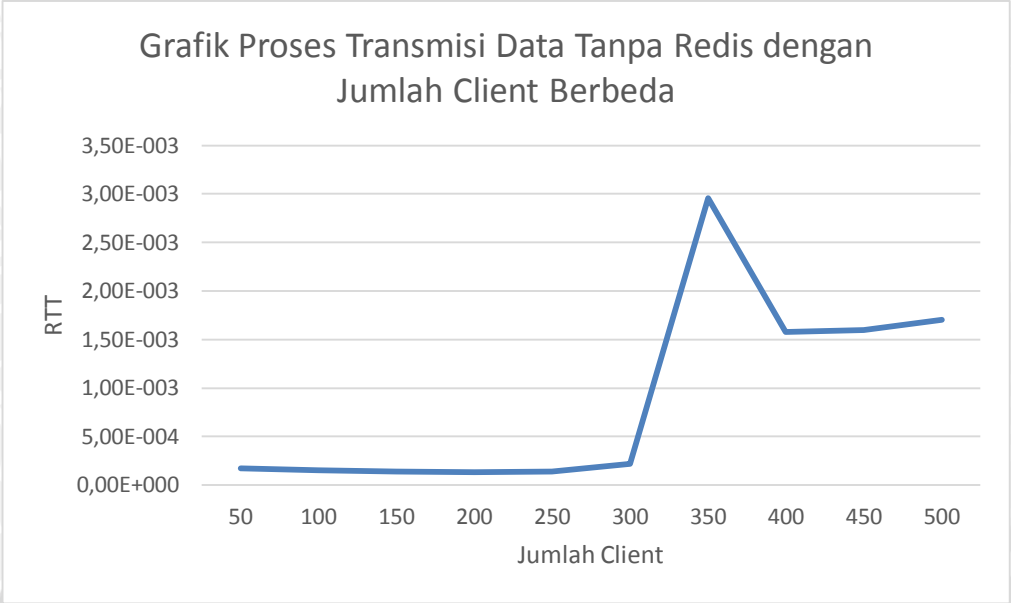
Dalam pengujian proses transmisi data dengan client sebagai parameternya, sebuah benchmark digunakan dalam pengujian ini. Benchmark yang dipakai dalam pengujian ini benchmark thor.

Pengujian yang ketiga adalah pengujian proses transmisi data tanpa menggunakan Redis per detik dengan jumlah client sebagai parameternya. Hasil dari pengujian proses transmisi data yang ketiga dapat dilihat pada **Tabel 6.3.**

Tabel 6. 3 Hasil Pengujian RTT Sistem yang Tidak Menggunakan Redis dengan Jumlah Client yang Berbeda

No.	Jumlah client	Round Trip Time (second)
1.	50	0,0001740913391112 detik
2.	100	0,0001542830467226 detik
3.	150	0,0001384579340616 detik
4.	200	0,0001299200057984 detik
5.	250	0,0001410608291626 detik
6.	300	0,0002212476730346 detik
7.	350	0,002956653129712 detik
8.	400	0,001574401759402 detik
9.	450	0,00159514892947 detik
10.	500	0,001700545070534 detik

Dari data yang telah didapat dari pengujian, hasil dari pengujian proses transmisi data sistem yang tidak menggunakan Redis dengan jumlah client yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.3.**



Gambar 6. 3 Grafik Pengujian RTT Sistem yang Tidak Menggunakan Redis dengan Jumlah Client yang Berbeda

Gambar 6.3 menunjukkan grafik proses transmisi data tanpa menggunakan Redis dengan parameter jumlah client yang berbeda. Dari grafik yang didapatkan dapat diketahui bahwa dalam proses transmisi data tanpa menggunakan Redis memiliki Round Trip Time di bawah 1 detik. Terjadi penurunan performa pada saat terdapat 350 client yang terhubung dengan server dan mengirimkan masing-masing 1 request. Tetapi RTT yang didapatkan kembali mengecil ketika terdapat 400, 450, dan 500 client.

4. Pengujian Proses Transmisi Data Sistem yang Menggunakan Redis dengan Jumlah Client yang Berbeda

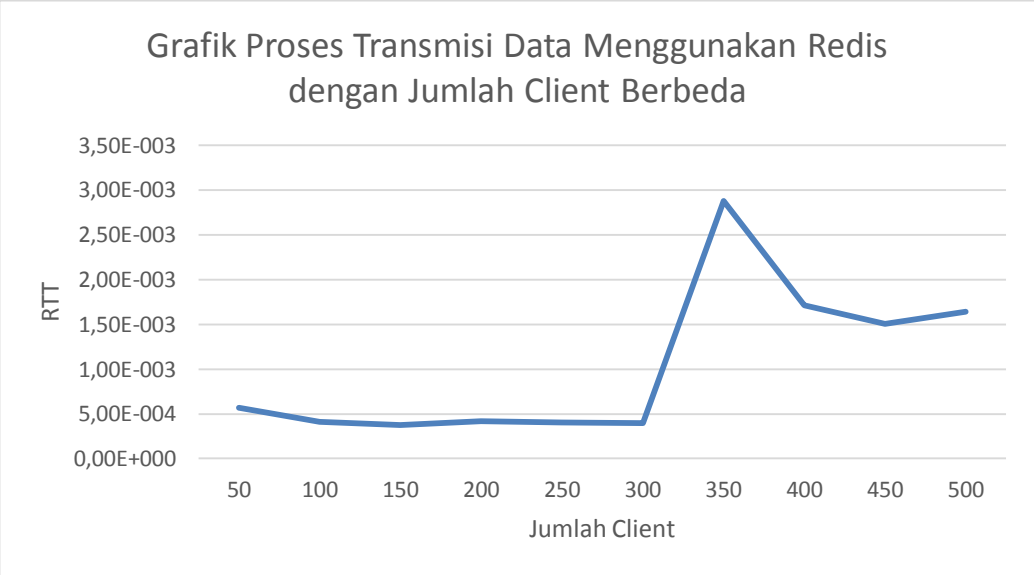
Pengujian yang keempat adalah pengujian proses transmisi data menggunakan Redis per detik dengan jumlah client sebagai parameternya. Hasil dari pengujian proses transmisi data yang keempat dapat dilihat pada **Tabel 6.4.**

Tabel 6. 4 Hasil Pengujian RTT Sistem yang Menggunakan Redis dengan Jumlah Client yang Berbeda

No.	Jumlah client	Round Trip Time (second)
1.	50	0,0005658788681032 detik
2.	100	0,0004101386070252 detik
3.	150	0,000377179145813 detik
4.	200	0,0004158730506898 detik
5.	250	0,0004054092407226 detik
6.	300	0,0003946862220764 detik
7.	350	0,002880354422478 detik

8.	400	0,00171579638348 detik
9.	450	0,001506178915502 detik
10.	500	0,001639120856378 detik

Dari data yang telah didapat dari pengujian, hasil dari pengujian proses transmisi data sistem yang menggunakan Redis dengan jumlah client yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.4.**



Gambar 6. 4 Grafik Pengujian RTT Sistem yang Menggunakan Redis dengan Jumlah Client yang Berbeda

Gambar 6.4 menunjukkan grafik proses transmisi data menggunakan Redis dengan parameter jumlah client yang berbeda. Dari grafik yang didapatkan dapat diketahui bahwa dalam proses transmisi data menggunakan Redis memiliki Round Trip Time di bawah 1 detik. Terjadi penurunan performa pada saat terdapat 350 client yang terhubung dengan server mengirimkan masing-masing 1 request. Tetapi RTT yang didapatkan kembali mengecil ketika terdapat 400, 450, dan 500 client.

6.1.2 Hasil Pengujian Kehandalan Sistem Melayani Request

Pengujian kehandalan sistem melayani request dilakukan pada pengimplementasian monitoring suhu dan kelembapan. Sama dengan pengjian proses transmisi data, client akan dituliskan dalam bahasa python. Untuk pengujian jumlah client sebagai parameternya, pengujian dilakukan menggunakan benchmark thor. Proses pengujian akan dilakukan selama 1 detik, setelah 1 detik berlalu sistem akan langsung berhenti. Pengujian kehandalan sistem melayani requests dilakukan total sebanyak 4 kali dengan kondisi dan parameter yang berbeda-beda, pengujian tersebut adalah pengujian kehandalan sistem melayani request tanpa menggunakan Redis per detik dengan jumlah request sebagai

parameternya, pengujian proses kehandalan sistem melayani request menggunakan Redis per detik dengan jumlah request sebagai parameternya. pengujian proses kehandalan sistem melayani request menggunakan Redis per detik dengan jumlah client sebagai parameternya dan pengujian kehandalan sistem melayani request menggunakan Redis per detik dengan jumlah client sebagai parameternya.

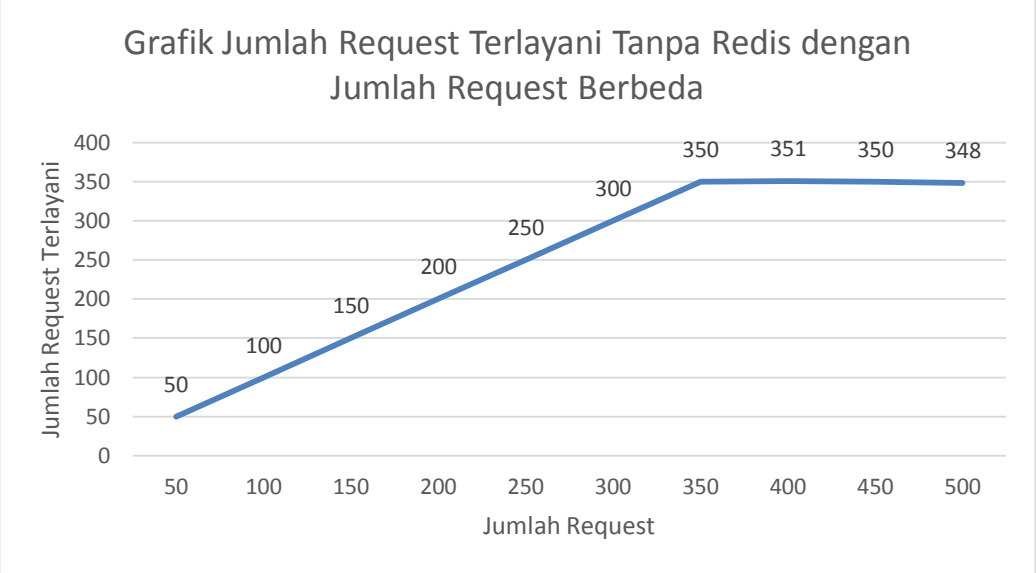
1. Pengujian Kehandalan Sistem yang Tidak Menggunakan Redis Dalam Melayani Request dengan Jumlah Request yang Berbeda

Pengujian yang pertama adalah pengujian kehandalan sistem melayani request tanpa menggunakan Redis per detik dengan jumlah request sebagai parameternya. Hasil dari pengujian proses transmisi data yang pertama dapat dilihat pada **Tabel 6.5.**

Tabel 6. 5 Hasil Pengujian Kehandalan Sistem yang Tidak Menggunakan Redis Dalam Melayani Request dengan Jumlah Request yang Berbeda

No.	Jumlah Request	Jumlah Request yang Terlayani
1.	50	50
2.	100	100
3.	150	150
4.	200	200
5.	250	250
6.	300	300
7.	350	350
8.	400	351
9.	450	350
10.	500	348

Dari data yang telah didapat dari pengujian, hasil dari pengujian kehandalan sistem yang tidak menggunakan redis dalam melayani request dengan jumlah request yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.5**.



Gambar 6. 5 Grafik Pengujian Kehandalan Sistem yang Tidak Menggunakan Redis Dalam Melayani Request dengan Jumlah Request yang Berbeda

Gambar 6.5 menunjukkan grafik dari analisis kedua yang dilakukan untuk menghitung jumlah request yang mampu dilayani oleh sistem per detiknya. Dengan parameter jumlah request yang berbeda-beda, sistem yang tidak menggunakan redis mampu melayani request dengan maksimal 351 request per detiknya. Pada saat terdapat 400, 450, atau 500 request yang dikirimkan secara bersamaan, sistem hanya hanya mampu melayani sebanyak maksimal 351 request dan sistem tidak dapat melayani semua request dalam satu detik.

2. Pengujian Kehandalan Sistem yang Menggunakan Redis Dalam Melayani Request dengan Jumlah Request yang Berbeda

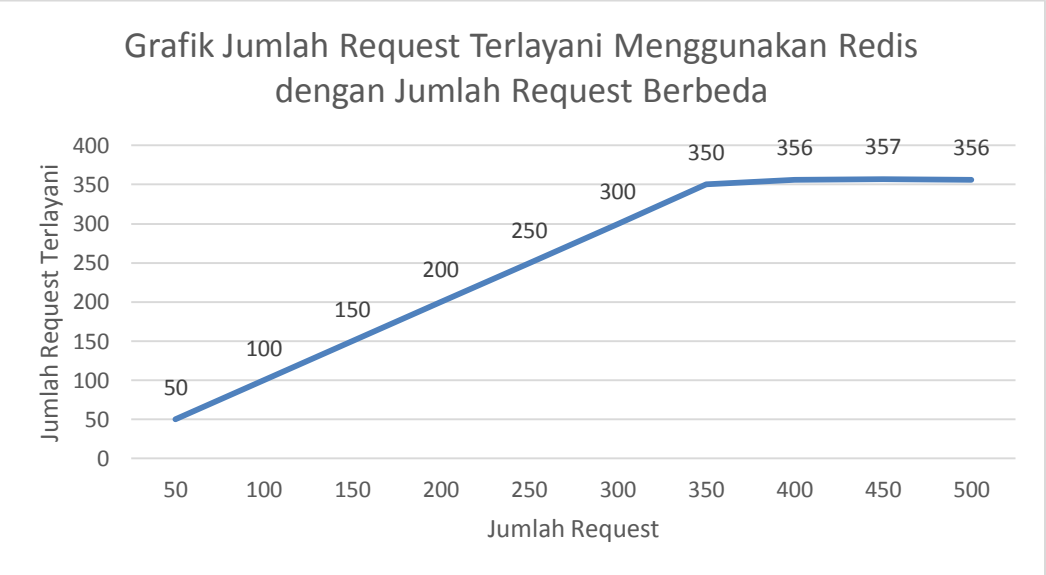
Pengujian yang kedua adalah pengujian kehandalan sistem melayani request menggunakan Redis per detik dengan jumlah request sebagai parameternya. Hasil dari pengujian proses transmisi data yang kedua dapat dilihat pada **Tabel 6.6.**

Tabel 6. 6 Hasil Pengujian Kehandalan Sistem yang Menggunakan Redis Dalam Melayani Request dengan Jumlah Request yang Berbeda

No.	Jumlah Request	Jumlah Request yang Terlayani
1.	50	50
2.	100	100
3.	150	150
4.	200	200
5.	250	250
6.	300	300
7.	350	350

8.	400	356
9.	450	357
10.	500	356

Dari data yang telah didapat dari pengujian, hasil dari pengujian kehandalan sistem yang menggunakan redis dalam melayani request dengan jumlah request yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.6.**



Gambar 6. 6 Grafik Pengujian Kehandalan Sistem yang Menggunakan Redis Dalam Melayani Request dengan Jumlah Request yang Berbeda

Gambar 6.6 menunjukkan analisis grafik request yang mampu dilayani dengan menggunakan redis dengan jumlah request yang berbeda. Dengan sistem yang menggunakan redis terlihat sistem mampu melayani request dengan maksimal 357 request per detiknya. Pada saat terdapat 400, 450, atau 500 client yang terhubung secara bersamaan, sistem hanya hanya mampu melayani sebanyak maksimal 357 request dan sistem tidak dapat melayani semua request dalam satu detik.

3. Pengujian Kehandalan Sistem yang Tidak Menggunakan Redis Dalam Melayani Request dengan Jumlah Client yang Berbeda

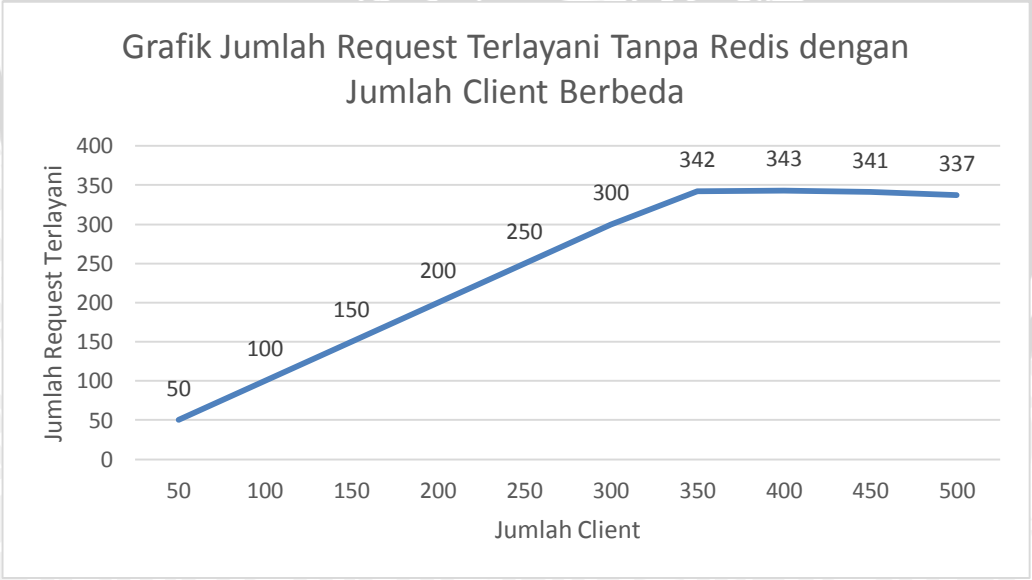
Dalam pengujian kehandalan sistem melayani request dengan client sebagai parameternya, sebuah benchmark digunakan dalam pengujian ini. Benchmark yang dipakai dalam pengujian ini benchmark thor.

Pengujian yang ketiga adalah kehandalan sistem melayani request tanpa menggunakan Redis per detik dengan jumlah client sebagai parameternya. Hasil dari pengujian proses transmisi data yang ketiga dapat dilihat pada **Tabel 6.7.**

Tabel 6. 7 Hasil Pengujian Kehandalan Sistem yang Tidak Menggunakan Redis Dalam Melayani Request dengan Jumlah Client yang Berbeda

No.	Jumlah client	Jumlah Request yang Terlayani
1.	50	50
2.	100	100
3.	150	150
4.	200	200
5.	250	250
6.	300	300
7.	350	342
8.	400	343
9.	450	341
10.	500	337

Dari data yang telah didapat dari pengujian, hasil dari pengujian kehandalan sistem yang tidak menggunakan redis dalam melayani request dengan jumlah client yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.7.**



Gambar 6. 7 Grafik Pengujian Kehandalan Sistem yang Tidak Menggunakan Redis Dalam Melayani Request dengan Jumlah Client yang Berbeda

Gambar 6.7 menunjukkan analisis grafik request yang mampu dilayani tanpa menggunakan redis dengan jumlah client yang berbeda. Dengan sistem yang

tanpa menggunakan redis terlihat sistem mampu melayani request maksimal sebanyak 342 client. Sistem tidak mampu melayani semua request dengan client diatas 300 yang terhubung secara bersamaan.

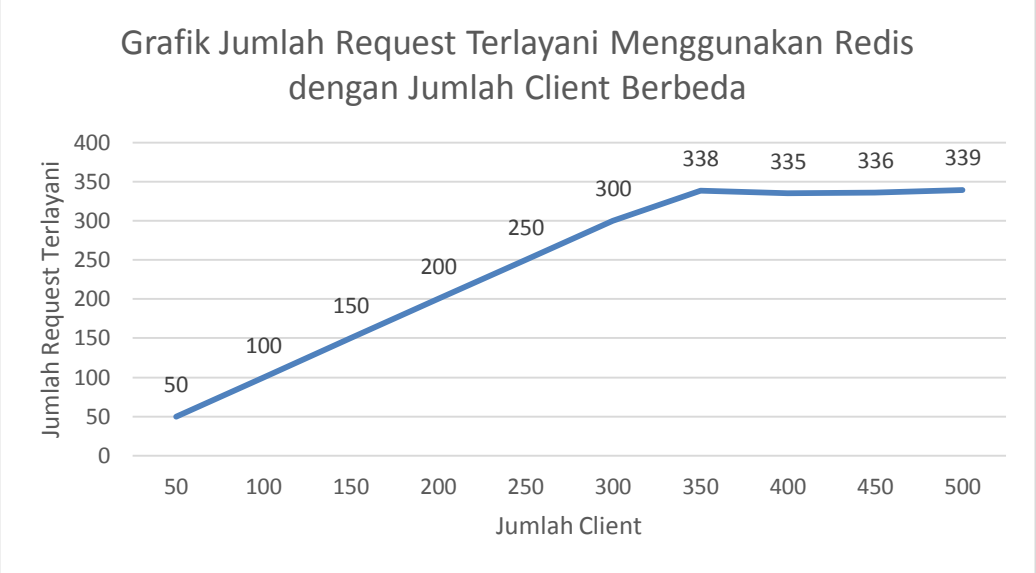
4. Pengujian Kehandalan Sistem yang Menggunakan Redis Dalam Melayani Request dengan Jumlah Client yang Berbeda

Pengujian yang keempat adalah kehandalan sistem melayani request menggunakan Redis per detik dengan jumlah client sebagai parameternya. Hasil dari pengujian proses transmisi data yang keempat dapat dilihat pada **Tabel 6.8.**

Tabel 6. 8 Hasil Pengujian Kehandalan Sistem yang Menggunakan Redis Dalam Melayani Request dengan Jumlah Client yang Berbeda

No.	Jumlah client	Jumlah Request yang Terlayani
1.	50	50
2.	100	100
3.	150	150
4.	200	200
5.	250	250
6.	300	300
7.	350	338
8.	400	336
9.	450	336
10.	500	339

Dari data yang telah didapat dari pengujian, hasil dari pengujian kehandalan sistem yang menggunakan redis dalam melayani request dengan jumlah client yang berbeda dapat direpresentasikan dalam bentuk grafik garis yang ditunjukkan pada **Gambar 6.8.**



Gambar 6. 8 Grafik Pengujian Kehandalan Sistem yang Menggunakan Redis Dalam Melayani Request dengan Jumlah Client yang Berbeda

Gambar 6.8 menunjukkan analisis grafik request yang mampu dilayani dengan menggunakan redis dengan jumlah client yang berbeda. Dengan sistem yang menggunakan redis terlihat sistem mampu melayani request maksimal sebanyak 338 client. Sistem tidak mampu melayani semua request dengan client diatas 300 yang terhubung secara bersamaan.

6.1.3 Hasil Pengujian Fungsionalitas Aplikasi Web Client

Pengujian fungsionalitas aplikasi web client dilakukan untuk menguji apakah sistem dapat melakukan semua fungsi dengan benar atau belum. Terdapat 2 pengujian fungsionalitas yaitu pengujian fungsionalitas aplikasi web kontrloing lampu *LED* dan pengujian fungsionalitas aplikasi web monitoring suhu dan kelembapan. Hasil kedua pengujian fungsionalitas dapat dilihat dalam **Tabel 6.**9 dan **Tabel 6.10.**

Tabel 6. 9 Hasil Pengujian Fungsionalitas Aplikasi Web Kontroling LED

No.	Deskripsi	Masukan	Keluaran yang Diharapkan	Hasil yang Didapat
1.	Pengujian pembuatan koneksi antara client dengan server	Alamat websocket	Sistem menampilkan pesan “connected” dan komunikasi antara client dengan server terbentuk	Sistem menampilkan pesan “connected” dan komunikasi antara client dengan server terbentuk

2.	Pengujian pemutusan koneksi antara client dengan server	Tekan tombol disconnect	Sistem menampilkan pesan “disconnected” dan komunikasi antara client dengan server terputus.	Sistem menampilkan pesan “disconnected” dan komunikasi antara client dengan server terputus.
3.	Pengujian pengiriman pesan hidup semua	Tekan tombol hidup semua	Sistem mengirim pesan hidup semua kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup semua kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.
4.	Pengujian pengiriman pesan mati semua	Tekan tombol mati semua	Sistem mengirim pesan mati semua kepada server dan menampilkan “sent : mati semua” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati semua kepada server dan menampilkan “sent : mati semua” dan waktu eksekusi yang dikirim oleh server.
5.	Pengujian pengiriman pesan hidup 2	Tekan tombol hidup 2	Sistem mengirim pesan hidup 2 kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 2 kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.
6.	Pengujian pengiriman pesan mati 2	Tekan tombol mati 2	Sistem mengirim pesan mati 2 kepada server dan menampilkan “sent : mati 2” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati 2 kepada server dan menampilkan “sent : mati 2” dan waktu eksekusi yang dikirim oleh server.
7.	Pengujian pengiriman pesan hidup 3	Tekan tombol hidup 3	Sistem mengirim pesan hidup 3 kepada server dan menampilkan “sent : hidup 3” dan waktu eksekusi	Sistem mengirim pesan hidup 3 kepada server dan menampilkan “sent : hidup 3” dan waktu eksekusi yang dikirim oleh server.

			yang dikirim oleh server.	
8.	Pengujian pengiriman pesan mati 3	Tekan tombol mati 3	Sistem mengirim pesan mati 3 kepada server dan menampilkan “sent : mati 3” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati 3 kepada server dan menampilkan “sent : mati 3” dan waktu eksekusi yang dikirim oleh server.
9.	Pengujian pengiriman pesan hidup 4	Tekan tombol hidup 4	Sistem mengirim pesan hidup 4 kepada server dan menampilkan “sent : hidup 4” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 4 kepada server dan menampilkan “sent : hidup 4” dan waktu eksekusi yang dikirim oleh server.
10.	Pengujian pengiriman pesan mati 4	Tekan tombol mati 4	Sistem mengirim pesan mati 4 kepada server dan menampilkan “sent : mati 4” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati 4 kepada server dan menampilkan “sent : mati 4” dan waktu eksekusi yang dikirim oleh server.
11.	Pengujian pengiriman pesan hidup 5	Tekan tombol hidup 5	Sistem mengirim pesan hidup 5 kepada server dan menampilkan “sent : hidup 5” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 5 kepada server dan menampilkan “sent : hidup 5” dan waktu eksekusi yang dikirim oleh server.
12.	Pengujian pengiriman pesan mati 5	Tekan tombol mati 5	Sistem mengirim pesan mati 5 kepada server dan menampilkan “sent : mati 5” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati 5 kepada server dan menampilkan “sent : mati 5” dan waktu eksekusi yang dikirim oleh server.

Hasil pengujian fungsionalitas aplikasi web monitoring suhu dan kelembapan ditunjukkan pada **Tabel 6.10.**

Tabel 6. 10 Hasil Pengujian Fungsionalitas Aplikasi Web Monitoring Suhu dan Kelembapan

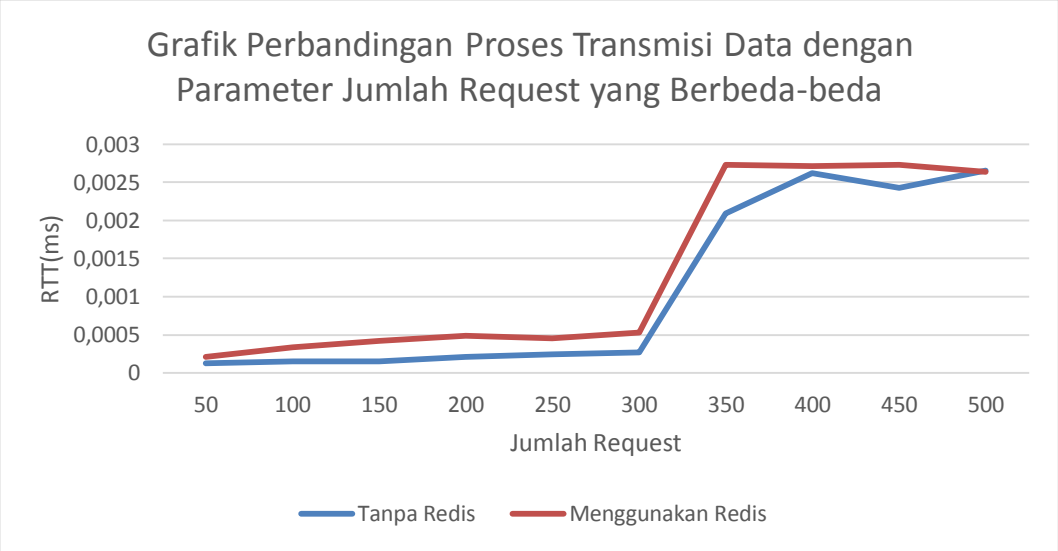
No.	Deskripsi	Masukan	Keluaran yang Diharapkan	Hasil yang Didapat
1.	Pengujian pembuatan koneksi antara client dengan server	Alamat websocket	Sistem menampilkan pesan "connected" dan komunikasi antara client dengan server terbentuk	Sistem menampilkan pesan "connected" dan komunikasi antara client dengan server terbentuk
2.	Pengujian pemutusan koneksi antara client dengan server	Tekan tombol disconnect	Sistem menampilkan pesan "disconnected" dan komunikasi antara client dengan server terputus.	Sistem menampilkan pesan "disconnected" dan komunikasi antara client dengan server terputus.
3.	Pengujian menampilkan pesan dari server	Data suhu C, suhu K, suhu F, kelembapan dan waktu	Sistem menampilkan pesan dari server yaitu pesan waktu, suhu dalam celcius, suhu dalam fahrenheit, suhu dalam kelvin, dan kelembapan	Sistem menampilkan pesan dari server yaitu pesan waktu, suhu dalam celcius, suhu dalam fahrenheit, suhu dalam kelvin, dan kelembapan
4.	Pengujian menampilkan grafik garis	Data JSON	Sistem menampilkan grafik garis sesuai dengan isi dari JSON yang dikirimkan	Sistem menampilkan grafik garis sesuai dengan isi dari JSON yang dikirimkan
5.	Pengujian menampilkan grafik speedometer	Data JSON	Sistem menampilkan grafik speedometer sesuai dengan isi dari JSON yang dikirimkan	Sistem menampilkan grafik speedometer sesuai dengan isi dari JSON yang dikirimkan

6.2Analisis

Setelah melakukan pengujian, dilakukan analisis terhadap hasil dari pengujian. Secara keseluruhan, sistem telah mampu berjalan sesuai dengan perancangan yang telah dibuat. Pengujian perlu dilakukan untuk mengetahui kinerja sistem dengan parameter uji berupa rata-rata Round Trip Time, keberhasilan sistem dalam melayani request, dan keberhasilan sistem dalam menjalankan semua fungsi. Pengujian pengamatan data dilakukan sebanyak 8 kali. Lalu nantinya data tersebut akan dibandingkan antara pengujian dengan memakai Redis dengan pengujian dengan tidak menggunakan Redis.

6.2.1 Analisis Pengujian Proses Transmisi Data

1. Perbandingan Pengujian Proses Transmisi Data Antara Sistem yang Menggunakan dengan Sistem yang Tidak Mengunakan Redis dengan Jumlah Request yang Berbeda-beda



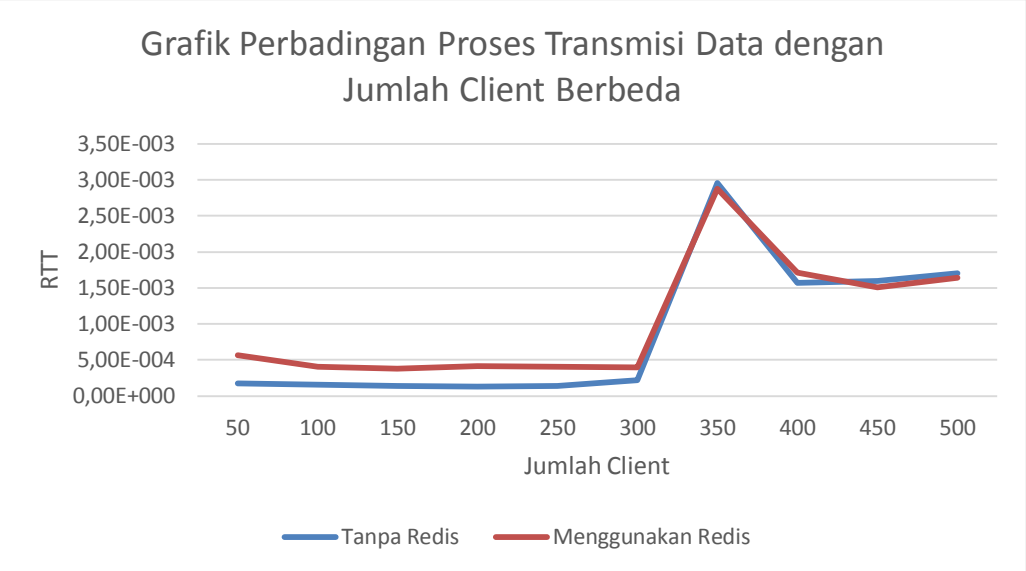
Gambar 6. 9 Grafik Perbandingan Pengujian RTT dengan Jumlah Request Berbeda

Gambar 6.9 menjelaskan tentang perbandingan antara proses transmisi data tanpa menggunakan Redis dengan proses transmisi data menggunakan Redis. Dari Grafik dapat ditarik kesimpulan waktu proses transmisi data dengan parameter jumlah request yang berbeda tanpa menggunakan Redis lebih unggul daripada waktu proses transmisi data menggunakan Redis. Perbedaan RTT yang dihasilkan dari 2 pengujian tidak jauh berbeda satu sama lain. Kedua sistem mampu memberikan nilai RTT kurang dari 1 detik. Pada pengujian tersebut keduanya sama-sama mengalami peningkatan waktu RTT pada saat client mengirimkan 350 request.

Terjadi peningkatan waktu RTT pada saat dikirimkan 350 request hingga seterusnya, hal ini dikarenakan dalam TCP terdapat metode *flow control*. *Flow control* digunakan untuk mengosongkan *buffer* supaya tidak terjadi *flooding* data.

Sehingga ketika *buffer* sudah penuh maka akan dilakukan pengosongan *buffer* sampai *buffer* tersebut kosong kembali.

2. Perbandingan Pengujian Proses Transmisi Data Antara Sistem yang Menggunakan dengan Sistem yang Tidak Mengunakan Redis dengan Jumlah Client yang Berbeda-beda



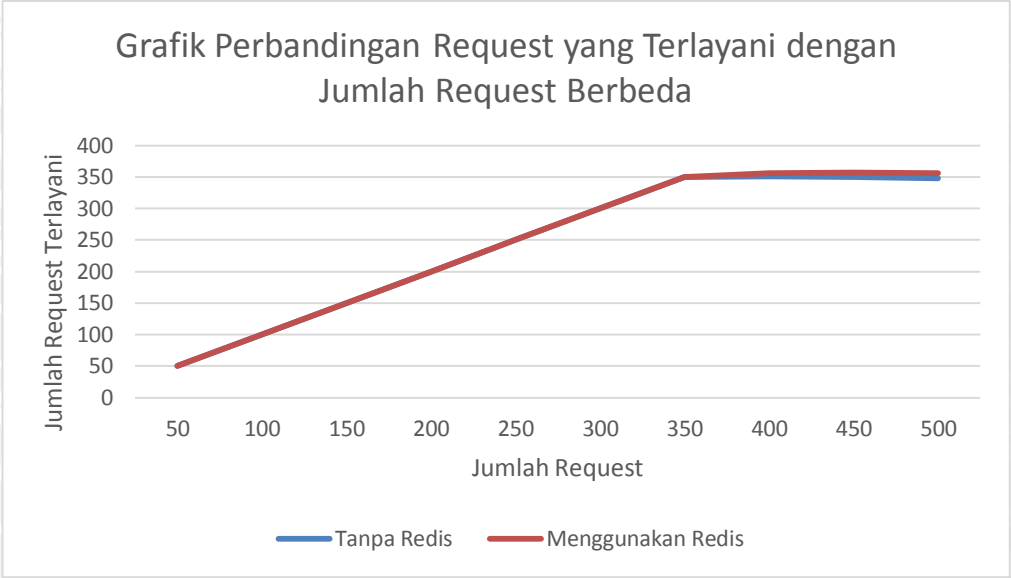
Gambar 6. 10 Grafik Perbandingan Pengujian RTT dengan Jumlah Client Berbeda

Gambar 6.10 menjelaskan tentang perbandingan antara proses transmisi data tanpa menggunakan Redis dengan proses transmisi data menggunakan Redis. Dari Grafik dapat ditarik kesimpulan waktu proses transmisi data tanpa menggunakan Redis lebih unggul daripada waktu proses transmisi data menggunakan Redis. Perbedaan RTT yang dihasilkan dari 2 pengujian tidak jauh berbeda satu sama lain. Kedua pengujian sama-sama menghasilakn nilai RTT kurang dari 1 detik. Pada pengujian tersebut keduanya sama-sama mengalami peningkatan waktu RTT pada saat client mengirimkan 350 request. Nilai RTT kembali mengecil ketika dikirimkan 400, 450, dan 500 request yang dikirim bersamaan.

Terjadi peningkatan waktu RTT pada saat terdapat 350 client yang terkoneksi secara bersamaan, hal ini dikarenakan ketika server mengirimkan data ke salah satu client, client tersebut telah terputus koneksinya dengan server. Untuk menyelesaikan satu request TCP memerlukan ACK dari client untuk menandakan data sudah sampai tujuan. Sehingga data yang tidak sampai ke client akan menunggu sampai waktu *timed out* berlalu dan data tersebut akan di drop.

6.2.2 Analisis Pengujian Kehandalan Sistem Melayani Request

1. Perbandingan Pengujian Kehandalan Sistem Sistem yang Menggunakan dengan Sistem yang Tidak Mengunakan Redis dengan Jumlah Request yang Berbeda-beda

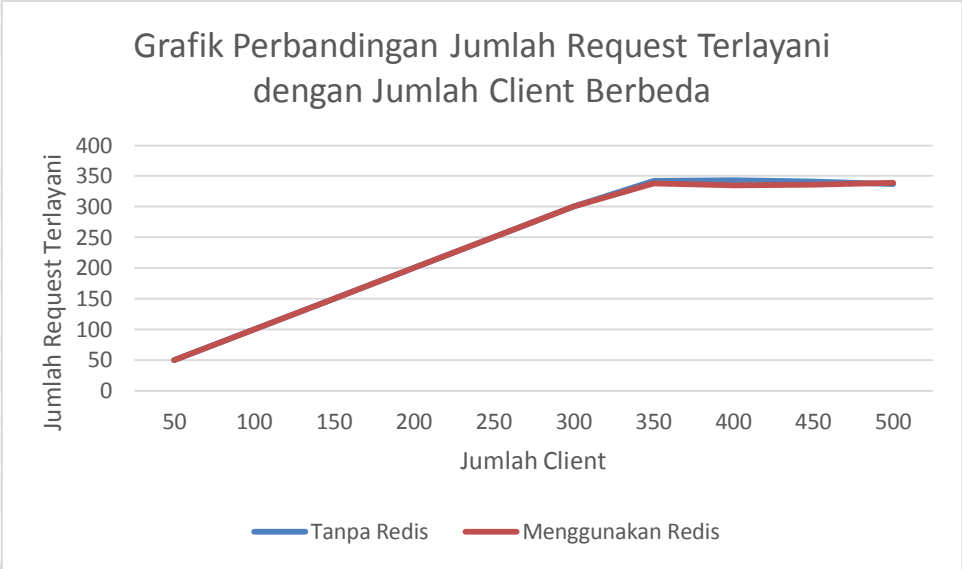


Gambar 6. 11 Grafik Perbandingan Pengujian Kehandalan Sistem Melayani Request dengan Jumlah Request yang Berbeda

Gambar 6.11 menunjukkan hasil perbandingan analisis jumlah request yang terlayani antara sistem yang menggunakan redis dengan sistem yang tidak menggunakan redis dengan jumlah request yang berbeda. Dari grafik tersebut dapat dijelaskan bahwa kedua sistem memiliki kehandalan melayani request yang hampir sama. Kedua sistem sama-sama dapat melayani request dengan range antara 350 sampai 357 request yang dikirim secara bersamaan selama satu detik.

Kedua sistem memiliki kehandalan melayani request yang hampir sama, hal ini dikarenakan database Redis tersimpan dalam memori. Sehingga untuk mengambil nilai yang terletak pada Redis tidak perlu memerlukan waktu yang lama.

2. Perbandingan Pengujian Kehandalan Sistem Sistem yang Menggunakan dengan Sistem yang Tidak Mengunakan Redis dengan Jumlah Client yang Berbeda-beda



Gambar 6. 12 Grafik Perbandingan Pengujian Kehandalan Sistem Melayani Request dengan Jumlah Client yang Berbeda

Gambar 6.12 menunjukkan hasil perbandingan analisis jumlah request yang terlayani antara sistem yang menggunakan redis dengan sistem yang tidak menggunakan redis dengan jumlah client yang berbeda. Dari grafik tersebut dapat dijelaskan bahwa kedua sistem memiliki kehandalan melayani request yang hampir sama. Kedua sistem sama-sama dapat melayani request dengan range antara 338 sampai 342 request client yang terhubung secara bersamaan.

Kedua sistem memiliki kehandalan melayani request yang hampir sama, hal ini dikarenakan database Redis tersimpan dalam memori. Sehingga untuk mengambil nilai yang terletak pada Redis tidak perlu memerlukan waktu yang lama.

6.2.3 Analisis Fungsionalitas Aplikasi Web Client

Analisis pengujian fungsionalitas dilakukan untuk menilai apakah semua fungsi yang tersedia pada aplikasi web kontroling dan monitoring sudah berjalan dengan benar. Hasil penilaian dari pengujian fungsionalitas dari kedua aplikasi web client dapat dilihat pada **Tabel 6.11** dan **Tabel 6.12**

Tabel 6. 11 Analisis Fungsionalitas Aplikasi Web Kontroling LED

No.	Deskripsi	Masukan	Keluaran yang Diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian pembuatan koneksi antara client dengan server	Alamat websocket	Sistem menampilkan pesan "connected" dan komunikasi antara client dengan server terbentuk	Sistem menampilkan pesan "connected" dan komunikasi antara client dengan server terbentuk	Benar
2.	Pengujian pemutusan koneksi antara client dengan server	Tekan tombol disconnect	Sistem menampilkan pesan "disconnected" dan komunikasi antara client dengan server terputus.	Sistem menampilkan pesan "disconnected" dan komunikasi antara client dengan server terputus.	Benar
3.	Pengujian pengiriman pesan hidup semua	Tekan tombol hidup semua	Sistem mengirim pesan hidup semua kepada server dan menampilkan "sent : hidup semua" dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup semua kepada server dan menampilkan "sent : hidup semua" dan waktu eksekusi yang dikirim oleh server.	Benar
4.	Pengujian pengiriman pesan mati semua	Tekan tombol mati semua	Sistem mengirim pesan mati semua kepada server dan menampilkan "sent : mati semua" dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati semua kepada server dan menampilkan "sent : mati semua" dan waktu eksekusi	Benar

				yang dikirim oleh server.	
5.	Pengujian pengiriman pesan hidup 2	Tekan tombol hidup 2	Sistem mengirim pesan hidup 2 kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 2 kepada server dan menampilkan “sent : hidup semua” dan waktu eksekusi yang dikirim oleh server.	Benar
6.	Pengujian pengiriman pesan mati 2	Tekan tombol mati 2	Sistem mengirim pesan mati 2 kepada server dan menampilkan “sent : mati 2” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati 2 kepada server dan menampilkan “sent : mati 2” dan waktu eksekusi yang dikirim oleh server.	Benar
7.	Pengujian pengiriman pesan hidup 3	Tekan tombol hidup 3	Sistem mengirim pesan hidup 3 kepada server dan menampilkan “sent : hidup 3” dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 3 kepada server dan menampilkan “sent : hidup 3” dan waktu eksekusi yang dikirim oleh server.	Benar
8.	Pengujian pengiriman pesan mati 3	Tekan tombol mati 3	Sistem mengirim pesan mati 3 kepada server dan menampilkan “sent : mati 3” dan waktu eksekusi yang	Sistem mengirim pesan mati 3 kepada server dan menampilkan “sent : mati 3” dan waktu eksekusi yang	Benar

			dikirim oleh server.	dikirim oleh server.	
9.	Pengujian pengiriman pesan hidup 4	Tekan tombol hidup 4	Sistem mengirim pesan hidup 4 kepada server dan menampilkan "sent : hidup 4" dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 4 kepada server dan menampilkan "sent : hidup 4" dan waktu eksekusi yang dikirim oleh server.	Benar
10.	Pengujian pengiriman pesan mati 4	Tekan tombol mati 4	Sistem mengirim pesan mati 4 kepada server dan menampilkan "sent : mati 4" dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan mati 4 kepada server dan menampilkan "sent : mati 4" dan waktu eksekusi yang dikirim oleh server.	Benar
11.	Pengujian pengiriman pesan hidup 5	Tekan tombol hidup 5	Sistem mengirim pesan hidup 5 kepada server dan menampilkan "sent : hidup 5" dan waktu eksekusi yang dikirim oleh server.	Sistem mengirim pesan hidup 5 kepada server dan menampilkan "sent : hidup 5" dan waktu eksekusi yang dikirim oleh server.	Benar
12.	Pengujian pengiriman pesan mati 5	Tekan tombol mati 5	Sistem mengirim pesan mati 5 kepada server dan menampilkan "sent : mati 5" dan waktu eksekusi yang	Sistem mengirim pesan mati 5 kepada server dan menampilkan "sent : mati 5" dan waktu eksekusi yang	Benar

			dikirim oleh server.	dikirim oleh server.	

Hasil penialain pengujian fungsionalitas aplikasi web monitoring suhu dan kelembapan ditunjukkan pada **Tabel 6.12**

Tabel 6. 12 Analisis Fungsionalitas Aplikasi Web Monitoring Suhu dan Kelembapan

No.	Deskripsi	Masukan	Keluaran yang Diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian pembuatan koneksi antara client dengan server	Alamat websocket	Sistem menampilkan pesan “connected” dan komunikasi antara client dengan server terbentuk	Sistem menampilkan pesan “connected” dan komunikasi antara client dengan server terbentuk	Benar
2.	Pengujian pemutusan koneksi antara client dengan server	Tekan tombol disconnect	Sistem menampilkan pesan “disconnected” dan komunikasi antara client dengan server terputus.	Sistem menampilkan pesan “disconnected” dan komunikasi antara client dengan server terputus.	Benar
3.	Pengujian menampilkan pesan dari server	Data suhu C, suhu K, suhu F, kelembapan dan waktu	Sistem menampilkan pesan dari server yaitu pesan waktu, suhu dalam celcius, suhu dalam fahrenheit, suhu dalam kelvin, dan kelembapan	Sistem menampilkan pesan dari server yaitu pesan waktu, suhu dalam celcius, suhu dalam fahrenheit, suhu dalam	Benar

				kelvin, dan kelembapan	
4.	Pengujian menampilkan grafik garis	Data JSON	Sistem menampilkan grafik garis sesuai dengan isi dari JSON yang dikirimkan	Sistem menampilkan grafik garis sesuai dengan isi dari JSON yang dikirimkan	Benar
5.	Pengujian menampilkan grafik speedometer	Data JSON	Sistem menampilkan grafik speedometer sesuai dengan isi dari JSON yang dikirimkan	Sistem menampilkan grafik speedometer sesuai dengan isi dari JSON yang dikirimkan	Benar

Dari **Tabel 6.11** dan **Tabel 6.12** terlihat semua fungsi yang dibuat untuk menjalankan sistem sudah berjalan dengan benar semuanya. Dapat ditarik kesimpulan bahwa aplikasi web client monitoring dan kontrling sudah sesuai dengan perancangan yang dibuat sebelumnya

BAB 7 PENUTUP

7.1Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian, serta analisis dari pengimplementasian Websocket untuk Kontroling dan Monitoring Perangkat Berbasis Arduino, dapat disimpulkan bahwa:

1. Untuk mengimplementasikan Websocket pada kedua implementasi yaitu kontroling LED dan monitoring suhu dan kelembapan, digunakan *library twisted txws* pada server dan Websocket API pada aplikasi web client. Dengan mengimplementasikan Websocket, pengguna dapat menjalankan sistem melalui browser seperti layaknya *HTTP*. Untuk melakukan koneksi dengan Websocket, sisi client perlu memasukkan alamat server dengan format ws://alamat server:port. Dengan mengimplementasikan twisted dan Websocket API, sistem yang dibuat mampu membuat komunikasi secara *full-duplex*. Selain itu dengan sistem yang dibuat dengan Websocket mampu melayani beberapa client yang terhubung secara bersamaan secara real time.
2. Untuk komunikasi server dengan Arduino, dalam penelitian ini menggunakan komunikasi serial. Untuk membentuk komunikasi serial, server perlu menuliskan *port USB* serta *baud rate* yang dipakai Arduino. Pada sisi Arduino untuk menerima data dari server, perlu menggunakan fungsi *serial.available* untuk mengecek apakah terdapat data yang masuk ke Arduino. Pada sisi server untuk menerima data dari Arduino perlu menggunakan fungsi *serial.readline* untuk mendapatkan nilai dari Arduino. Dengan menggunakan komunikasi serial, proses pengiriman data akan dikirimkan per bit secara berurutan.
3. Berdasarkan hasil pengujian proses transmisi data, waktu yang dibutuhkan oleh sistem untuk menerima pesan sampai mengirimnya kembali ke client membutuhkan waktu kurang dari 1 detik. Dari hasil pengujian, sistem yang menggunakan Redis memiliki waktu RTT yang tidak jauh berbeda dengan sistem yang tidak menggunakan Redis. Dapat disimpulkan bahwa dengan mengimplementasikan Websocket, sistem yang dibuat dapat bekerja secara real time. Selain itu penggunaan Redis sebagai database tidak terlalu mempengaruhi proses transmisi data dengan memberikan nilai RTT yang tidak jauh berbeda dengan sistem yang tidak menggunakan database.
4. Berdasarkan hasil pengujian kehandalan sistem dalam melayani request per detik, sistem yang menggunakan Redis memiliki kehandalan melayani request yang hampir sama dengan kehandalan sistem yang tidak menggunakan Redis. Kedua sistem sama-sama mampu melayani request dengan jumlah client dibawah 360 client yang terhubung secara bersamaan. Dapat ditarik kesimpulan bahwa sistem yang dibangun dengan Websocket mampu melayani beberapa client yang terkoneksi secara bersamaan dengan jumlah yang cukup banyak. Selain itu penggunaan

database Redis juga tidak mempengaruhi kehandalan sistem hal ini terbukti dengan jumlah request yang terlayani dengan menggunakan Redis hampir sama dengan sistem yang tidak menggunakan database.

7.2Saran

Setelah menyelesaikan penelitian, ada beberapa saran yang dapat disampaikan oleh penulis guna untuk mengembangkan Pengimplementasian Websocket untuk Kontroling dan Monitoring Perangkat Berbasis Arduino :

1. Perlu dilakukan penelitian lebih lanjut mengenai protokol yang lebih cocok untuk mengimplementasikan perangkat berbasis Arduino.
2. Perlu dilakukan penelitian lebih lanjut mengenai perangkat yang lebih beragam, baik itu dalam mikrokontroler yang digunakan maupun perangkat yang digunakan.
3. Perlu dilakukan pengimplementasian yang lebih beragam selain mengontrol lampu LED dan monitoring suhu dan kelembapan.

DAFTAR PUSTAKA

Candelas, F., Garcia, G., Puente, S., Pomares, J., Jara, C., Perez, J., . . . Torres, F. (2015). *Experiences of Using Arduino for Laboratory Experiments of Automatic Control and Robotics.* Alicante: IFAC.

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

Durfee, W. (2011). *Arduino Microcontroller Guide.* University of Minnesota.

Frecon, E. (t.thn.). *Web Like Protocols for the Internet of Things.* ICE.

Heidemann, J., Obraczka, K., & Touch, J. (1997). *Modeling the Performance of HTTP Over Several Transport Protocols* (Vol. 5). IEEE.

Labs, T. M. (2016). *Twisted Documentation.*

Ma, K., & Sun, R. (2013). *Introducing Websocket-Based Real Time Monitoring System for Remote Intelligent Buildings* (Vol. 2013). Jinan: University of Jinan.

McCurdy, A., & Hattem, R. v. (2016). *Redis-py Documentation.*

Megyesi, P., Kramer, Z., & Molnar, S. (t.thn.). *Comparison of Web Transfer Protocols.* Budapest: High Speed Network Laboratory.

Miftakhuddin, M., Suadi, W., & Pratomo, B. A. (t.thn.). *Implementasi Key-Value Store dengan Struktur Data List dan Tree Menggunakan Python.* Institut Teknologi Sepuluh November.

Nugroho, G. P., Mazharuddin, A., & Studiawan, H. (2013). *Sistem Pendeteksi Banjir Menggunakan Sensor Kecepatan Air dan Sensor Ketinggian Air pada Mikrokontroler Arduino* (Vol. 2). Jurnal Teknil Pomits.

Pette, I., & Melkinov, A. (2011). *Websocket Protocol.* IETF.

Pimentel, V., & Nickerson, B. (2011). *Web Display of Real-Time Wind Sensor Data.* Fredericton: University of New Brunswick.

Salzmann, C., Sten, G., Halimi, W., & Gillet, D. (t.thn.). *The Smart Device Specification for Remote Labs.* Lausame: EPFL.

Seguin, K. (t.thn.). *Little Redis Book.*

Srinivasan, L., Scharnagl, J., & Klaus, S. (2013). *Analysis of Websocket as the New Age Protocol for Remote Robot Tele-operation.* Seoul: 3rd IFAC.

W. Evans, B. (2007). *Arduino Programming Notebook* (1 ed.). California: Creative Commons.

Weber, R. (2015). *Internet of Things: Privacy Issues Revisited.* Zurich: University of Zurich.

Zhang, W., Passow, P., Jovanov, E., Stoll, R., & Thurow, K. (t.thn.). *A Seccure and Scalable Telemonitoring System Using Ultra Low Energy Wireless Sensor*

Interface for Long Term Monitoring In Life Science Applications. Rosctock: University of Rosctock.