RANCANG BANGUN TRANSMISI DATA *HEART RATE* MENGGUNAKAN PROTOKOL MQTT

TUGAS AKHIR

Program Studi

S1 Teknik Komputer

Oleh:

Muhammad Reza Bintami

15410200034

FAKULTAS TEKNOLOGI DAN INFORMATIKA
INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA
2019

RANCANG BANGUN TRANSMISI

DATA *HEART RATE* MENGGUNAKAN PROTOKOL MQTT

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Teknik

Disusun Oleh :

Nama : Muhammad Reza Bintami

NIM : 15410200034

Program : S1 (Strata Satu)

Jurusan : Teknik Komputer

FAKULTAS TEKNOLOGI DAN INFORMATIKA

INSTITUT BISNIS DAN INFORMATIKA STIKOM SURABAYA

2019

“KESEMPATAN ADALAH HARTA MULIA”

Kupersembahkan Kepada

ALLAH SWT

Ibu, Bapak dan semua keluarga,

Yang selalu mendukung, memotivasi dan mendoakan yang terbaik untuk saya.

Beserta semua orang dan rekan-rekan S1 Sistem Komputer yang selalu membantu, mendukung dan memotivasi agar tetap berusaha menjadi lebih baik.

TUGAS AKHIR

RANCANG BANGUN TRANSMISI DATA *HEART RATE* MENGGUNAKAN PROTOKOL MQTT

Dipersiapkan dan disusun oleh

Muhammad Reza Bintami

NIM : 15410200034

Telah diperiksa, diuji dan disetujui oleh Dewan Penguji

Pada : 22 Juli 2019

Susunan Dewan Penguji

Pembimbing

I. **Ira Puspasari, S.Si., M.T.**
NIDN. 0710078601

II. **Musayyanah, S.ST., M.T.**
NIDN. 0730069102

Pembahas

I. **Heri Pratikno, M.T., MTCNA., MTCRE.**
NIDN. 071611702

Tugas Akhir ini telah diterima sebagai salah satu persyaratan untuk memperoleh gelar Sarjana

29/7/19

Dr. Jusak

Dekan Fakultas Teknologi dan Infomatika

SURAT PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Institut Bisnis dan Informatika Stikom Surabaya, saya :

Nama	:	Muhamad Reza Bintami
NIM	:	15410200034
Program Studi	:	S1 Teknik Komputer
Fakultas	:	Fakultas Teknologi dan Informatika
Jenis Karya	:	Tugas Akhir
Judul Karya	:	**RANCANG BANGUN TRANSMISI DATA *HEART RATE* MENGGUNAKAN PROTOKOL MQTT**

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Institut Bisnis dan Informatika Stikom Surabaya Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right)* atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data *(database)* untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta
2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 22 Juli 2019

Yang menyatakan

Muhammad Reza Bintami

NIM : 15410200034

ABSTRAK

Olahraga merupakan salah satu aktivitas yang menjadi gaya hidup masyarakat modern. Hal ini terlihat dari banyaknya aktivitas yang dilakukan oleh masyarakat terutama orang dewasa maupun orang tua. Disamping itu olahraga mempunyai kendala yang cukup besar bagi seseorang yang mempunyai penyakit seperti jantung. Pada tahun 2013 terjadi kejadian seorang pelari mengikuti Jakarta Marathon 2013 yang mendadak pingsan saat lomba lari, sebelum meninggal dunia akibat sakit jantung.

Di penelitian sebelumnya integrasi *hardware* yang meliputi sensor *Grove Finger Clip Heart Rate,* Modul Transmisi *Wireless* (Modul ESP8266), dan Arduino. Performansi 0,6% untuk tingkat kesalahan sensor dan kemampuan transmisi data dengan paket loss <1% untuk modul ESP. *Hardware Monitoring* tersebut terintegrasi baik dengan aplikasi *monitoring* lewat jaringan lokal dan akses *web* lewat jaringan internet.

Pada penelitian ini penulis membuat rancang bangun transmisi data *heart rate* menggunakan protokol MQTT. Pada rancangan tersebut telah ditetapkan oleh penulis bahwa terdapat QoS 0 dan QoS 1 pada protokol MQTT. Sehingga *prototype* mengirimkan nilai data *heart rate* menggunakan QoS 0 dan QoS 1 secara bersamaan, kemudian diterima oleh *broker* sebagai penghubung antara *prototype* (*Publisher*) dan aplikasi *desktop* maupun *smartphone* (*Subscriber*).

Broker yang digunakan oleh penulis adalah *Mosquitto*. Hasil pengujian yang diperoleh dengan membandingkan *Quality of Service* jaringan protokol MQTT dan TCP/IP dengan parameter *delay, throughput* dan *packet loss*. Hasil yang didapatkan

dari perbandingan tersebut yaitu : *delay* pada QoS *level* 0 sebesar 17,39 ms untuk PC dan 9,37 ms untuk *smartphone*, QoS *level* 1 sebesar 20,21 ms untuk PC dan 13,11 ms untuk *smartphone*, TCP/IP sebesar 925,42 ms. *Throughput* pada QoS *level* 0 sebesar 516,18 bps, pada QoS *level* 1 sebesar 532,29 bps, TCP/IP sebesar 4111,87 bps. *Packet loss* pada QoS *level* 0 sebesar 0,17% untuk PC dan 0,13% untuk *smartphone*, pada QoS *level* 1 sebesar 0,2% untuk PC dan 0,2% untuk *smartphone*, TCP/IP sebesar 2,48% sehingga tergolong sangat bagus.

Kata Kunci : MQTT, *Heart rate*, *Quality of Service*, TCP/IP, *Mosquitto*.

KATA PENGANTAR

Pertama-tama penulis panjatkan puji dan syukur atas kehadiran Allah SWT, karena berkat izin, Rahmat dan hidayah-nya penulis dapat menyelesaikan laporan penelitian ini yang merupakan salah satu syarat menempuh Tugas Akhir pada Program Studi S1 Teknik Komputer di Institut Bisnis dan Informatika Stikom Surabaya. Shalawat serta salam tidak lupa selalu penulis panjatkan kepada Rasulullah SAW.

Di dalam buku Tugas Akhir ini dilakukan pembahasan mengenai rancang bangun transmisi data *Heart Rate* menggunakan protokol MQTT. Dalam usaha menyelesaikan Tugas Akhir ini penulis banyak mendapatkan bantuan dari berbagai pihak baik moral maupun materi. Oleh karena itu penulis mengucapkan terima kasih dan penghargaan setinggi-tingginya kepada:

1. Orang tua dan saudara-saudara saya tercinta yang telah memberikan dukungan dan bantuan baik moral maupun materi sehingga penulis dapat menempuh dan menyelesaikan Tugas Akhir maupun laporan ini.
2. Kepada Ibu Ira Puspasari, S.Si., M.T. dan Ibu Musayyanah, S.ST., M.T. selaku Dosen Pembimbing. Kemudian Bapak Heri Pratikno, M.T., MTCNA., MTCRE. selaku Dosen Pembahas. Terima kasih atas bimbingan yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir dengan baik.
3. Kepada Bapak Pauladie Susanto, S.Kom., M.T., selaku Ketua Program Studi Teknik Komputer Surabaya atas ijin yang diberikan untuk mengerjakan Tugas Akhir ini.

4. Semua staf dosen yang telah mengajar dan memberikan ilmunya.
5. Terima kasih terhadap seluruh rekan-rekan S1 Teknik Komputer khususnya rekan-rekan seperjuangan angkatan 2015 khususnya Prodi S1 Teknik Komputer yang selalu memberikan semangat dan bantuannya.
6. Serta semua pihak lain yang tidak dapat disebutkan secara satu per satu, yang telah membantu dalam menyelesaikan Tugas Akhir ini baik secara langsung maupun tidak langsung.

Penulis berharap semoga laporan ini dapat berguna dan bermanfaat untuk menambah wawasan bagi pembacanya. Penulis juga menyadari dalam penulisan buku Tugas Akhir ini masih terdapat banyak kekurangan. Oleh karena itu penulis berharap adanya saran maupun kritik dalam memperbaiki kekurangan dan berusaha untuk lebih baik lagi kedepannya.

Surabaya, 22 Juli 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Olahraga merupakan salah satu aktivitas yang menjadi gaya hidup masyarakat modern. Hal ini terlihat dari banyaknya aktivitas yang dilakukan oleh masyarakat terutama orang dewasa dan orang tua. Disamping itu olahraga mempunyai kendala yang cukup besar bagi seseorang yang mempunyai penyakit seperti jantung. Pada tahun 2013 terjadi kejadian seorang pelari di Jakarta Marathon 2013 yang mendadak pingsan saat lomba lari sebelum meninggal dunia akibat sakit jantung dan intensitas latihan lari yang dilakukan tinggi. Menurut (Michael, 2013), jika seseorang melakukan olahraga secara rutin, maka dihimbau untuk memeriksa denyut jantung (*heart rate*). Selain itu, diperlukan proses pemanasan dan pendinginan sebelum dan sesudah berolahraga.

Dalam kehidupan sehari-hari banyak para pelari yang sering lupa akan denyut jantung mereka. Perhitungan denyut jantung sering kali masih dilakukan secara manual yaitu dengan cara menghitung denyut nadi di pergelangan tangan selama satu menit. Metode ini dianggap kurang akurat karena mengandalkan indera peraba manusia. Selain

perhitungan denyut jantung secara manual, metode yang sering digunakan adalah metode elektrik yaitu dengan menggunakan

Electrocardiograph (ECG) dan metode *Photoplethysmography* (PPG). Hasil yang diberikan oleh ECG memiliki keakuratan yang lebih baik karena menggunakan sensor yang diletakkan pada bagian tertentu tubuh untuk mendeteksi detak jantung. Akan tetapi alat ini berukuran besar dan mahal sehingga sulit digunakan secara mobile, yang mengakibatkan kurang efisiennya penggunaan alat tersebut. Sedangkan metode PPG merupakan metode pengukuran denyut jantung yang berbasis cahaya led, dimana pengukurannya melibatkan perubahan volume darah berdasarkan variasi intensitas cahaya yang lewat atau dipantulkan organ tubuh manusia (Yulian, 2017) serta pengukurannya dilakukan di ujung jari. Berdasarkan hal tersebut, maka teknik pengambilan denyut jantung pada proposal ini menggunakan sensor yang berbasis PPG, yaitu dengan menggunakan sensor *Grove-Finger Clip*.

Berdasarkan perkembangan era teknologi IoT (*Internet of Things*), pantauan HR untuk atlit lari atau pemula dapat terintegrasi dengan jaringan internet, yang dapat diakses dengan web atapun aplikasi tertentu, sehingga pantauan untuk seorang atlit yang dilakukan oleh pelatih lebih mudah dan efesien.

Integrasi *hardware* yang meliputi sensor *Grove Finger Clip Heart Rate*, Modul Transmisi *Wireless* (Modul ESP 8266) , dan Arduino. Performansi 0,6% untuk tingkat kesalahan sensor dan kemampuan transmisi data dengan paket loss <1% untuk modul ESP. *Hardware Monitoring* tersebut terintegrasi baik dengan aplikasi *monitoring* lewat jaringan lokal dan akses web lewat jaringan internet (Musayyanah, 2018).

Pada penelitian (Puspasari, 2018), yang mengukur HR dengan sensor dan dilengkapi dengan sms *gateway* (*telereport*) untuk notifikasi THR. Hasil pengiriman tersebut masih memiliki delay pengiriman 28,52 detik.

Protokol MQTT sering digunakan terutama karena konsumsi daya yang rendah, bandwidth rendah, skalabilitas tinggi dan overhead yang sangat rendah. Karena keunggulannya, MQTT telah diterapkan secara luas untuk kontrol industri (SCADA), penandaan rel (Delta-rail), pemantauan transportasi realtime (*Red Funnel*), otomatisasi, perawatan kesehatan dan aplikasi jejaring sosial seperti Facebook (Govindan, 2015).

Protokol MQTT merupakan salah satu protokol yang digunakan untuk transmisi data dengan data rendah, basis dari komunikasi protokol ini adalah *publish-subscriber*. Beberapa penelitian yang menggunakan protokol ini seperti sistem remote *wireless* remote cerdas, menentukan kuota tempat parkir. MQTT digunakan untuk transmisi dengan daya rendah dan bisa digunakan untuk optimasi jaringan dengan daya yang rendah (Saputra, 2017).

Berdasarkan kelebihan dari protokol MQTT dan permasalahan penelitian, maka diperlukan sebuah sistem komunikasi untuk memantau *heart rate* pelari bagi pelatih secara *realtime* dengan memusatkan kinerja pada protokol MQTT tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, dapat dirumuskan permasalahannya adalah

1. Bagaimana mengintegrasikan *prototype heart rate* pelari dengan *server*/*broker* dari protokol MQTT.
2. Bagaimana kinerja QoS dari protokol MQTT untuk transmisi *heart rate* pada *monitoring* kondisi pelari.

1.3 Batasan Masalah

Dalam perancangan ini, terdapat beberapa batasan masalah, antara lain :

1. Sampel pengujian yang aktif berolahraga yang berusia 20-25 tahun.
2. *Sampling* data adalah 1000 ms.
3. Topologi sistem yang digunakan adalah topologi *end to end wireless*.
4. Perangkat Lunak yang digunakan adalah MQTT *Mosquitto Opensource.*

1.4 Tujuan

Tujuan perancangan alat ini adalah

1. Mengintegrasikan *prototype heart rate* pelari dengan *server/broker* dari protokol MQTT.
2. Mengetahui kinerja QoS dari protokol MQTT untuk transmisi *heart rate* pada monitoring kondisi pelari.

1.5 Sistematika Penulisan

BAB I PENDAHULUAN

Pada bab ini dijelaskan tentang latar belakang rumusan masalah batasan masalah, tujuan dari penelitian ini, dan sistematika penulisan tugas akhir.

BAB II LANDASAN TEORI

Pada bab ini membahas teori penunjang baik secara metode, software dan hardware secara singkat sebagai acuan pada penelitian tugas akhir.

BAB III METODE PENELITIAN

Bab ini berisi tentang pengujian sistem yang meliputi prosedur perancangan perangkat lunak baik secara program, metode MQTT beserta QoS tiap *level* serta perangkat keras.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini dibahas tentang pengujian dari perangkat keras sebagai *publisher* dengan *subscriber*, hasil pengujian keseluruhan sistem dan hasil pengujian QoS tiap *level*.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan dari penelitian yang telah dilakukan penulis serta saran kedepan sebagai pengembangan penelitian.

BAB II

LANDASAN TEORI

2.1 *Heart Rate*

Heart Rate adalah jumlah detak jantung per satuan waktu atau bisa dinyatakan dalam denyut per menit atau *beats per minute* (bpm). Detak jantung bervariasi, tergantung pada kebutuhan tubuh untuk menyerap Oksigen dan mengeluarkan CO2 dalam berbagai keadaan, misalnya saat olah raga atau tidur. *Heart Rate* bisa diukur di belakang lutut, paha bagian dalam, leher, kaki, pelipis, pergelangan tangan dan jari. *Heart rate* normal orang dewasa adalah 60-100 bpm. Terdapat faktor yang mempengaruhi *heart rate* antara lain adalah: aktifitas fisik, suhu udara sekitar, posisi tubuh (tidur/berdiri), tingkatan emosi, usia dan obat-obatan yang sedang dikonsumsi. Jadi heart rate orang yang dalam kondisi beristirahat, habis olahraga, sedang mendaki, sedang, emosi ataupun sedang bahagia itu berbeda.

2.2 Mikrokontroler

Arduino adalah pengendali mikro single-board yang bersifat open-source, diturunkan dari *Wiring platform*, yang dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Pada Gambar 2.1 terlihat bahwa tipe Arduino ada bermacam-macam. *Hardware*-nya memiliki prosesor Atmel AVR dan *software*-nya memiliki bahasa pemrograman sendiri seperti bahasa C dan bahasa mesin.

Kelebihan dari Arduino, yaitu:

- Bahasa pemrograman relatif mudah karena *software* Arduino dilengkapi dengan kumpulan *library* yang cukup lengkap.

- Mendukung komunikasi USB, sehingga pengguna *laptop* yang tidak memiliki *port* serial/RS323 bisa menggunakan nya.
- Tidak perlu perangkat chip *programmer* karena di dalamnya sudah ada *bootloader* yang akan menangani *upload* program dari komputer.
- Memiliki modul siap pakai (*shield*) yang bisa ditancapkan pada *board* Arduino. Misalnya shield GPS, Ethernet, SD Card, dan lain-lain.

Gambar 2.1 Tipe-tipe Arduino

2.3 Modul ESP 8266

ESP8266 merupakan modul *wifi* yang berfungsi sebagai perangkat tambahan mikrokontroler seperti Arduino agar dapat terhubung langsung dengan *wifi* dan membuat koneksi TCP/IP.

Gambar 2.2 ESP8266-01

Pada Gambar 2.2 tersebut membutuhkan daya sekitar 3.3V dengan memiliki tiga *mode* wifi yaitu *Station*, *Access Point* dan *Both* (Keduanya). Modul ini juga dilengkapi dengan *prosesor*, memori dan GPIO dimana jumlah *pin* bergantung dengan jenis ESP8266 yang kita gunakan. Sehingga modul ini bisa berdiri sendiri tanpa menggunakan mikrokontroler apapun karena sudah memiliki perlengkapan layaknya mikrokontroler.

Firmware default yang digunakan oleh perangkat ini menggunakan AT Command, selain itu ada beberapa Firmware SDK yang digunakan oleh perangkat ini berbasis *opensource* yang diantaranya adalah sebagai berikut:

- NodeMCU dengan menggunakan *basic programming* lua
- MicroPython dengan menggunakan *basic programming* python
- AT *Command* dengan menggunakan perintah AT *command*

2.4 *Finger Clip Heart Rate* Sensor

Finger clip Heart Rate Sensor yang merupakan sensor detak jantung berbasis PAH8001EI-2G. Modul tersebut bekerja berdasarkan teknologi optik yang dapat mengukur variasi gerakan darah pada pembuluh darah manusia. Pada Gambar 2.3 dapat mendeteksi detak jantung yang terintegrasi dengan STM32F102 (ARM Cortex-M3), dan dilengkapi dengan antarmuka SWD yang memungkinkan user memprogram ulang.

Gambar 2.3 Sensor *Grove Finger Clip Heart Rate*

2.5 OLED

Dalam perancangan suatu system perangkat keras banyak sekali yang menggunakan OLED sebagai tambahannya. Pada Gambar 2.4 OLED berfungsi sebagai penampil suatu nilai hasil sensor, menampilkan teks, atau menampilkan menu pada aplikasi mikrokontroler. OLED yang digunakan adalah jenis OLED M1632. OLED M1632 merupakan modul dengan tampilan 128x64 baris dengan konsumsi daya rendah.

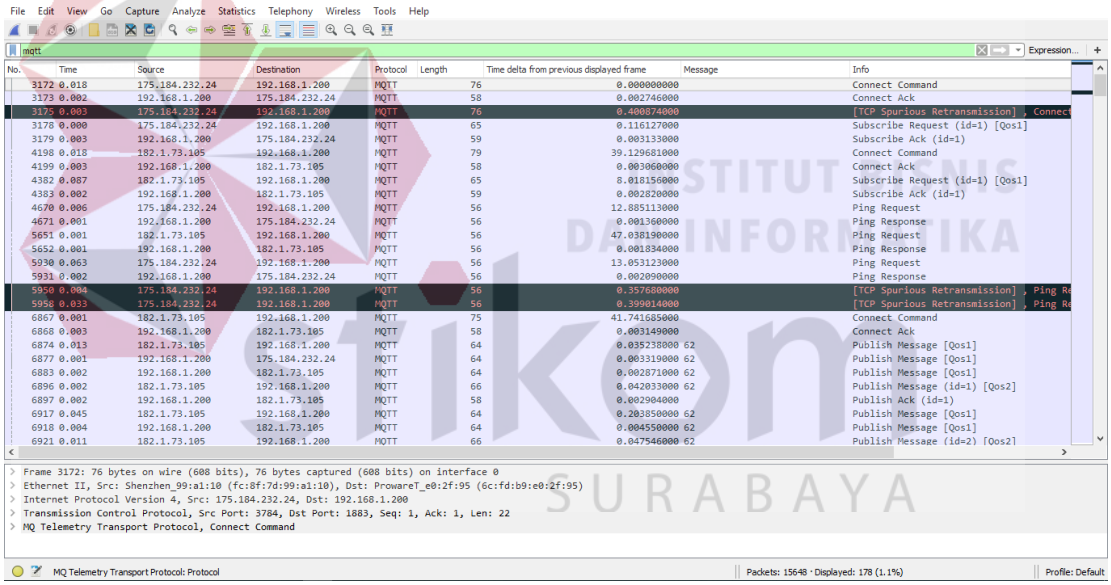
Gambar 2.4 OLED 128x64

2.6 *Wireshark*

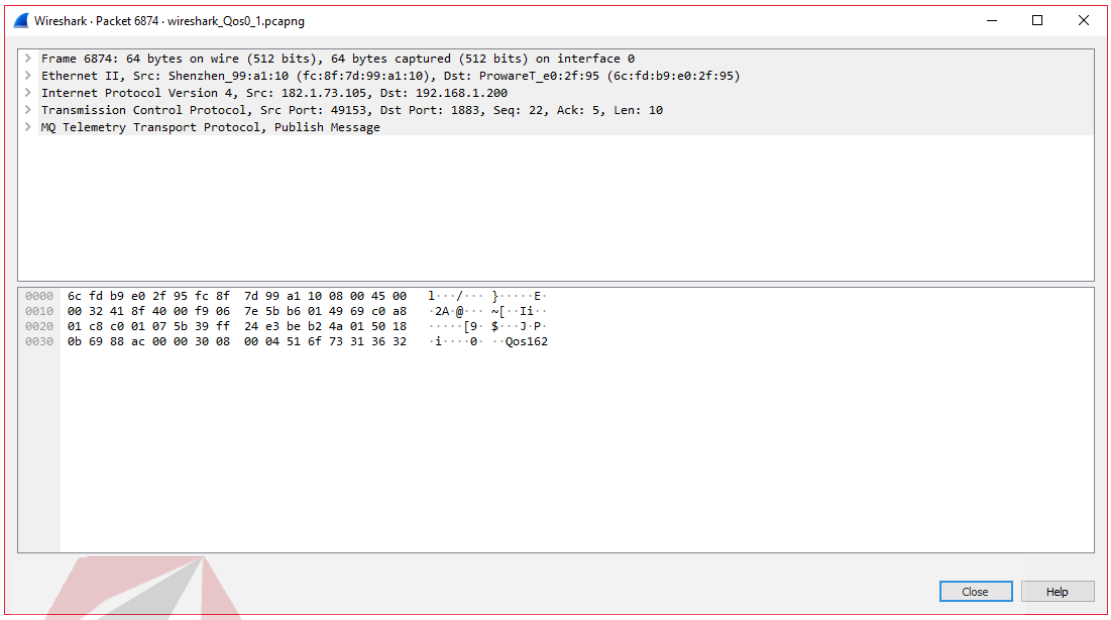
Salah satu dari *tool Network Analyzer* yang digunakan oleh *Network Administrator* pemecahan masalah jaringan, analisis, perangkat lunak dan pengembangan protokol komunikasi, dan pendidikan. *Wireshark* mampu merekam paket-paket data yang ada didalam jaringan seperti Gambar 2.5, sehingga semua jenis paket termasuk informasi data dalam berbagai format protokol pun akan dengan mudah dianalisa Gambar 2.26. *Wireshark* memiliki fitur yang lengkap yaitu :

- *Multiplatform* – Bisa dipakai untuk beberapa basis system operasi (Unix, Mac, Windows, serta Linux)
- Bisa lakukan *capture* paket data jaringan secara real time
- Bisa menampilkan informasi protokol jaringan dari paket data secara komplit
- Paket data bisa disimpan jadi *file* serta nantinya bisa di buka kembali untuk analisa lebih lanjut

- *Filtering* paket data jaringan
- Pencarian paket data dengan persyaratan spesifik
- Pewarnaan penampilan paket data untuk memudahkan analisis paket data
- Menampilkan data statistik
- Untuk lakukan *capture* paket data yang keluar maupun masuk pada jaringan, *wireshark* membutuhkan piranti fisik NIC (*Network Interface Card*).



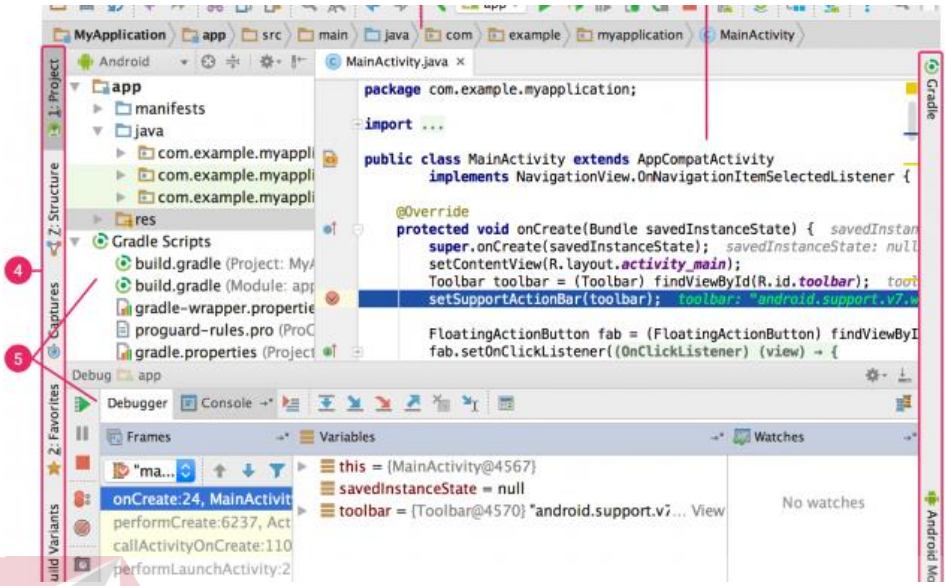
Gambar 2.5 Hasil *Capture*



Gambar 2.6 Isi Paket Yang Sudah Direkam

2.7 Android Studio

Android Studio adalah IDE resmi Android. Tujuannya dibuat untuk Android adalah untuk mempercepat pengembangan dan membantu pengguna dalam hal membuat aplikasi berkualitas tinggi untuk setiap perangkat Android. Pada perangkat Android sendiri memiliki *platform* yang terdiri dari Sistem Operasi berbasis Linux, sebuah GUI (*Graphic User Interface*), sebuah *web browser* dan Aplikasi *End-User* yang dapat di *download* dan juga para pengembang bisa dengan leluasa berkarya serta menciptakan aplikasi yang terbaik dan terbuka untuk digunakan oleh berbagai macam perangkat.



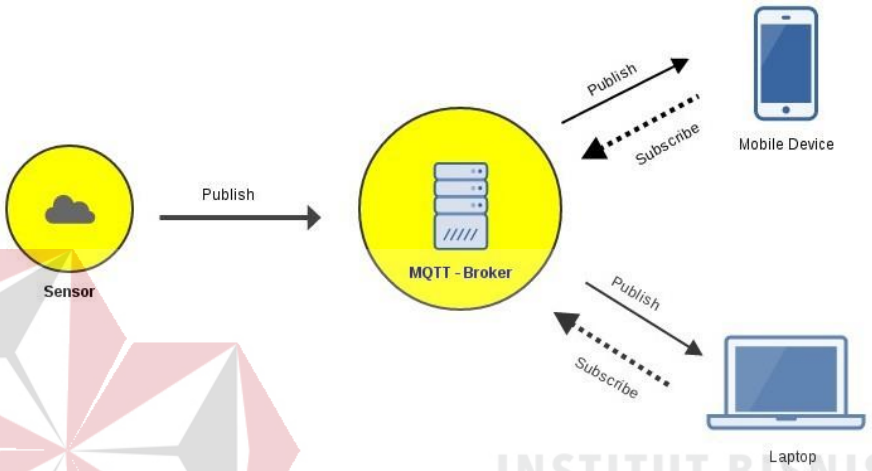
Gambar 2.7 Antarmuka Android Studio

2.8 MQTT

Govindan (2015: 290) berpendapat bahwa MQTT-SN telah dikembangkan untuk *Wireless Sensor Network* (WSN). MQTT-SN dapat dianggap sebagai versi MQTT yang disesuaikan dengan lingkungan komunikasi nirkabel. Tidak seperti MQTT, dalam sistem MQTT-SN, node akhir akan terhubung ke Gateway menggunakan protokol MQTT-SN melalui nirkabel kemudian *Gateway* terhubung ke *server/broker* menggunakan protokol MQTT melalui jaringan kabel.

Protokol MQTT (*Message Queuing Telemetry Transport*) adalah protokol yang berjalan pada di atas stack TCP/IP dan mempunyai ukuran paket data dengan *low overhead* yang kecil (minimum 2 *bytes*) sehingga berefek pada konsumsi catu daya yang juga cukup kecil.

Protokol ini adalah jenis protokol *data-agnostic* yang artinya bisa mengirimkan data apapun seperti data *binary*, *text* bahkan XML ataupun JSON dan protokol ini memakai model *publish/subscribe* daripada model *client-server*.



Gambar 2.8 Sistem Umum IoT Menggunakan MQTT

Stack TCP/IP sekarang sudah banyak didukung oleh mikrokontroler seperti seri STM32Fx7 maupun *device board* yang umum dipasaran seperti Arduino Yun, Arduino + *Ethernet Shield*, ESP8266 WiFi SoC, Raspberry Pi dan sebagainya. Jadi sebenarnya banyak pilihan untuk belajar IoT dengan memakai protokol ini ditambah lagi harga *device* yang semakin murah dan terjangkau.

Sistem umum MQTT seperti pada gambar di atas membutuhkan dua komponen perangkat lunak utama yaitu:

- MQTT *Client* yang nantinya akan di *install* di *device*. Untuk Arduino anda bisa memakai *library pubsubclient*, pustaka seperti *library* mqtt.js bisa dipakai pada platform Node.js di Raspberry Pi ataupun laptop.

- MQTT *Broker* yang berfungsi untuk menangani *publish* dan *subscribe* data. Untuk *platform Node*.js anda bisa memakai *broker* mosca sedangkan untuk *platform* lain lebih banyak tersedia seperti *mosquitto*.

Keuntungan dari sistem *publish/subscribe* adalah antara sumber pengirim data (*publisher*) dan penerima data (*client*) tidak saling mengetahui karena ada *broker* diantara mereka atau istilah kerennya yaitu *space decoupling* dan yang lebih penting lagi yaitu adanya time *decoupling* dimana *publisher* dan klien tidak perlu terkoneksi secara bersamaan, misalnya klien bisa saja *disconnect* setelah melakukan *subscribe* ke *broker* dan beberapa saat kemudian klien *connect* kembali ke *broker* dan klien tetap akan menerima data yang terpending sebelumnya proses ini dikenal dengan *mode offline*.

MQTT Broker

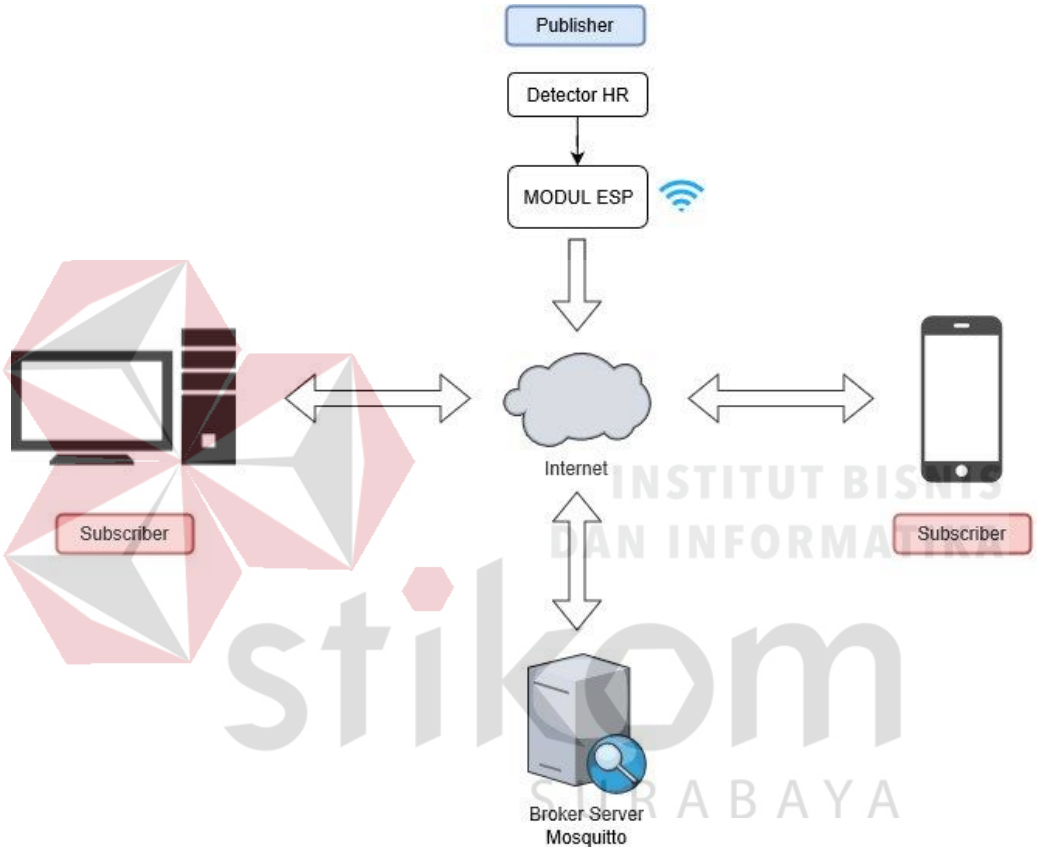
Yiming Xu (2016:2) berpendapat bahwa mesin *server* yang bertindak sebagai simpul pusat, menghubungkan antara MQTT *publishers* dengan MQTT *subscribers*. *Node broker* mengumpulkan *publish* data yang dihasilkan oleh perangkat IoT. Server/Broker MQTT menggunakan perangkat lunak *Mosquitto* yang dapat berjalan pada sistem operasi Windows, ataupun Linux. *Eclipse mosquitto* ™ merupakan broker pesan *open source* (EPL / EDL berlisensi) yang mengimplementasikan MQTT protokol versi 3.1 dan 3.1.1. MQTT menyediakan metode yang ringan untuk dapat mempublikasikan / berlangganan pesan sesuai dengan topik pesan apa yang diinginkan. Hal ini membuat *eclipse mosquito* cocok untuk "*Internet of Things*" dengan sensor daya rendah atau perangkat *mobile* seperti ponsel, sistem tertanam atau mikrokontroler seperti Arduino. *Server broker* MQTT secara default menggunakan *port* 1833

walaupun di dalam implementasinya server MQTT dapat menggunakan beberapa *port* lain dengan fungsi yang berbeda.

BAB III

METODE PENELITIAN

Pada bab ini membahas tentang proses komunikasi dan model sistem yang akan diterapkan :



Gambar 3.1 Blok Diagram

Pada Gambar 3.1 terdiri dari satu jaringan untuk transmisi data *heart rate* (HR), dimana area jangkauan pada jaringan *wireless* lebih fleksibel. Dimana nantinya di sisi *publisher* maupun *subscriber* akan terinstall MQTT *Mosquitto* untuk menentukan topik. Topik tersebut nantinya digunakan oleh *Broker* untuk memfilter pesan yang akan dikirim ke *subscriber* yang terdaftar. Misalnya Topik: Qos1 yang menghubungkan *publisher* dan *subscriber* untuk akses jaringan *wireless* oleh *Broker*. *Publisher*

mengirimkan data *heart rate* ke *subscriber* (*Level* 0), jika *subscriber* tidak mengenali data tersebut, maka *Broker* akan menginfokan ke *Publisher* bahwa *subscriber* telah menerima data yang dikirim (*Level* 1). Setiap Tahapan (*Level*) proses transmisi *publisher*, *subscriber* dan *broker* di atas, akan dihitung parameter *delay*, *throughput* dan *packet loss* dari setiap *level* dengan pengiriman data *payload* yang berbeda kapasitasnya. Hal ini berbeda jauh dengan TCP/IP sehingga tidak mendukung *Publish* dan *Subscriber* sebagai protokol, dimana tujuan dari MQTT jauh lebih sederhana dan terfokus dibandingkan dengan TCP/IP.

3.1 Perancangan Jaringan MQTT

Secara umum gambar pada Blok Diagram pada rancangan jaringan MQTT.

Gambar 3.2 Blok Diagram Sistem Jaringan MQTT *Mosquitto*

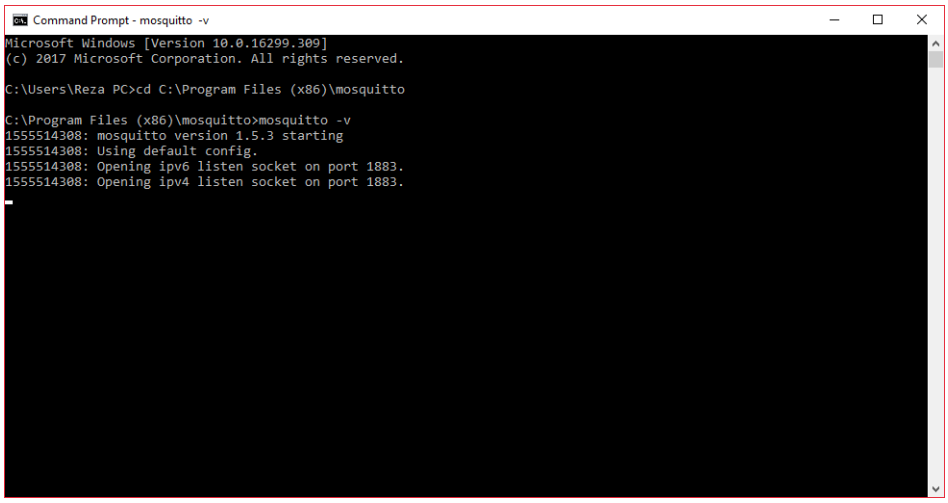
Penjelasan setiap bagian dari diagram blok sistem jaringan MQTT *Mosquitto* pada Gambar 3.2 sebagai berikut:

a) *Publisher*:

Dimana sebuah *client* mengirimkan sebuah pesan tanpa mengetahui keberadaan *client* yang lainnya dan bergantung pada topik yang diberikan. Misalnya dengan nama “Qos1”.

b) *Broker*:

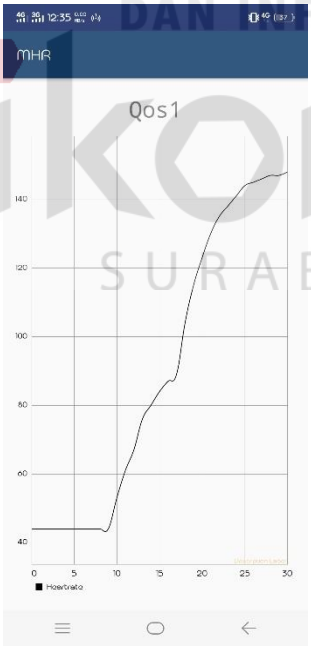
Sebuah *Server* (Gambar 3.3) yang berupa *command prompt* bertujuan untuk mem-*filter* pesan yang masuk dan akan meneruskan kepada *client* dengan topik yang sama.



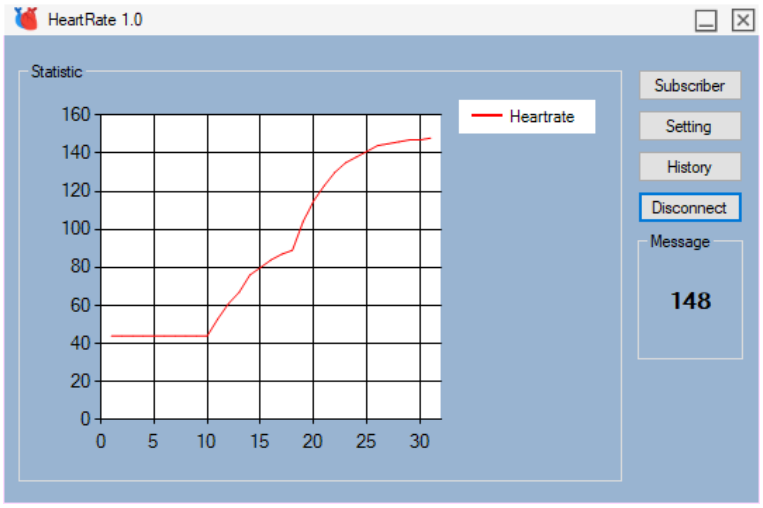
Gambar 3.3 MQTT *Mosquitto*

c) *Subscriber*:

Sebuah *Client* yang bertujuan untuk menerima pesan berdasarkan topik yang dituju. Dengan contoh pada Gambar 3.4 dan Gambar 3.5 berikut :



Gambar 3.4 *Subscriber* Pada S*martphone* Android



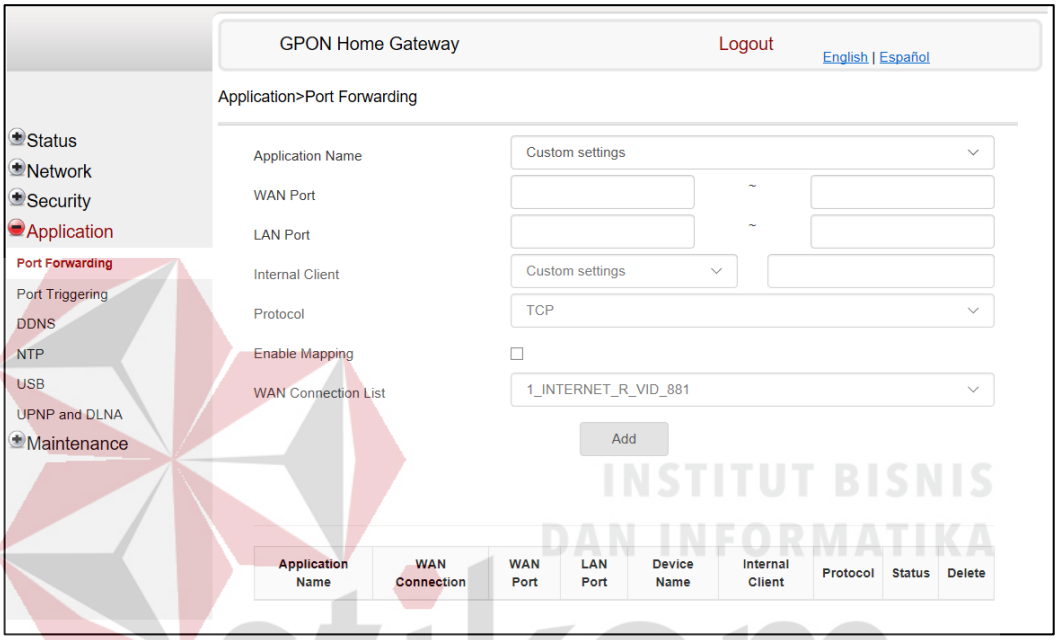
Gambar 3.5 *Subscriber* Pada *Desktop*

3.2 Perancangan Jaringan Internet

Dalam proses perancangan jaringan internet, penulis menggunakan ISP indihome untuk bisa mengakses ke jaringan internet. Pada jasa penyedia ISP tersebut terdapat router Nokia yang dapat diatur melalui IP *gateway* yang ditunjukkan pada Gambar 3.6 di bawah ini :

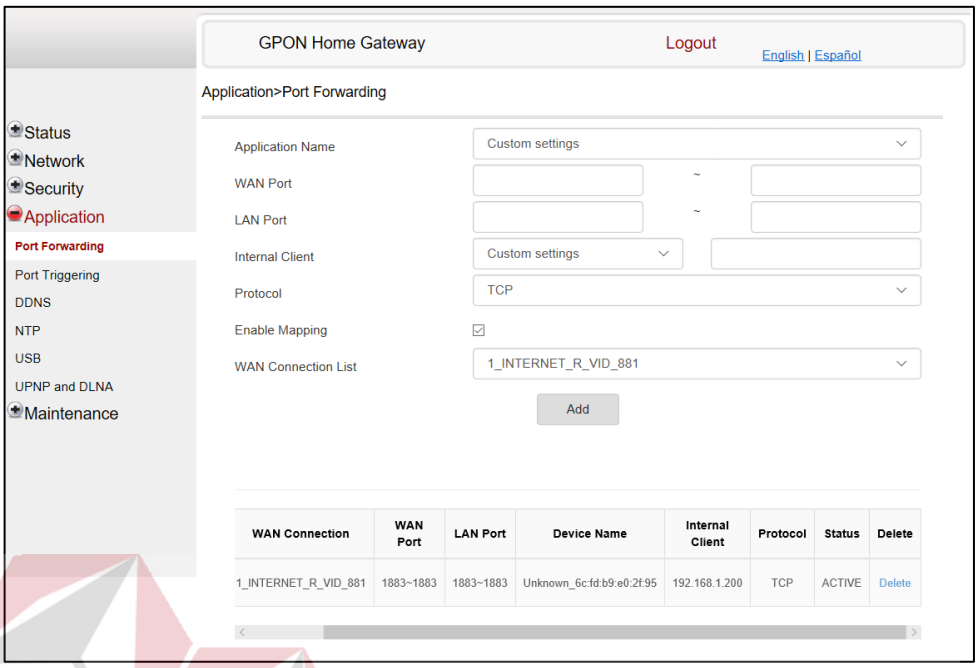
Gambar 3.6 Halaman *Login* Router

Untuk mengaturnya perlu *login* pada *router* Nokia. Pada Gambar 3.7 di bawah ini isikan WAN port yang akan dibuka pada jaringan internet, isikan LAN port 1883 yaitu port MQTT. Kemudian pada internal client isi IP Komputer/*Broker* dan centang *Enable Mapping*.



Gambar 3.7 Menu Router

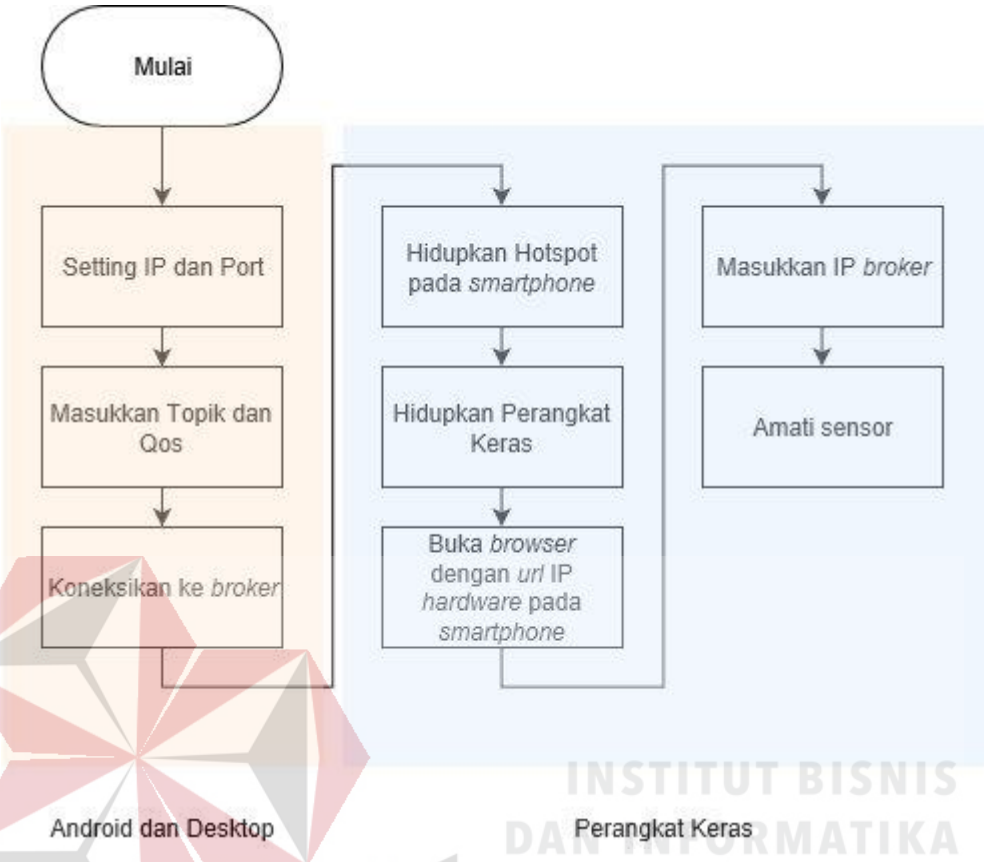
Jika pengaturannya sudah selesai, klik Add dan tabel yang sebelumnya kosong akan terisi seperti Gambar 3.8 sebagai berikut :



Gambar 3.8 *Port Forwarding* Aktif

3.3 Perancangan Keseluruhan Sistem

Pada Gambar 3.9 merupakan perancangan keseluruhan sistem dari segi *Software* maupun *Hardware*, *Software* program berupa aplikasi android dan *desktop* dimana program tersebut akan membaca nilai sensor yang dikirim dari perangkat keras. Untuk *Hardware* berupa pemasangan setiap modul antara Arduino pro mini dan ESP8266.



Gambar 3.9 Keseluruhan Sistem

Komponen yang digunakan untuk rancang bangun transmisi data *heart rate* sebagai berikut :

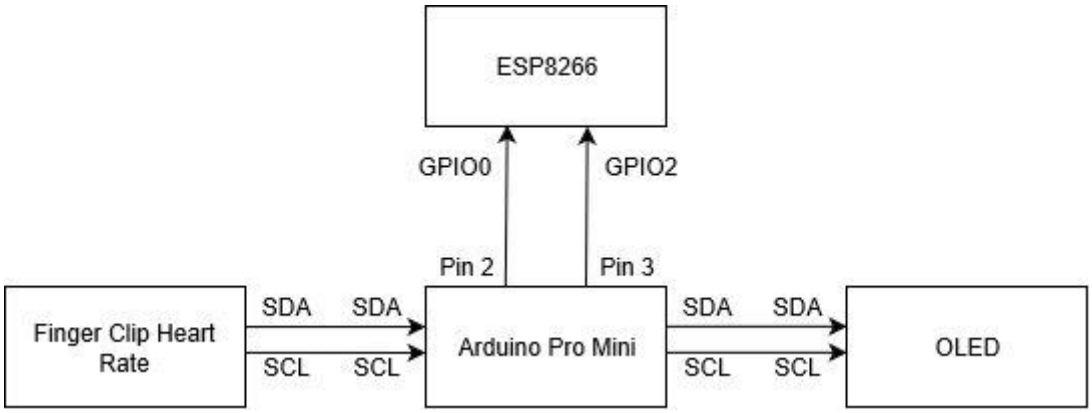
1. Arduino Pro Mini
2. ESP8266
3. Sensor Finger Clip *Heart Rate*
4. *Battery*
5. OLED 128x64
6. Smartphone android
7. PC atau Laptop

Prosedur yang dilakukan untuk proses transmisi data *heart rate* melalui langkah-langkah sebagai berikut :

1. Buka aplikasi HRM yang ada pada *smartphone* dan *desktop*.
2. Atur IP dan port *broker* pada menu *setting* dengan port 1883.
3. Masukkan topik dan pilih QoS yang sesuai pada menu *subscribe*.
4. Pada *desktop* tekan *connect* dan pada *smartphone* tekan menu *statistic* untuk menunggu nilai sensor yang akan di *publish* dari perangkat keras.
5. Hidupkan *hotspot* wifi yang ada pada *smartphone* dengan nama smart dan password smartHRM.
6. Hidupkan perangkat keras.
7. Buka *browser* pada *smartphone* dengan url dari IP perangkat keras.
8. Masukkan IP *broker*.
9. Amati nilai sensor pada *smartphone* dan *desktop* dengan aplikasi HRM.

3.4 Perancangan Perangkat Keras (*Detector* HR) dan *Flowchart* Pemrograman

Pada perancangan perangkat keras, terdapat rangkaian perangkat keras ditunjukkan pada Gambar 3.10 di bawah ini :



Gambar 3.10 Rangkaian Perangkat Keras

Komponen yang digunakan untuk rancang bangun transmisi data *heart rate* sebagai berikut :

1. Sensor *Finger Clip Heart rate*
2. Arduino Pro Mini
3. OLED
4. ESP8266
5. *Casing*

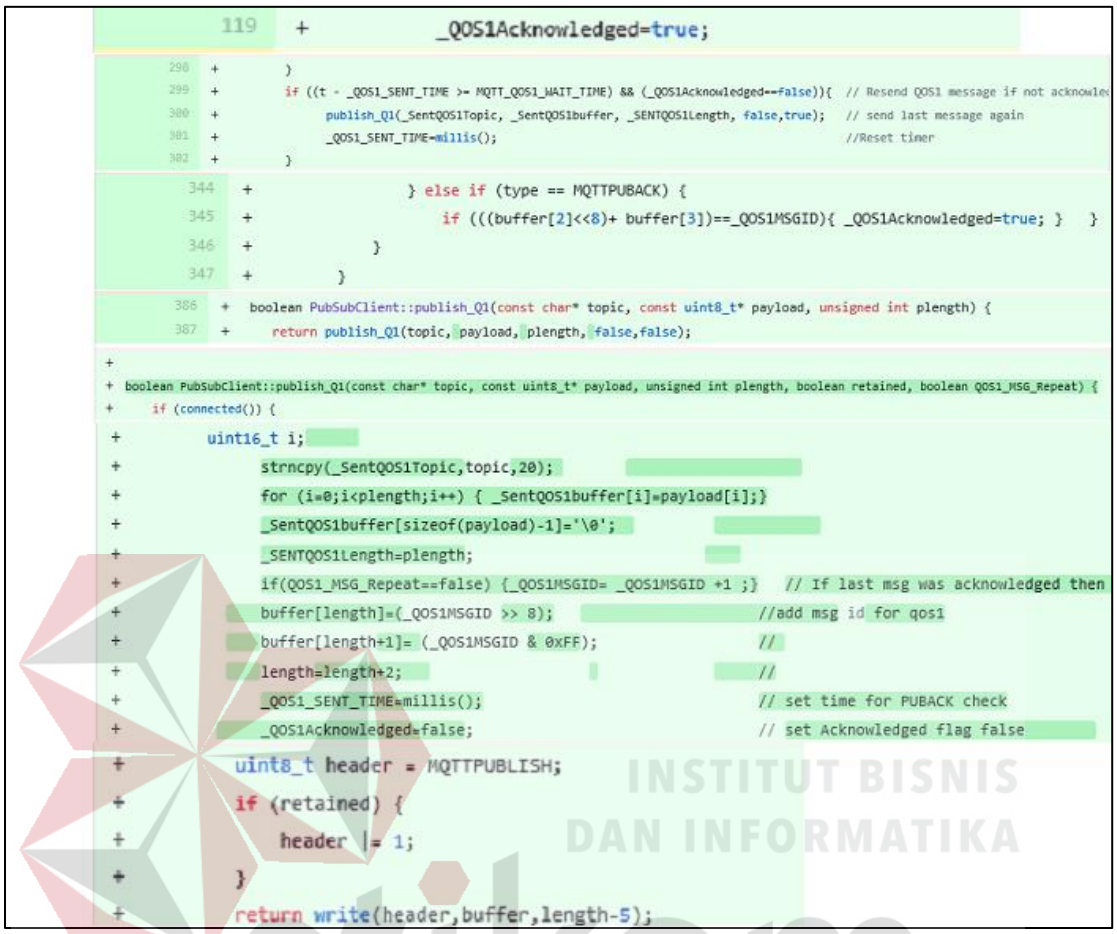
Penjelasan dari masing-masing komponen diatas adalah

1. *Finger Clip Heart Rate* Sensor : Sensor untuk deteksi *heart rate* titik uji coba pengambilan *heart rate* dilakukan di ujung jari (jari telunjuk), dimana kedua sensor tersebut telah diuji dengan kalibrasi menggunakan alat dari pabrik yang terbukti keakuratannya yaitu Oxymeter.
2. Mikrokontroller : *device* yang digunakan Arduino Pro Mini yang nantinya disesuaikan dengan desain *casing* yang digunakan untuk data pengambilan aktifitas lari.

3. Media Trasmisi : Media Transmisi yang digunakan adalah Modul ESP8266 yang digunakan sebagai media transmisi untuk proses pengiriman data *heart rate* ke sisi *broker*. Pada modul ESP8266 terdapat *library* yang bernama *pubsubclient* dan mempunyai keterbatasan *publish message* dengan QoS *level* 0. Untuk QoS *level* 1 diperlukan tambahan *function* dan *header* pada *library*. Pada *header* terdapat tambahan seperti Gambar berikut:

Gambar 3.11 Tambahan *Header Pubsubclient*

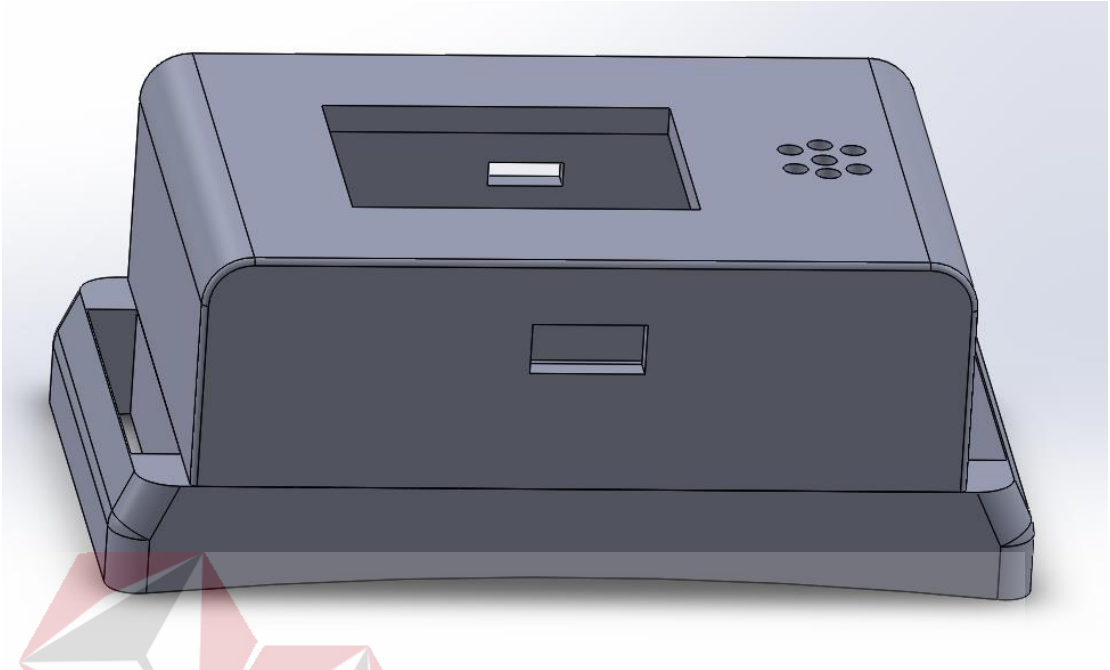
Untuk *function* pada *library* diperlukan tambahan sebagai berikut :



```
119 +            _QOS1Acknowledged=true;

298 +        }
299 +        if ((t - _QOS1_SENT_TIME >= MQTT_QOS1_WAIT_TIME) && (_QOS1Acknowledged==false)){  // Resend QOS1 message if not acknowle
300 +            publish_Q1(_SentQOS1Topic, _SentQOS1buffer, _SENTQOS1Length, false,true);   // send last message again
301 +            _QOS1_SENT_TIME=millis();                                                   //Reset timer
302 +        }

344 +                } else if (type == MQTTPUBACK) {
345 +                    if (((buffer[2]<<8)+ buffer[3])==_QOS1MSGID){ _QOS1Acknowledged=true; }   }
346 +            }
347 +        }

386 +  boolean PubSubClient::publish_Q1(const char* topic, const uint8_t* payload, unsigned int plength) {
387 +    return publish_Q1(topic, payload, plength, false,false);

+
+ boolean PubSubClient::publish_Q1(const char* topic, const uint8_t* payload, unsigned int plength, boolean retained, boolean QOS1_MSG_Repeat) {
+     if (connected()) {
+         uint16_t i;
+             strncpy(_SentQOS1Topic,topic,20);
+             for (i=0;i<plength;i++) { _SentQOS1buffer[i]=payload[i];}
+             _SentQOS1buffer[sizeof(payload)-1]='\0';
+             _SENTQOS1Length=plength;
+             if(QOS1_MSG_Repeat==false) {_QOS1MSGID= _QOS1MSGID +1 ;}   // If last msg was acknowledged then
+             buffer[length]=(_QOS1MSGID >> 8);                          //add msg id for qos1
+             buffer[length+1]= (_QOS1MSGID & 0xFF);                     //
+             length=length+2;                                           //
+             _QOS1_SENT_TIME=millis();                                  // set time for PUBACK check
+             _QOS1Acknowledged=false;                                   // set Acknowledged flag false
+         uint8_t header = MQTTPUBLISH;
+         if (retained) {
+           header |= 1;
+         }
+         return write(header,buffer,length-5);
```

Gambar 3.12 Tambahan *Function Library Pubsubclient*

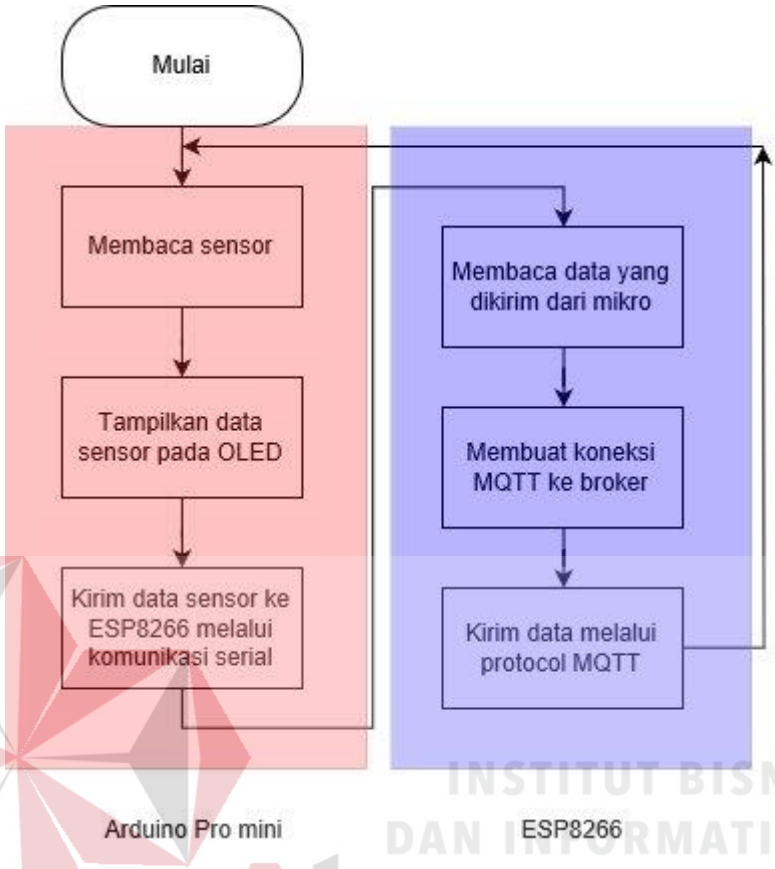
4. OLED : digunakan untuk menampilkan data sensor.
5. *Casing* : desain ini digunakan untuk *packaging* dari rangkaian yang digunakan untuk deteksi *heart rate*, dengan harapan bahwa alat ini bisa digunakan untuk aktfitas olahraga. Desainnya ditunjukkan pada Gambar 3.13 dengan ukuran panjang 7.4 cm, lebar 4.2 cm, tinggi 2.8 cm sebagai berikut :



Gambar 3.13 Desain *Casing* Perangkat Keras

A. *Flowchart* Pemrograman Arduino Pro Mini dan ESP8266

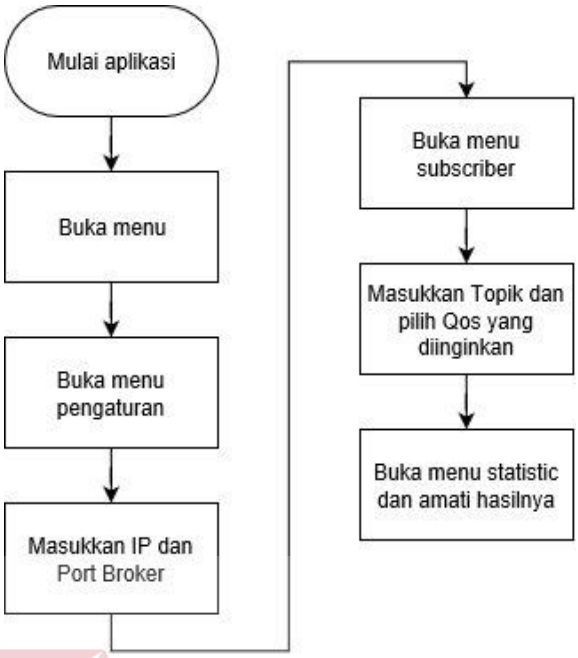
Pemrograman Arduino mengirimkan data yang terhubung dengan modul ESP8266, ditunjukkan pada Gambar 3.14 di bawah ini :



Gambar 3.14 *Flowchart* Pemrograman Arduino Pro Mini Dan ESP8266

B. *Flowchart* Pemrograman Pada Android

Dalam proses pembuatan aplikasi Android sebagai *subscriber* bisa dilakukan melalui langkah-langkah dari proses yang ditunjukkan pada Gambar 3.15 di bawah ini :

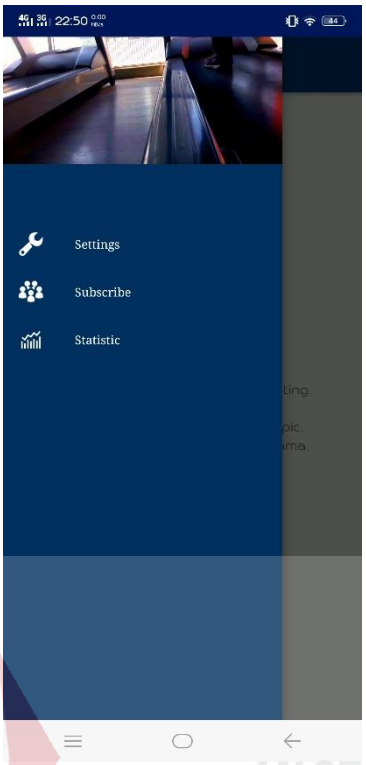


Gambar 3.15 *Flowchart* Aplikasi Android

1) Jalankan aplikasi MHR pada *smartphone* yang bertujuan untuk mengetahui data *heart rate* seperti Gambar 3.16 di bawah ini :

Gambar 3.16 Halaman Awal

2) Buka menu aplikasi HRM di pojok kiri atas. Pada menu tersebut (Gambar 3.17) terdapat menu *settings*, menu *subscribe*, menu *statistic*.



Gambar 3.17 Menu Awal

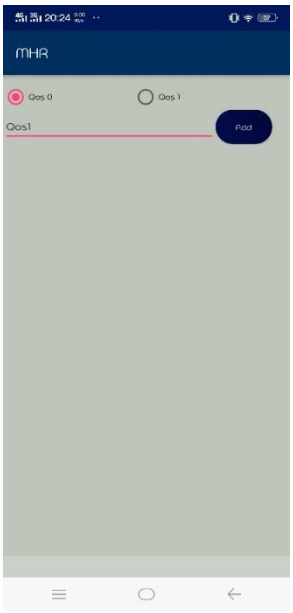
3) Buka menu *settings* (Gambar 3.18) untuk mengatur IP dari *server*/*broker*, *port* dari *server*/*broker*, *username* (opsional) dan *password* (opsional).

Gambar 3.18 Menu *Settings*

4) Jika tidak terjadi kesalahan pada penulisan pengaturan yang ada pada menu *settings* maka akan muncul notifikasi seperti Gambar 3.19 di bawah ini.

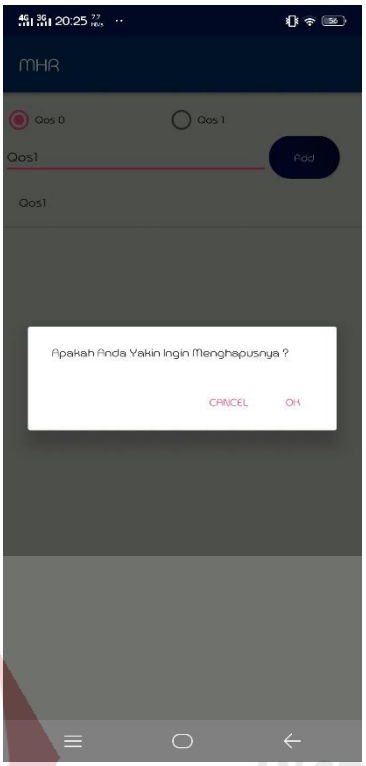
Gambar 3.19 Notifikasi Jika Sudah Tersambung

5) Kembali ke menu awal kemudian masuk pada menu *subscribe* (Gambar 3.20) untuk mengisi nama topik yang dituju dan memilih Qos yang akan dipilih. Kemudian tekan tombol *Add*. Jika sudah ter-*subscriber* maka akan muncul nama topik pada menu *list* (Gambar 3.21). Jika *user* ingin menghapus nama topik *subscriber* perlu menekan nama dari topik tersebut sehingga muncul notifikasi (Gambar 3.22).



Gambar 3.20 Menu *Subscriber*

Gambar 3.21 *List* Yang Terisi

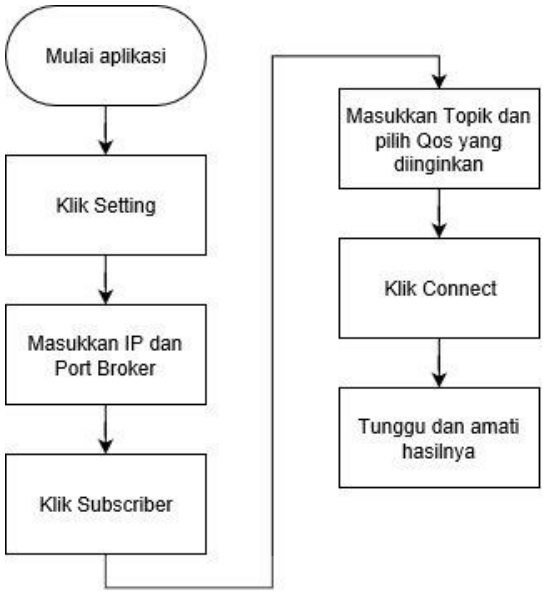


Gambar 3.22 Notifikasi Untuk Menghapus Nama Topik *Subscriber*

6) Pada menu awal tekan menu *statistic* sehingga pengguna akan disungguhi grafik (Gambar 3.4) untuk bisa mengamati data *heart rate*.

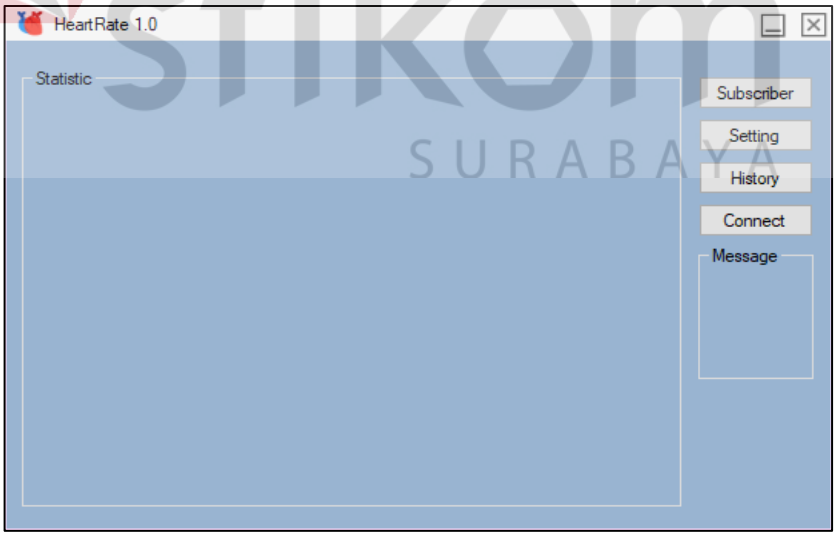
C. *Flowchart* Pemrograman Pada Visual Studio

Pada proses pembuatan aplikasi *dekstop*, sebagai *subscriber* bisa dilakukan melalui langkah-langkah (Gambar 3.23) di bawah ini :



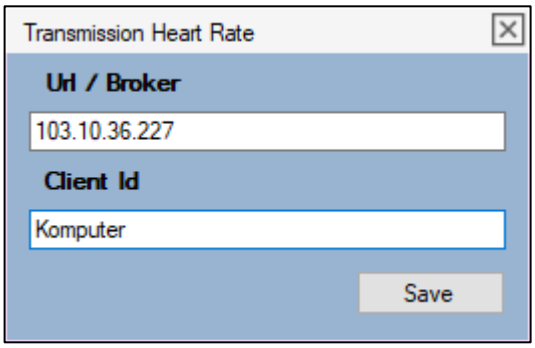
Gambar 3.23 *Flowchart* Aplikasi *Desktop*

1) Buka aplikasi *HeartRate* 1.0 kemudian akan muncul halaman awal aplikasi (Gambar 3.24) seperti di bawah ini :



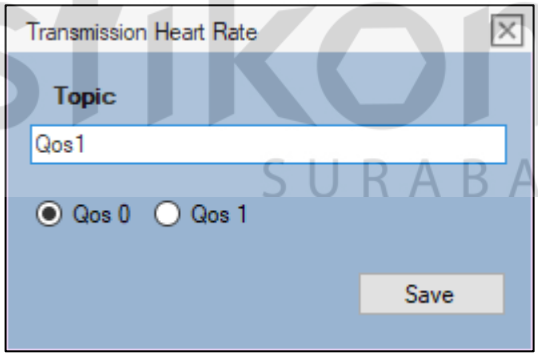
Gambar 3.24 Menu Utama *Desktop*

2) Klik menu *setting* (Gambar 3.25) untuk mengisi IP dari *broker*/*server* dan *client* id dari *subscriber* tersebut untuk mengetahui id yang dibuat. Kemudian Klik Save.



Gambar 3.25 Menu Setting pada Aplikasi *Desktop*

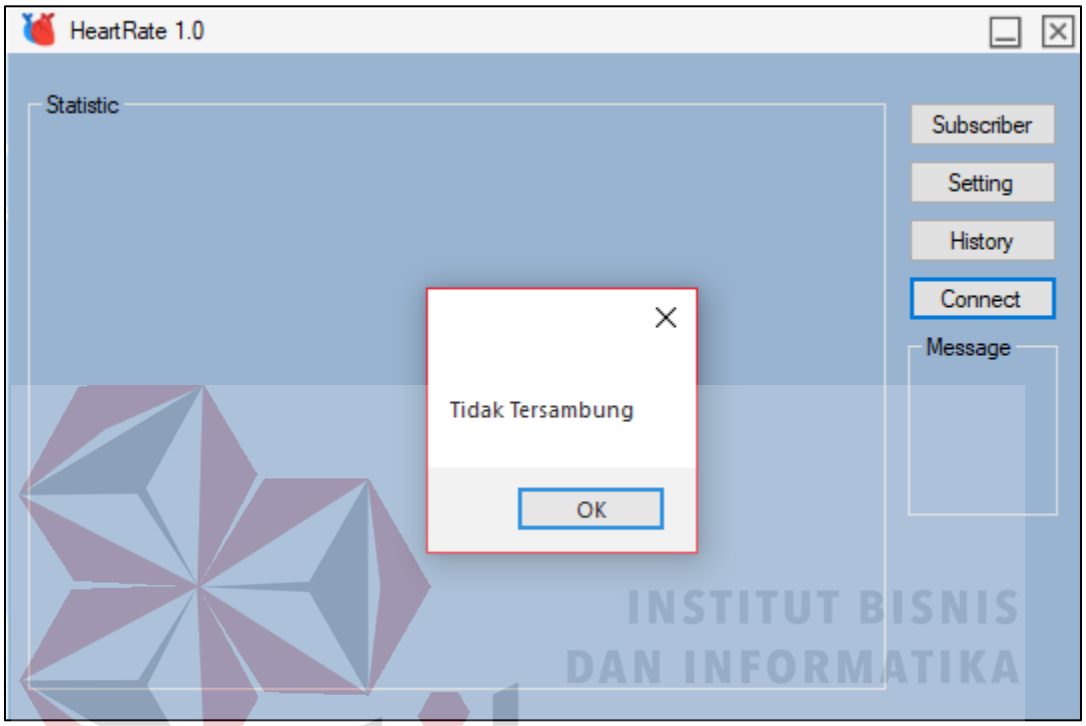
3) Pada menu utama klik *subscriber* (Gambar 3.26) untuk mengisi topik yang dituju bertujuan untuk menerima pesan dari *publisher* dengan topik yang sama. Kemudian pilih Qos 0 atau Qos 1. Setelah itu klik *button save.*



Gambar 3.26 Menu *Subscriber* pada *Desktop*

4) Sesudah mengatur semuanya, kemudian klik *connect* untuk membuka koneksi pada *broker/server*. Jika aplikasi tidak tersambung maka akan muncul notifikasi seperti

Gambar 3.27 di bawah ini dan jika tersambung akan seperti Gambar 3.5. Kemudian amati data yang masuk pada *subscriber*.



Gambar 3.27 Notifikasi Tidak Terhubung ke *Broker*

3.5 *Quality Of Service*

Quality of Service (QoS) didefinisikan sebagai suatu pengukuran tentang seberapa baik jaringan dan merupakan suatu usaha untuk mendefinisikan karakteristik dan sifat dari suatu layanan. Berikut ini merupakan Parameter QoS :

1) *Delay*

Delay atau *Latency* adalah waktu tunda yang dibutuhkan dalam proses transmisi data. Misalkan paket data yang berasal dari terminal A akan dikirimkan menuju ke terminal B, di dalam perjalanannya data tersebut mengalami propagasi menuju

ke terminal B sehingga membutuhkan waktu tertentu untuk sampai ke terminal B. selisih waktu antara paket diterima dengan waktu paket dikirim disebut sebagai *delay* atau *latency*. Kategori latensi besar *delay* ditunjukkan pada Tabel 3.1 dan mempunyai rumus sebagai berikut :

$$Delay = Tr - Ts \ldots.(3.1)$$

Yang mana :

Tr = Waktu penerimaan paket (detik)

Ts = Waktu pengiriman paket (detik)

Tabel 3.1 Kategori Latensi Besar *Delay* (ETSI, 2002)

Kategori Latensi	Besar *Delay*
Sangat bagus	< 150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Buruk	> 450 ms

2) *Throughput*

kecepatan (*rate*) transfer data efektif, yang diukur dalam bps. *Throughput* adalah jumlah total kedatangan paket yang sukses yang diamati pada *destination* selama *interval* waktu tertentu dibagi oleh durasi *interval* waktu tersebut. *Throughput* dapat dihitung dengan rumus :

$$Throughput = \frac{Paket\ data\ yang\ diterima}{Lama\ pengamatan} \ldots.(3.2)$$

3) *Packet Loss*

Packet loss adalah jumlah paket data yang hilang per detik. *Packet loss* dapat disebabkan oleh sejumlah faktor, mencakup penurunan *signal* dalam media jaringan, melebihi batas saturasi jaringan, paket yang *corrupt* yang menolak

untuk transit, dan kesalahan perangkat keras jaringan. Degradasi *packet loss* ditunjukkan pada Tabel 3.2. *Packet loss* dapat dirumuskan sebagai :

$$Packet\ Loss = (Pd / Ps) \times 100\% \quad ....(3.3)$$

Yang mana :

Pd = Jumlah paket yang mengalami *drop*/gagal (paket)

Ps = Jumlah paket yang dikirim (paket)

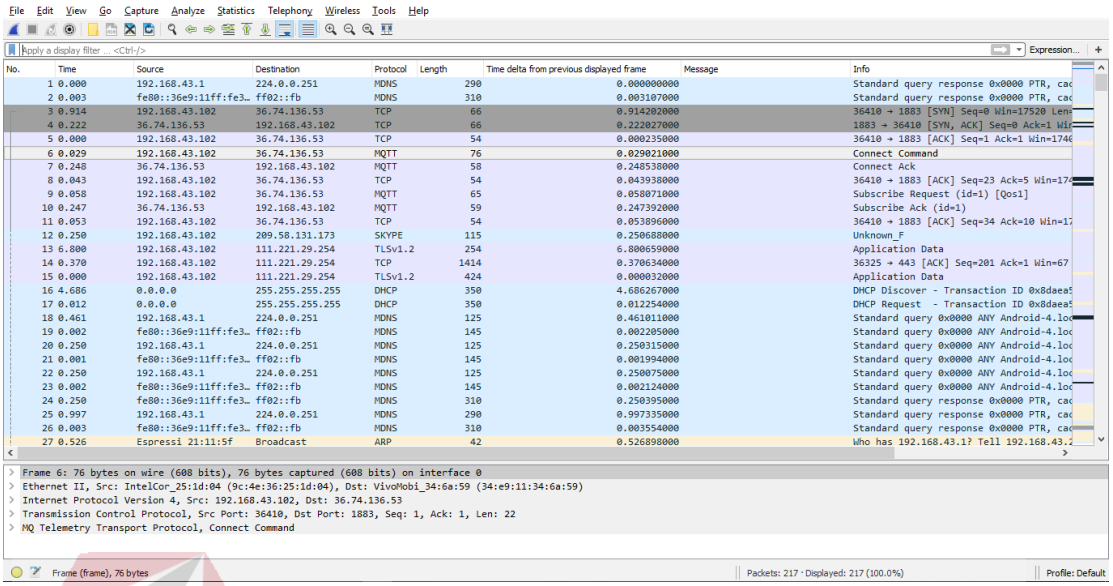
Tabel 3.2 Kategori Degradasi *Packet Loss* (ETSI, 2002)

Kategori Degradasi	Packet Loss
Sangat bagus	0 %
Bagus	3 %
Jelek	15 %
Sangat jelek	25 %

3.6 Pengambilan Data Pada *Wireshark*

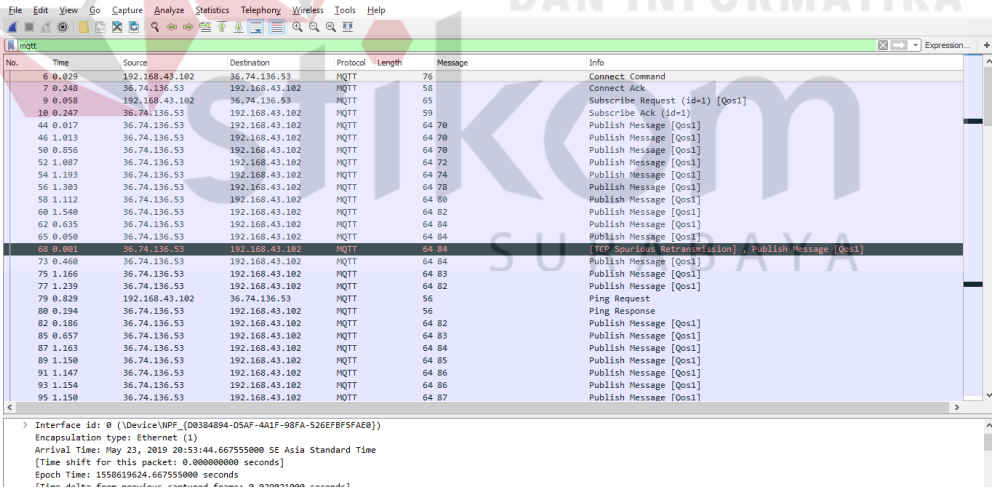
Dalam proses pengambilan data pada *Wireshark* dilakukan dengan cara sebagai berikut :

1) Membuka *wireshark* yang telah meng-*crop* semua jaringan yang tertangkap (Gambar 3.28).



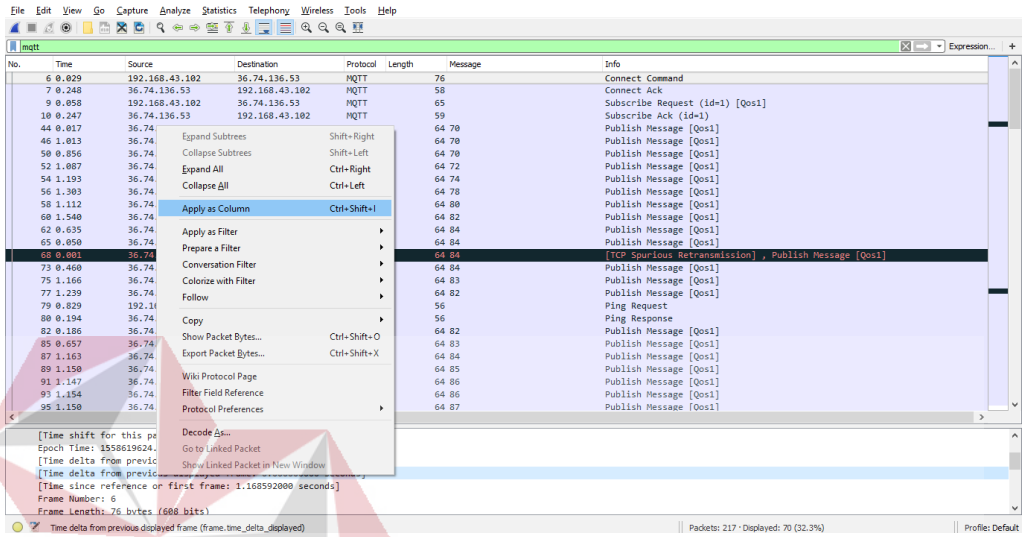
Gambar 3.28 Hasil *Capture* Pada *Wireshark*

2) Lakukan filtering jaringan protokol MQTT pada *wireshark* seperti Gambar 3.29.

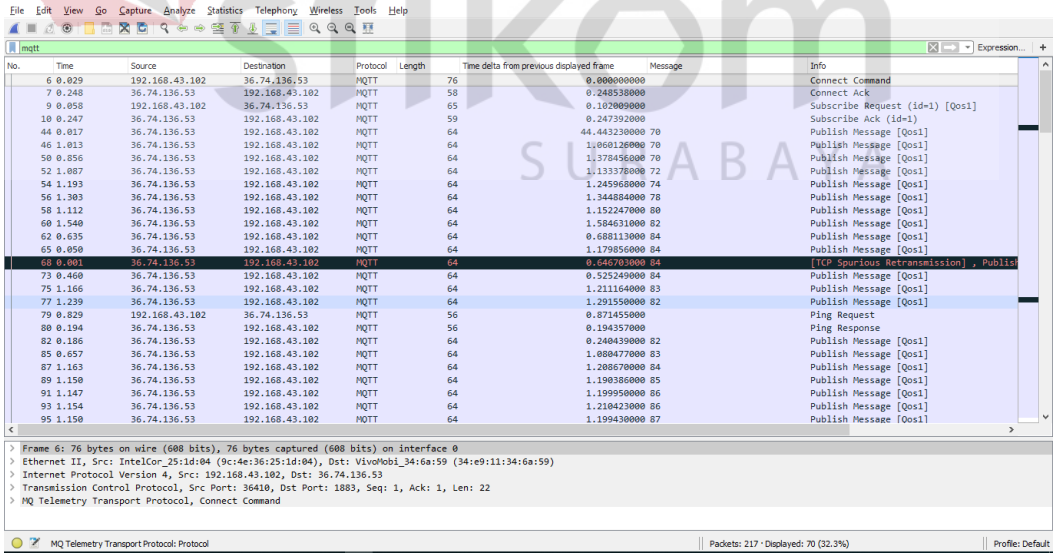


Gambar 3.29 *Filtering* Protokol MQTT

3) Klik tombol panah pada *Frame* kemudian pilih *Time delta from previous displayed frame*, lalu klik kanan dan pilih *Apply as Column* (Gambar 3.30) untuk menampilkan kolom baru (Gambar 3.31).



Gambar 3.30 Membuka *Collapse Group Frames* Pada *Wireshark*



Gambar 3.31 Kolom *Time Delta From Previous Displayed Frame*

3.7 Transmisi Qos 0 Dan Qos 1 Pada *Wireshark* Menggunakan Koneksi MQTT

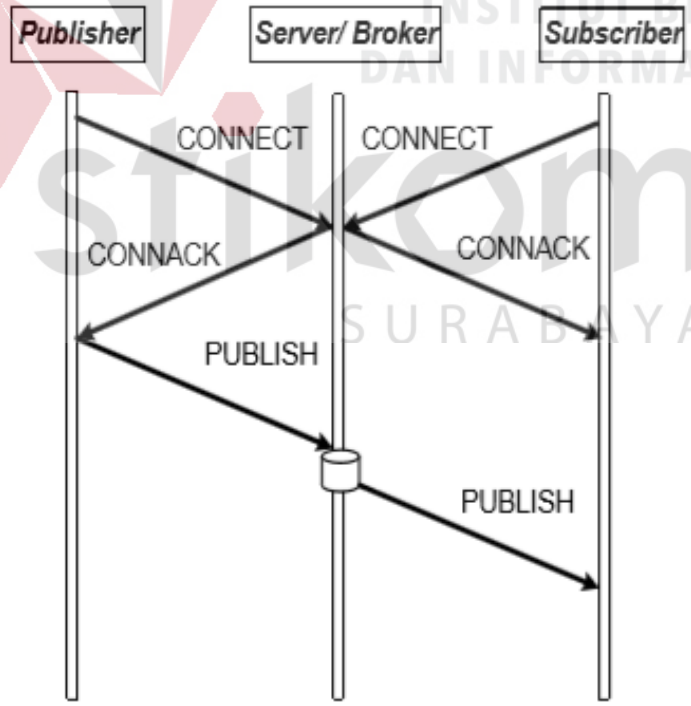
Dalam melakukan pengiriman data *heart rate* melalui protokol MQTT, protokol MQTT memiliki 3 *level* QoS. Tetapi penulis menggunakan QoS *level* 0 dan QoS *level* 1 dengan keterbatasan pada *hardware*. MQTT memiliki tipe pesan seperti Tabel 3.3 sebagai berikut :

Tabel 3.3 Tipe Pesan MQTT

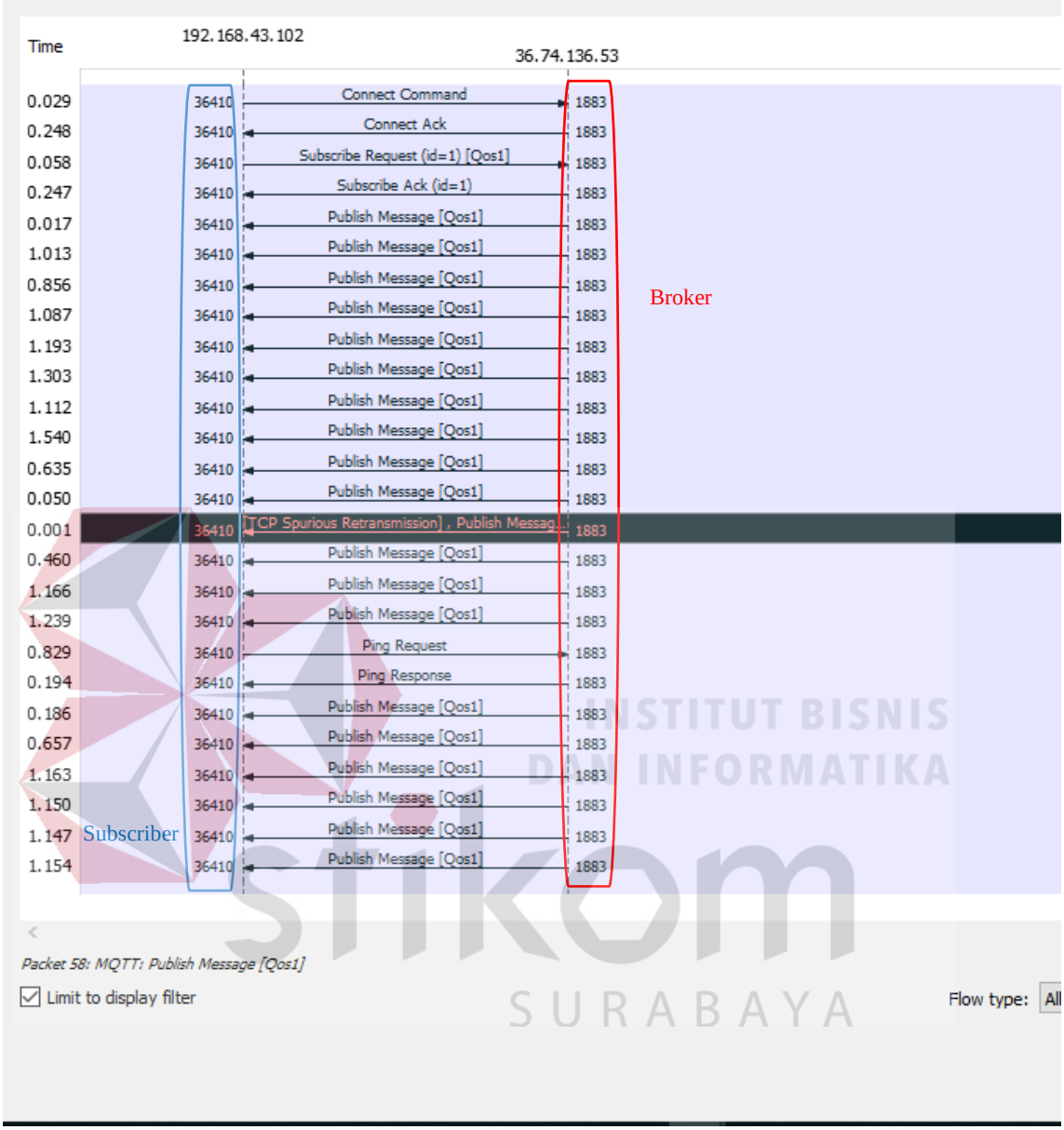
Mnemonic	No.	Description
CONNECT	1	*Client Request to Connect to Server*
CONNACK	2	*Connect Acknowledgment*
PUBLISH	3	*Publish Message*
PUBACK	4	*Publish Acknowledgment*
PUBREC	5	*Publish Received-Assured Delivery Part 1*
PUBREL	6	*Publish Release-Assured Delivery Part 2*
PUBCOMP	7	*Publish Complete-Assured Delivery Part 3*
SUBSCRIBE	8	*Client Subscribe Request*
SUBACK	9	*Subscribe Acknowledgment*
UNSUBSCRIBE	10	*Client Unsubscribe Request*
UNSUBACK	11	*Unsubscribe Acknowledgment*
PINGREQ	12	*PING Request*
PINGRESP	13	*PING Response*
DISCONNECT	14	*Client is Diconnecting*

1. **QoS *level* 0**

Pesan dikirim dengan menggunakan jaringan TCP/IP. Respon atau jawaban dari *client* tidak diminta dan tidak akan ada pengiriman ulang pesan. Pesan akan diterima hanya sekali atau tidak sama sekali. Seperti contoh pada Gambar 3.32 *publisher* mengirim tipe pesan *CONNECT* bersamaan dengan *subscriber*, tetapi dari sisi *subscriber* terdapat sebuah topik dalam tipe pesan tersebut. *Broker* akan menerima pesan tersebut dan akan mengirimkan *acknowledge* yang bernama *CONNACK* kepada *publisher* dan *subscriber*. Dari sisi *publisher* akan mengirimkan sebuah pesan *(PUBLISH)* berupa data dan topik yang akan diterima oleh sisi *broker*. *Broker* akan meneruskan sebuah pesan *(PUBLISH)* sesuai topik yang sama pada *subscriber*.



Gambar 3.32 *Publish Message* dengan QoS *Level* 0

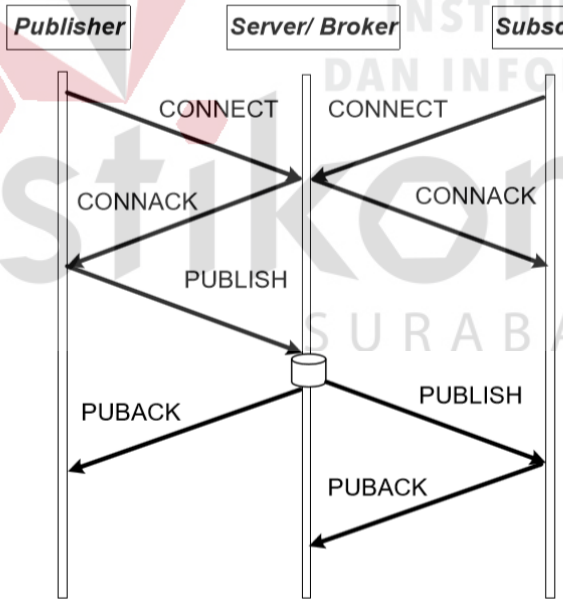


Gambar 3.33 Hasil *Flow Graph* MQTT QoS *Level* 0 pada *Wireshark*

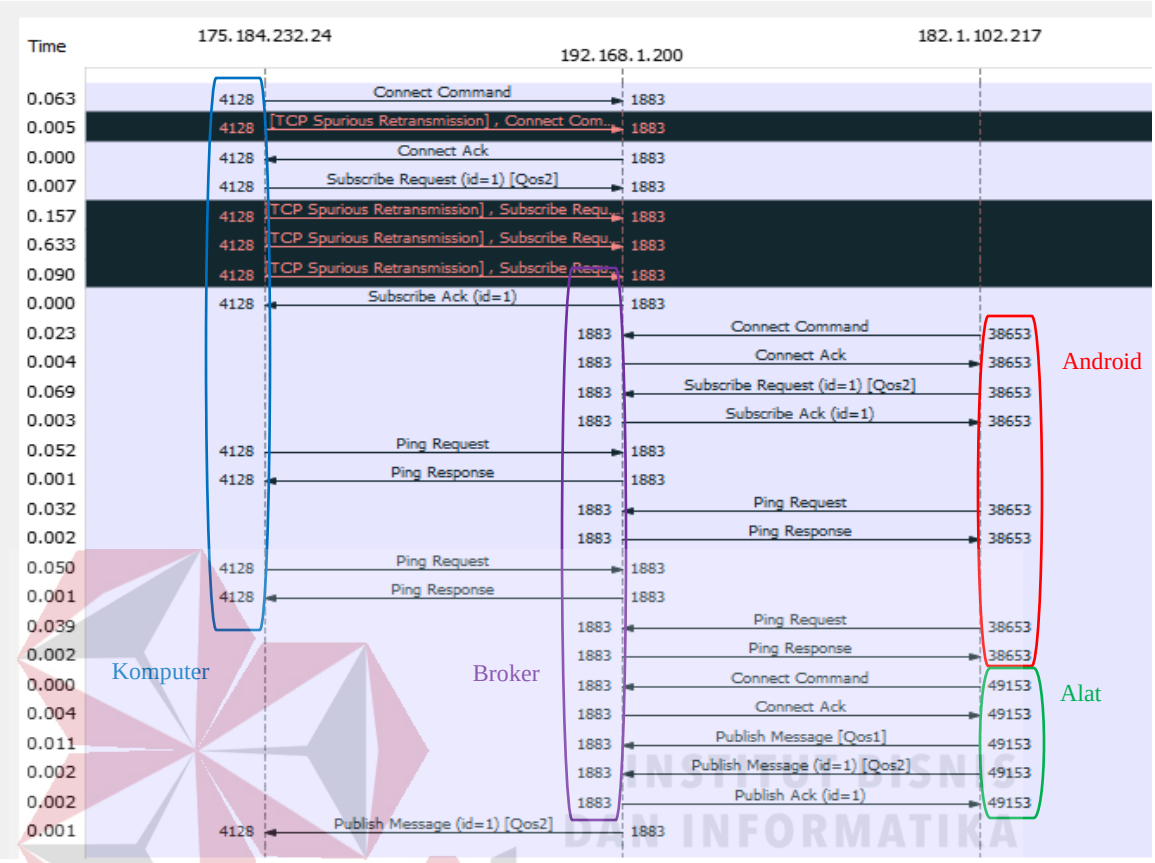
2. QoS *level* 1

pesan akan dikirim oleh *publisher* jika terjadi kegagalan pengiriman pesan atau pesan konfirmasi tidak diterima, *publisher* akan mengirim pesan dengan menyertakan bit DUP di bagian kepala pesan dan menerima pesan *acknowledge* dari pesan yang dikirim. Seperti contoh pada Gambar 3.34 *publisher* mengirim tipe pesan *CONNECT*

bersamaan dengan *subscriber*, tetapi dari sisi *subscriber* terdapat sebuah topik dalam tipe pesan tersebut. *Broker* akan menerima pesan tersebut dan akan mengirimkan *acknowledge* yang bernama *CONNACK* kepada *publisher* dan *subscriber*. Dari sisi *publisher* akan mengirimkan sebuah pesan *(PUBLISH)* berupa data dan topik yang akan diterima oleh sisi *broker*. Dari sisi *broker* akan mengirimkan tipe pesan *PUBACK* dimana pesan dari *publisher* sudah diterima dan akan mengirim sebuah *acknowledge* kepada *publisher*. Dari sisi *broker* yang lainnya akan meneruskan sebuah pesan *(PUBLISH)* sesuai topik yang sama pada *subscriber*. *Subscriber* akan mengirimkan tipe pesan *PUBACK* dimana pesan dari *broker* sudah diterima dan akan mengirim sebuah *acknowledge* kepada *broker*.



Gambar 3.34 *Publish Message* dengan QoS *Level* 1

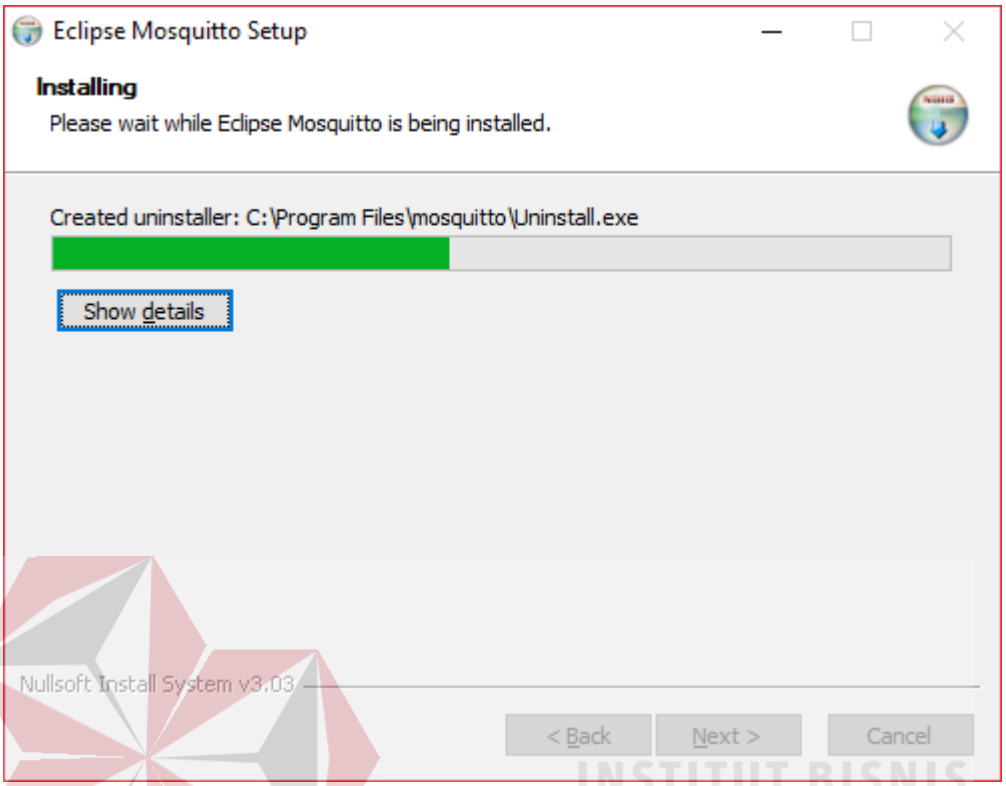


Gambar 3.35 Hasil *Flow Graph* MQTT QoS *Level* 1 pada *Wireshark*

3.8 *Install Mosquitto Broker*

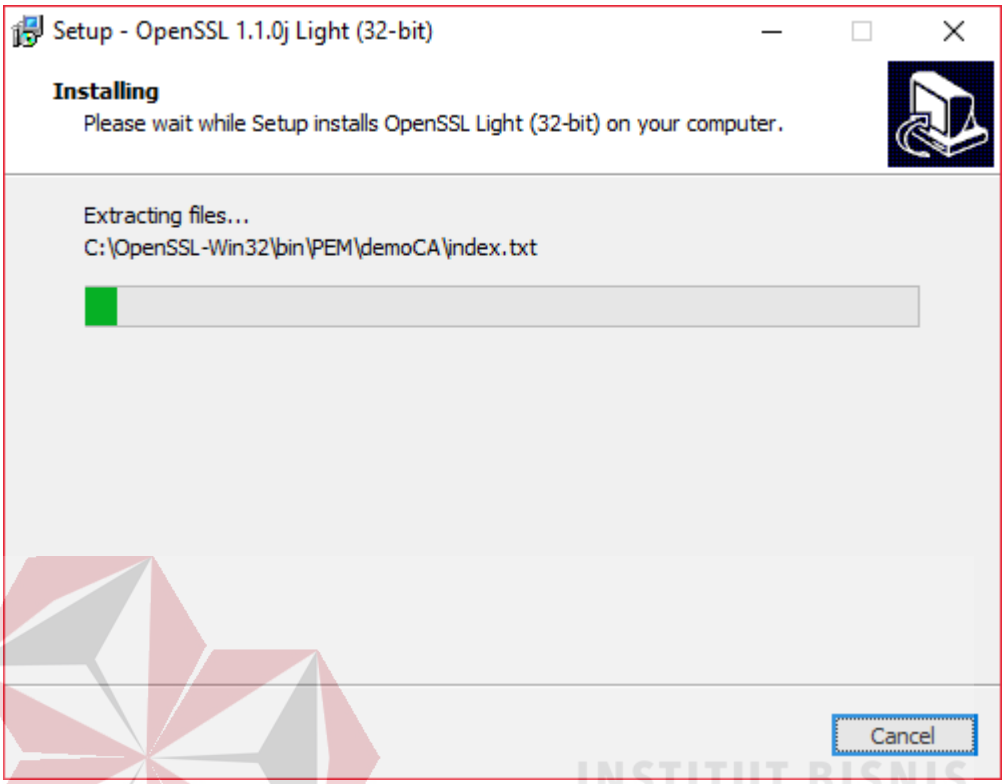
1) *Download mosquitto* 32 bit yang ada pada url https://mosquitto.org/download/

2) *Install mosquitto broker* pada *windows*.



Gambar 3.36 *Install Mosquitto Broker*

3) Setelah selesai *install mosquito broker*, kemudian *download* dan *install* openssl yang ada pada url http://slproweb.com/products/Win32OpenSSL.html.



Gambar 3.37 *Install OpenSSL*

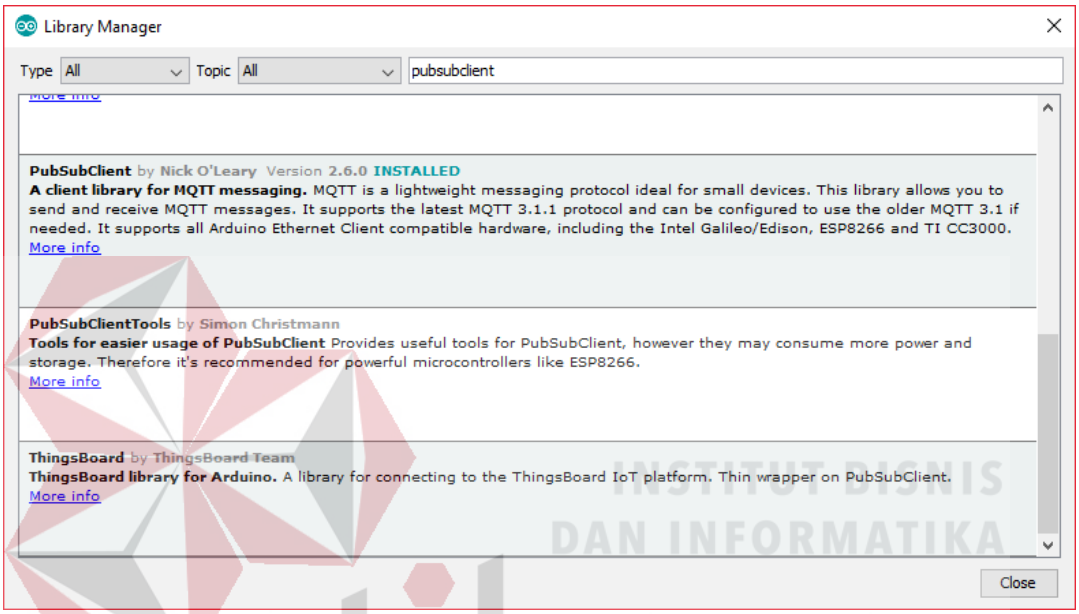
4) Copy folder bin yang ada pada C:\openssl\bin\ kedalam *directory mosquitto* C:\Program Files (x86)\mosquitto.
5) Copy file libeay32.dll dan ssleay32.dll yang ada pada directory C:\openssl\ kedalam *directory mosquitto* C:\Program Files (x86)\mosquitto.
6) Download file pthreadVC2.dll yang ada pada url ftp://sources.redhat.com/pub/pthreads-win32/dll-latest/dll/x86/.
7) Copy file pthreadVC2.dll yang sudah di download kemudian masukkan kedalam *directory mosquitto* C:\Program Files (x86)\mosquitto.

3.9 *Install Mosquitto Client*

1) Buka aplikasi Arduino.

2) Klik *Sketch*. Kemudian Manage Libraries yang ada pada include Library.

3) *Download library pubsubclient*.



Gambar 3.38 *Install PubSubClient*

BAB IV

HASIL DAN PEMBAHASAN

Hasil dari penelitian ini merupakan hasil dari pengamatan dan pengujian dari prototipe dan perangkat elektronik yang telah dirancang oleh penulis.

4.1 Pengujian Arduino Pro Mini

4.1.1 Tujuan

Pengujian Arduino pro mini bertujuan untuk mengetahui apakah Arduino pro mini dapat berjalan dengan baik, serta dapat mengeksekusi program berjalan dengan benar.

4.1.2 Alat yang digunakan

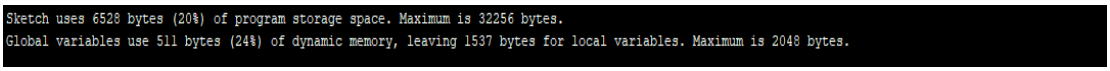
1. Arduino Pro Mini
2. USB TTL
3. PC atau Laptop
4. Software Arduino

4.1.3 Prosedur Pengujian

1. Menghidupkan PC atau laptop.
2. Sambungkan Arduino Pro mini dengan kabel USB TTL.
3. Jalankan program Arduino.
4. Sebelum *upload* program yang telah dibuat, pilih *verify* untuk memastikan tidak ada error, apabila tidak ada pilih *upload*.

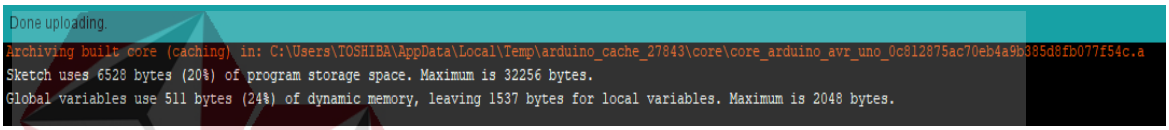
4.1.4 Hasil Pengujian

Pada Gambar 4.1 yaitu melakukan *Verify* program menggunakan *software* Arduino



Gambar 4.1 *Verify* Program

Setelah *verify* program selesai, maka langkah selanjutnya yaitu meng-*upload* program ke Arduino dengan membuka *software* Arduino seperti pada Gambar 4.2.



Gambar 4.2 *Upload* Program

4.2 Hasil Pengujian Sensor *Heart Rate*

4.2.1 Tujuan

Pada pengujian sensor *heart rate* dibuat program untuk dapat membaca detak jantung menggunakan *software* Arduino dengan jalur komunikasi I2C yang telah disediakan pada Arduino Pro Mini.

4.2.2 Alat Yang Digunakan

1. PC atau *laptop*
2. Rangkaian Arduino Pro Mini
3. OLED 128x64
4. Sensor *Finger Clip Heart Rate*
5. USB TTL

4.2.3 Prosedur Pengujian

1. Hubungkan sensor *heart rate* dan OLED 128x64 pada Arduino Pro Mini.
2. Sambungkan USB TTL.
3. *Upload* program untuk membaca sensor *heart rate* pada Arduino Pro Mini.
4. Amati data dari Sensor di OLED.

4.2.4 Hasil Pengujian

Pada pengujian sensor *heart rate* yaitu melakukan pembacaan detak jantung dan ditampilkan pada OLED. Satuan dari sensor *heart rate* yaitu Bpm (*beat per minute)*. Pengujian sensor *heart rate* menggunakan Oxymeter sehingga selisih perbandingan dapat terlihat berdasarkan HR *max* dan THR (*Target Heart Rate)*. Adapaun hasil percobaan sensor *heart rate* pada Tabel 4.1.

Gambar 4.3 Pengujian Sensor *Heart Rate*

Tabel 4.4 Selisih Perbandingan Sensor *Heart Rate* dan Oxymeter

No	Sampel	Nilai HR Max (bpm)	THR (bpm)		Error (%)
			Finger Clip	Oxymeter	
1	A	192, 16	66, 612	66, 612	0
2	B	192, 16	77, 112	77, 812	0,9
3	C	193,5	75, 95	76, 65	0,9
4	D	178,09	70, 763	70, 763	0
5	E	182,24	77, 868	78, 568	0,9
Rata-Rata Kesalahan (%)					0,5

Kesimpulan daripada hasil percobaan sensor *heart rate* dan Oxymeter dapat dilihat pada tabel di atas yaitu akurasi untuk sensor *finger clip* lebih baik dibandingkan sensor Oxymeter, sehingga perbedaan antara nilai sensor sebesar 0,5%.

4.3 Pengujian Koneksi *Broker* Menggunakan ESP8266

4.3.1 Tujuan

Pengujian ESP8266 yang bertujuan untuk melakukan transmisi data dengan protokol MQTT, serta program dapat berjalan dengan baik dan benar.

4.3.2 Alat yang digunakan

1. PC atau laptop
2. Modul ESP8266
3. USB TTL

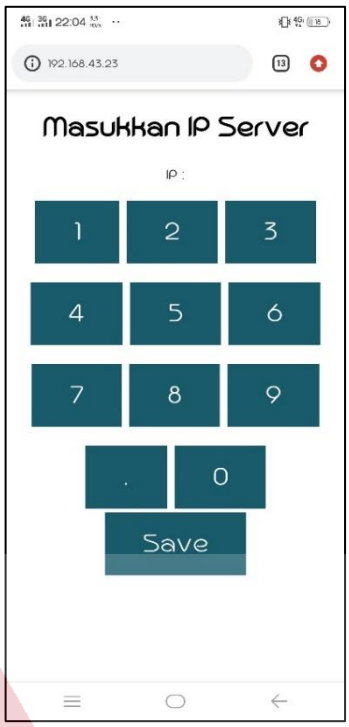
4.3.3 Prosedur Pengujian

1. Menghidupkan PC atau laptop.
2. Sambungkan Modul ESP8266 dengan kabel USB TTL.

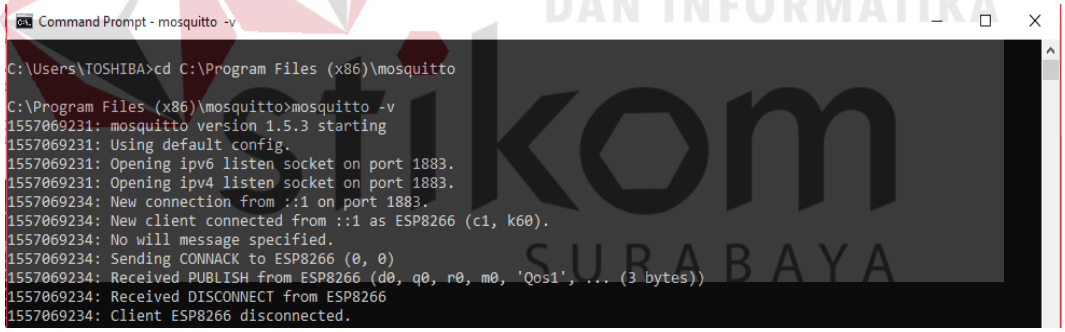
3. Jalankan program Arduino.
4. Sebelum *upload* program yang telah dibuat, pilih verify untuk memastikan tidak ada error, apabila tidak ada pilih *upload*.
5. Buat *hotspot* Wifi menggunakan *smartphone* dengan nama smart dan password smartHRM.
6. Nyalakan *hardware* dan jangan menyentuh sensor pada *hardware*. Tunggu sampai terhubung ke *hotspot.* Buka *browser* yang ada pada *smartphone* dan isikan url dari IP modul ESP8266 yang didapat dari *hotspot* wifi yang dibuat.
7. Masukkan IP *broker*.
8. Sentuh sensor *heart rate* dan amati nilai sensor yang masuk ke *broker*.

4.3.4 Hasil Pengujian

Pengujian koneksi broker dilakukan dengan membuka url IP (Gambar 4.4) yang didapat dari *hotspot wifi* untuk membuat koneksi antara perangkat keras dengan *broker*. Nilai sensor yang masuk bisa dilihat pada *broker* (Gambar 4.5).



Gambar 4.4 *Setting* IP *Broker*



Gambar 4.5 *Broker*

4.4 Pengujian Serial Komunikasi

4.4.1 Tujuan

Pada pengujian serial komunikasi terdapat dua modul yaitu Arduino pro mini dan ESP8266. Kedua modul tersebut akan melakukan komunikasi, sehingga nilai

sensor yang tersambung pada Arduino pro mini akan dikirim ke modul ESP8266 melalui komunikasi serial.

4.4.1 Alat yang digunakan

1. Arduino Pro Mini
2. ESP8266
3. Sensor Finger Clip *Heart Rate*
4. PC atau Laptop

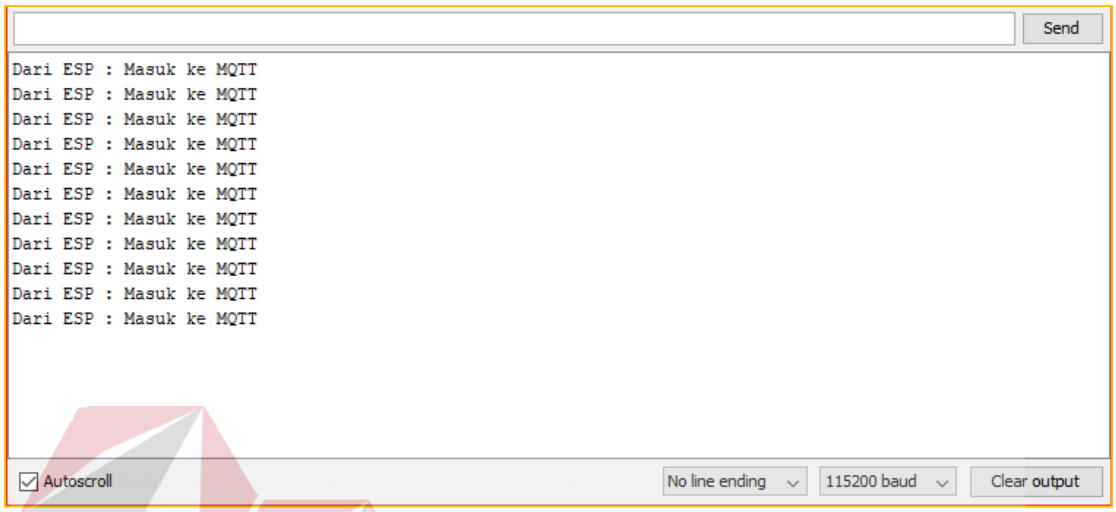
4.4.2 Prosedur Pengujian

1. Menghidupkan PC atau Laptop
2. Hubungkan pin 2 Arduino Pro Mini (*Receiver*) dengan pin 0 ESP8266 (*Transmitter*).
3. Hubungkan pin 3 Arduino Pro Mini (*Transmiter*) dengan pin 2 ESP8266 (*Receiver*).
4. Amati pada *broker*.

4.4.3 Hasil Pengujian

Pengujian serial komunikasi dilakukan dengan cara melihat pada *serial monitor* yang ada pada Arduino dengan Port COM dari Arduino Pro Mini. Ketika membaca sensor, arduino akan mengirim data tersebut melalui pin 3 sebagai *transmitter* yang dihubungkan dengan pin 2 ESP8266 sebagai *receiver*. Setelah membaca data dari Arduino Pro Mini maka data tersebut akan dikirim pada *broker* dan diterima dengan baik, ESP8266 akan mengirim timbal balik kepada Arduino bahwa

data sudah terkirim melalui pin 0 ESP8266 sebagai *Transmitter* yang dihubungkan dengan pin 2 Arduino Pro Mini sebagai *receiver*.



Gambar 4.6 *Serial Monitor* pada Arduino Pro Mini

4.5 Pengujian Keseluruhan Sistem

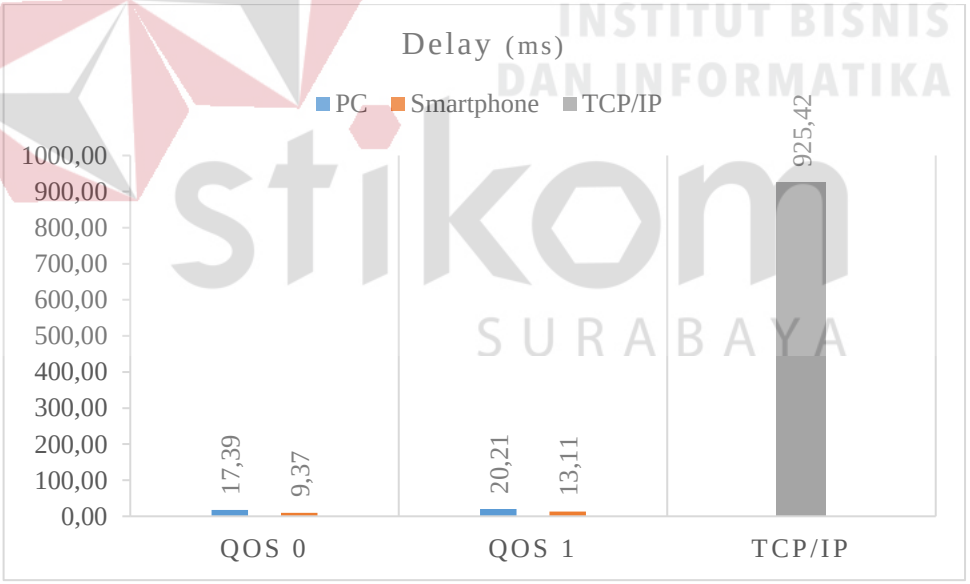
Hasil pengujian keseluruhan sistem meliputi pengujian dari segi *Software* maupun *Hardware*, *Software* program berupa aplikasi Android dan *desktop* dimana program tersebut akan membaca nilai sensor yang dikirim dari perangkat keras. Untuk *Hardware* berupa pemasangan setiap modul antara Arduino pro mini dan ESP8266. Setiap sampel akan diuji sebanyak 9 kali sebagai pengujian transmisi data *heart rate* yang menggunakan protokol TCP/IP sebanyak 3 kali dan protokol MQTT sebanyak 6x termasuk QoS 0 dan QoS 1. Untuk pengujian transmisi data dilakukan dengan menghitung *Delay*, *Throughput* dan *Packet Loss* setiap protokol yang akan digunakan.

4.5.1 *Delay*

Setelah pengambilan data selesai dilakukan dan dirata-rata kemudian didapatkan hasil nilai *Delay* yang ditampilkan pada Tabel 4.5 sebagai berikut:

Tabel 4.5 Hasil Pengujian *Delay*

Delay (ms)						
No	Sampel	MQTT				TCP/IP
		QOS 0		QOS 1		
		PC	Smartphone	PC	Smartphone	
1	1	11,30	13,17	29,57	4,80	942,17
2	2	14,83	6,30	20,43	3,60	785,40
3	3	8,73	7,17	10,40	15,87	1039,00
4	4	33,33	10,13	4,93	30,33	929,77
5	5	18,77	10,10	35,73	10,97	930,77
Rata-rata		17,39	9,37	20,21	13,11	925,42



Gambar 4.7 Grafik Rata-Rata *Delay*

Pada Gambar 4.7 didapatkan nilai *delay* setiap *frame* dari protokol MQTT dan protokol TCP/IP melalui jalur internet disimpulkan bahwa *delay* pada protokol MQTT yang dikategorikan sangat bagus dengan besar delay kurang dari 150 ms sehingga jauh

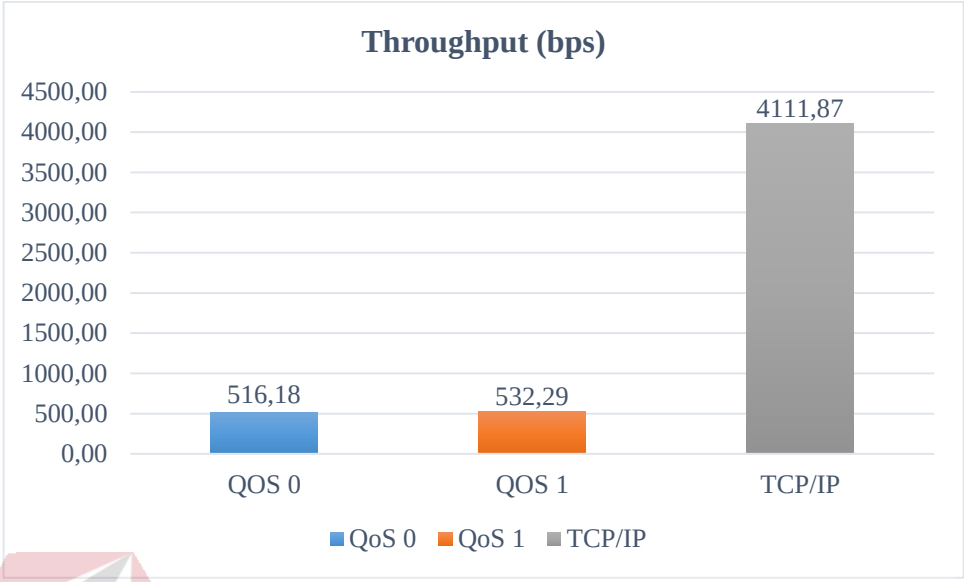
lebih cepat dibandingkan *Delay* dari protokol TCP/IP yang lebih dari 450 ms. Maka pada pengujian ini dapat disimpulkan bahwa pada saat aktifitas olahraga lari, untuk memonitoring dari jarak jauh lebih tepatnya menggunakan protokol MQTT sebagai jalur transmisi data melalui internet.

4.5.2 *Throughput*

Setelah pengambilan data selesai dilakukan dan dirata-rata kemudian didapatkan hasil nilai *Throughput* yang akan ditampilkan pada Tabel 4.6 sebagai berikut :

Tabel 4.6 Hasil Pengujian *Throughput*

Throughput (bps)				
No	Sampel	MQTT		TCP/IP
		QOS 0	QOS 1	
1	1	515,07	531,02	4196,667
2	2	516,71	532,22	4069,333
3	3	516,76	532,31	4104
4	4	516,58	533,02	4053,333
5	5	515,78	532,89	4136
Rata-rata		516,18	532,29	4111,87



Gambar 4.8 Grafik Rata-Rata *Throughput*

Pada Gambar 4.8 didapatkan nilai *Throughput* setiap *frame* dari protokol MQTT dan protokol TCP/IP melalui jalur internet disimpulkan sebagai berikut :

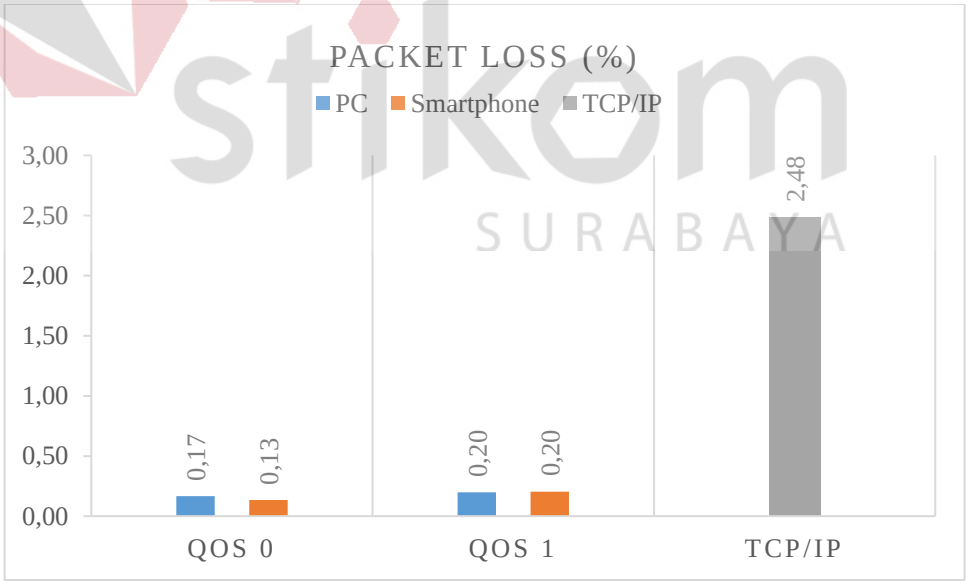
- Pada PC atau *smartphone* untuk QOS *level* 0 protokol MQTT menghasilkan *Throughput* sebesar 516,18 bps.
- Pada PC atau *smartphone* untuk QOS *level* 1 protokol MQTT menghasilkan *Throughput* yaitu diatasnya QOS *level* 0 sehingga terdapat PUBACK dan mendapatkan hasil 532,29 bps.
- TCP/IP memerlukan kecepatan transfer data sebesar 4111,87 bps. Dari kedua QOS yang ada dapat disimpulkan bahwa protokol MQTT lebih hemat daripada TCP/IP.

4.5.3 *Packet Loss*

Setelah pengambilan data selesai dilakukan dan dirata-rata kemudian didapatkan hasil nilai *Packet Loss* yang akan ditampilkan pada Tabel 4.7 sebagai berikut :

Tabel 4.7 Hasil Pengujian *Packet Loss*

Packet loss (%)						
No	Sampel	MQTT				TCP/IP
		QOS 0		QOS 1		
		PC	Smartphone	PC	Smartphone	
1	1	0,07	0,33	0,33	0,00	1,63
2	2	0,00	0,00	0,00	0,00	2,93
3	3	0,07	0,00	0,00	0,00	3,43
4	4	0,00	0,00	0,65	0,52	1,97
5	5	0,70	0,33	0,00	0,50	2,43
Rata-rata		0,17	0,13	0,20	0,20	2,48



Gambar 4.9 Grafik Rata-Rata *Packet Loss*

Pada Gambar 4.9 didapatkan nilai *Packet Loss* setiap *frame* dari protokol MQTT dan protokol TCP/IP melalui jalur internet disimpulkan sebagai berikut :

- Pada QOS 0 protokol MQTT menghasilkan *Packet Loss* sebesar 0,17% untuk PC dan 0,13% untuk *smartphone* yang tergolong sangat bagus.
- Pada QOS 1 protokol MQTT menghasilkan *Packet Loss* sebesar 0,2% untuk PC dan 0,2% untuk *smartphone* yang tergolong sangat bagus.
- Pada TCP/IP menghasilkan *Packet Loss* sebesar 2,48% sehingga tergolong sangat bagus.

BAB V

PENUTUP

Hasil dari pengujian pada rancang bangun transmisi data *Heart* Rate menggunakan protokol MQTT pada tugas akhir ini terdapat kesimpulan dan saran dari penulis diantaranya:

5.1 Kesimpulan

1. *Prototype heart rate* akan mengirim data menggunakan ESP8266 melalui protokol MQTT dan diterima oleh *broker*.
2. QoS untuk parameter *Delay* pada transmisi data *Heart Rate* menggunakan protokol MQTT tergolong sangat bagus, menghasilkan *delay* QoS *level* 0 sebesar 17,39 ms untuk PC 9,37 ms untuk *smartphone* dan QoS *level* 1 sebesar 20,21 ms untuk PC 13,11 ms untuk *smartphone*. Sedangkan pada TCP/IP mendapatkan 925,42 ms.
3. QoS untuk parameter *Throughput* pada transmisi data *Heart Rate* menggunakan protokol MQTT tiap pengiriman data dengan rata-rata QoS *level* 0 sebesar 516,18 bps dan QoS *level* 1 sebesar 532,29 bps sedangkan pada TCP/IP membutuhkan pengiriman data sebesar 4111,87 bps.
4. QoS untuk parameter *Packet Loss* pada transmisi data *Heart Rate* menggunakan protokol MQTT tergolong sangat bagus, menghasilkan *packet loss* kurang dari 1% dengan QoS *level* 0 sebesar 0,17% untuk PC 0,13% untuk *smartphone* dan QoS *level* 1 sebesar 0,2% untuk PC 0,2% untuk *smartphone*. Sedangkan pada TCP/IP mendapatkan 2,48% sehingga tergolong sangat bagus.

5.2 Saran

1. Dalam membuat sistem monitoring data *Heart Rate* diperlukan sebuah *database* yang mampu menampung data secara permanen seperti mySQL.
2. Disarankan untuk mengecilkan ukuran elektronik dengan merancang *embedded system* seperti SMD sehingga akan menjadikan perangkat lebih *wearable* dan ergonomis.
3. Perlu ditambahkan protokol UDP sebagai perbandingan kinerja protokol MQTT.

DAFTAR PUSTAKA

Agusriandi. (2019). Analisis Delay Jitter, Throughput, Dan Packet Lost Menggunakan IPERF3. *Academia.Edu*, hlm 1-3.

Chooruang, K., & Mangkalakeeree, P. (2016). Wireless Heart Rate Monitoring System Using MQTT. *iEEECON2016*, 161.

ETSI. (2002). Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 3; End-to-end Quality of Service in TIPHON systems; Part 7: Design guide for elements of a TIPHON connection from an end-to-end speech transmission performance point of. *ETSI*.

Govindan, Kannan & Azad, A.P. (2015). *End-to-end Service Assurance in IoT* MQTT-SN. *IEEE*, 290.

Musayyanah, Puspasari, I., & Susanto, P. (2018). Monitoring Target Heart Rate (THR) Untuk Optimalisasi Latihan Lari Berbasis Internet Of Things. *Jurnal Teknika*, 1.

Puspasari, I., Musayyanah, & Susanto, P. (2018). Telereport Target Heart Rate (THR) pada Cardio Exercise Berbasis Metode Karnoven. *SNATi*, A-42.

Saputra, G. Y. dkk. (2017). Penerapan Protokol MQTT Pada Teknologi WAN. Jurnal Informatika Mulawarman, 69.

Sutiono, M. A. (2016). *Rancang Bangun Smart Rice Cooker Menggunakan Protokol Komunikasi Wi-Fi dan Protokol Pertukaran Pesan MQTT*. Tangerang: Universitas Multimedia Nusantara.

Tarigan, S. O., Sitepu, H. I., & Hutagalung, M. (n.d.). Pengukuran Kinerja Sistem Publish/ Subscribe Menggunakan Protokol MQTT. *Telematika*, 27.

Xu, Yiming dkk. (2016). *Toward SDN-based Fog Computing*: MQTT *Broker Virtualization for Effective and Reliable Delivery*. *IEEE*, hlm 1-2.

Yulian, R., & Suprianto, B. (2017). Rancang Bangun *Photoplethysmography* (PPG) Tipe Gelang untuk Menghitung detak Jantung Berbasis Arduino. Jurnal Teknik Elektro, 230.