

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang berisi kajian pustaka dan dasar teori yang akan menunjang proses penelitian.

2.1 Kajian Pustaka

Kajian pustaka akan membahas beberapa penelitian yang telah ada dan telah diusulkan. Pada penelitian ini kajian pustaka akan diambil dari penelitian – penelitian yang relevan dan sudah pernah di implementasikan. Penggunaan penelitian yang relevant tersebut akan sangat membantu dalam proses penelitian karena akan digunakan sebagai data pendukung.

Untuk melakukan implementasi ataupun pengembangan interface Bluetooth Low Energy (BLE) pada komunikasi IoT Middleware telah dilakukan beberapa penelitian terdahulu seperti dari (Boulouache, 2015), dkk yang mendesain data collection system yang berbasis dengan BLE yang bekerja sebagai data collector. Tabel 2.1 akan menjelaskan kajian pustaka yang digunakan dalam penelitian.

Tabel 2. 1 Kajian Pustaka

NO	Judul	Penulis	Perbandingan	
			Kajian Pustaka	Skripsi Penulis
1	A BLE-based data collection system for IoT (2015)	Abd Elwahab Boulouache. (Boulouache-Nouali, 2015)	Menggunakan BLE sebagai data collector untuk mengambil data dari IoT	Menggunakan BLE sebagai penghubung dari node sensor ke IoT middleware
2	Bluetooth Low Energy (BLE) based Wireless sensor (2013)	Elke Mackensen and Matthias Lai. (Mackensen-Mathias, 2013)	Menggunakan BLE sebagai basis dari sensor tanpa kabel	Menggunakan BLE sebagai pengganti dari penggunaan WiFi dan Ethernet

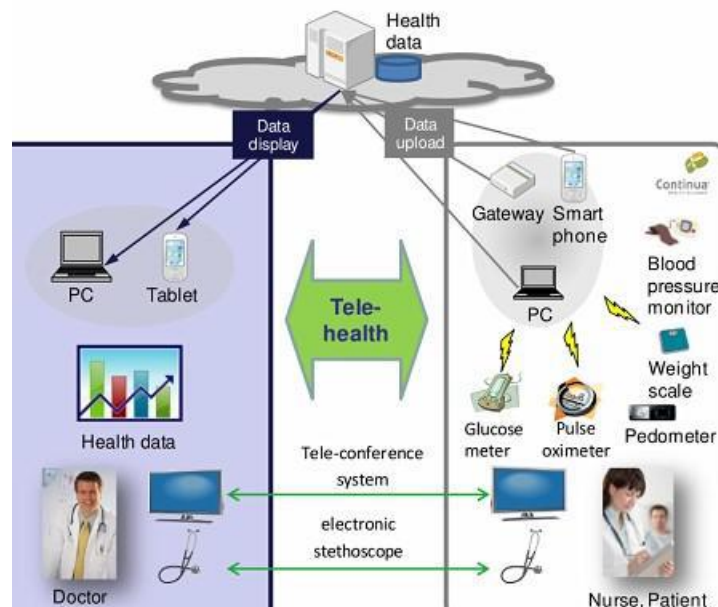
2.2 Dasar Teori

Berdasarkan informasi yang bisa diambil dari kajian pustaka, maka didalam proses “Pengembangan Interface Bluetooth Low Energy (BLE) pada komunikasi IoT Middleware untuk mendukung *Network Interoperability*” berikut adalah beberapa dasar teori yang digunakan, antara lain:

2.2.1 Internet of Thing (IoT)

Menurut (Burange,2015) *Internet of Things* (IoT) adalah susunan dimana objek dan manusia diberikan identitas eksklusif dan kemampuan untuk bertukar informasi tanpa memerlukan hubungan dua arah antara manusia ke manusia, atau manusia ke komputer. IoT perkembangannya sangat cepat dan menjanjikan untuk mengoptimalkan kegiatan pada kehidupan sehari – hari dengan bantuan sensor cerdas dan perangkat pintar yang berkerja sama melalui jaringan internet. Sebagian besar proses pada IoT dilakukan oleh sensor yang melakukan konversi data fisik mentah menjadi sinyal digital dan dikirimkan ke *control center*. Dengan cara seperti ini, maka proses untuk memonitor setiap perubahan di lingkungan yang jaraknya cukup jauh dapat menjadi lebih mudah melalui jaringan internet.

Penelitian pada IoT sudah banyak dilakukan untuk diterapkan di beberapa bidang keilmuan, seperti dibidang kesehatan, industry, informatika, dll. Contohnya penelitian milik (Rietal, 2014) mengerjakan riset tentang memonitoring kesehatan seorang pasien menggunakan sensor wireless yang diletakan pada tubuh pasien, hal – hal yang dipantau adalah tekanan darah pasien, psikologi, dan detak jantung, semua kegiatan tersebut dilakukan secara remote melalui sensor yang dipasangkan di tubuh pasien.



Gambar 2. 1 Pasien Monitoring

Masih pada bidang kesehatan dan medis seperti gambar 2.1, IoT juga diterapkan untuk melayani konsultasi pasien secara remote dimana kegiatan tersebut didukung menggunakan jaringan wireless dan internet.

2.2.2 IoT Middleware

IoT Middleware adalah sebuah komputer mini yang awalnya ditujukan hanya untuk kepentingan edukasi akan tetapi sekarang dikarenakan harganya yang semakin murah IoT Middleware juga digunakan sebagai kit elektronik di rumah (Bandyopadhyay, 2011). IoT middleware yang memiliki banyak sekali fungsi diantaranya adalah:

2.2.2.1 Komputer Desktop Mini

Raspberry Pi 3 memiliki kemampuan yang sangat baik jauh melebihi dari keluaran generasi pertamanya, pada generasi awal Rpi hanya memiliki ram sebesar 256 MB dan CPU berinti Tunggal (Single Core). Namun pada Rpi generasi 3 ram yang ditanamkan adalah sebesar 1 Gb dan terdapat CPU quad core, berikut adalah gambar dan spesifikasi lengkap dari Raspberry Pi 3 dapat dilihat pada gambar 2.2 dan tabel 2.2:



Gambar 2. 2 Raspberry Pi 3
[sumber : www.raspberrypi.org]

Tabel 2. 2 Spesifikasi Raspberry Pi 3

SPESIFIKASI RASPBERRY PI 3
A 1.2GHz 64-bit quad-core ARMv8 CPU
802.11n Wireless LAN
Bluetooth 4.1
Bluetooth Low Energy (BLE)
1GB RAM
4 USB ports
40 GPIO pins
Full HDMI port
Ethernet port
Combined 3.5mm audio jack and composite video
Camera interface (CSI)
Display interface (DSI)
Micro SD card slot (now push-pull rather than push-push)
VideoCore IV 3D graphics core

Pada Raspberry Pi 3 tidak disediakan power switch, yang bekerja untuk memberika suplai power ke raspberry pi adalah *Micro USB*, digunakannya *Micro USB* karena harganya yang murah dan sangat mudah untuk didapatkan. Raspberry Pi sendiri hanya membutuhkan daya sebesar 5V dengan arus minimal 700mA (Richardson and Wallace, 2012).

2.2.2.2 File Server

Pada generasi raspberry pi apapun pengguna dapat membuatnya sebagai File server dengan konfigurasi yang tepat dan membutuhkan sebuah hardisk eksternal yang akan di hubungkan ke Raspberry Pi maka akan menjadi sebuah media file server yang dapat digunakan dan dipakai oleh beberapa client (Rudito-Sularsa, 2015).

2.2.2.3 Access Point

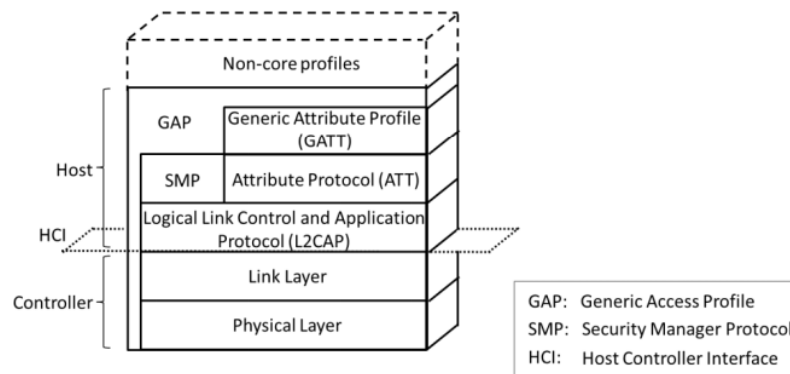
Dengan memasang *Adapter WiFi* yang *compatible* dengan Raspberry Pi, maka Raspberry Pi dapat berubah menjadi sebuah access point. Untuk dapat melakukan hal ini diperlukan tambahan aplikasi yang berguna untuk bekerja sebagai server dan untuk mengelola adapter WiFi yang akan berfungsi sebagai access point serta melakukan proses validasi terhadap permintaan koneksi dari client yang akan terhubung ke Raspberry pi (Richardson and Wallace, 2012).

2.2.3 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) merupakan sebuah teknologi jaringan tanpa kabel (*wireless*) yang dikembangkan oleh Bluetooth Special Interest Group (SIG) untuk komunikasi jarak pendek. Berbeda dengan Bluetooth sebelumnya BLE didesain sebagai kontrol dan monitoring aplikasi yang memiliki konsumsi daya yang kecil (Gomez-Oller, 2012).

BLE muncul ketika sudah ada banyak teknologi wireless yang memiliki konsumsi daya rendah seperti Zigbee, 6LoWPAN ataupun Zwave. Ketiga teknologi tersebut sudah banyak digunakan untuk aplikasi yang membutuhkan jaringan multihop. Akan tetapi BLE merupakan solusi dari jaringan single hop yang sudah berbeda ruang lingkup penggunaannya contoh kesehatan, smart energy, dan keamanan (Gomez-Oller, 2012).

Seperti halnya Bluetooth classic, Protokol stack BLE tersusun oleh 2 bagian utama yaitu Kontroller dan Host. Kontroller terdiri dari *Physical Layer* dan *Link Layer* dan biasanya diimplementasikan sebagai System-On-Chip(SOC) kecil yang terintegrasi dengan middleware. Untuk Host sendiri berjalan pada application processor dan layer di atasnya, contoh *Logical Link Control and Adaptation Protocol (L2CAP)*, *Attribute Protocol (ATT)*, *Generic Attribute Profile (GATT)*, *Security Manager Protocol (SMP)* dan *Generic Access Profile (GAP)*. Komunikasi antara Host dan Kontroller di standarisasi oleh *Host Controller Interface (HCI)* (Gomez-Oller, 2012). Susunan layer protokol Bluetooth dapat dilihat pada gambar 2.3:



Gambar 2. 3 Bluetooth Layer
[Sumber: Gomez-Oller 2012]

2.2.4 Bluetooth Low Energy Gateway

Bluetooth Low Energy (BLE) gateway yang digunakan dalam penelitian ini ialah EspruinoHub. EspruinoHub merupakan BLE bridge ke MQTT yang ditanamkan pada Middleware untuk melakukan translasi data yang awalnya dikirimkan menggunakan jaringan transmisi BLE kemudian dirubah menjadi MQTT. Cara kerja EspruinoHub sendiri adalah Middleware yang sudah tertanam module BLE akan menangkap Transmisi BLE dari sensor kemudian setelah mendapatkan data tersebut EspruinoHub akan bekerja merubah transmisi BLE menjadi jaringan Internet menggunakan Protokol MQTT.

2.2.5 Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) adalah protokol berbasis publish-subscribe yang ringan digunakan di atas protokol TCP/IP (Abdillah,2015). MQTT memiliki sifat terbuka, simple, yang didesain agar mudah untuk diimplementasikan. MQTT cukup ideal digunakan dalam situasi yang berbeda – beda, Protokol MQTT menjamin terkirimnya pesan, meskipun koneksi terputus untk sementara waktu.

2.2.6 Generic Attribute Profile (GATT)

GATT atau *Generic Attribute Profile* adalah sebuah interface yang digunakan agar dua perangkat BLE dapat terhubung dan mentransfer data satu sama lain dengan menggunakan konsep Servis dan karakteristik. GATT muncul ketika ada 2 perangkat yang saling terhubung secara eksklusif. Eksklusif dimaksudkan bahwa koneksi antara perangkat hanya bisa terjadi secara berpasangan tidak bisa lebih dari 2 perangkat yang terkoneksi (Gupta,2016).

2.2.7 Arduino IDE

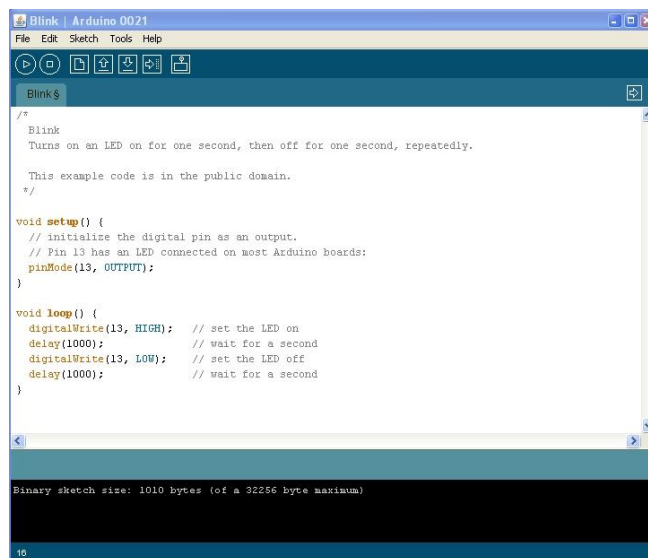
Arduino IDE merupakan editor program yang nantinya akan digunakan untuk menulis program, mengcompile, serta mengunggah program ke Arduino board. *Arduino Integrated Development Environment* berisi text editor untuk menulis

kode, area pesan, toolbar dengan tombol – tombol umum dan beberapa deretan menu (Louis, 2016).

Program yang ditulis dengan Arduino IDE disebut dengan *Sketches*. Sketches ini ditulis dengan text editor dan disave dengan ekstensi file *.ino*, didalam Arduino IDE terdapat beberapa menu yang sudah umum contohnya :

- Verify
Berfungsi untuk mengecek error pada saat mengcompile kode.
- Upload
Berfungsi untuk mengcompile kode dan akan langsung di unggah kedalam board Arduino.
- New
Berfungsi untuk membuat Sketches baru
- Open
Berfungsi untuk membuka sketches yang sudah ada didalam Arduino IDE
- Save
Berfungsi untuk menyimpan sketches.

Arduino IDE sampai Saat ini bersifat open source itulah yang membuat banyak para akademisi dan professional menggunakannya, sehingga library serta modul yang tersedia pada Arduino IDE sangat banyak dan mudah untuk diakses. Tampilan Arduino IDE dapat dilihat pada gambar 2.4 berikut:



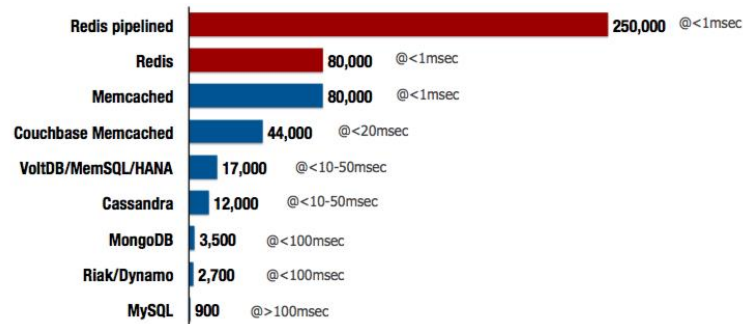
Gambar 2. 4 Arduino IDE
[Sumber : W. Durfee 2011]

2.2.8 Redis

Redis adalah generasi terbaru dari database NoSQL yang mempunyai penyimpanan data didalam memori. Redis sederhanya merupakan *Key value pair based data system* yang mendukung semua struktur data seperti variable, Linked

List, array, string, dan Queues. Namun tidak seperti database konvensional, redis tidak menyediakan keamanan yang cukup untuk melindungi data, Siapapun bisa mengakses data tersebut apabila mempunyai kuncinya dikarenakan data disimpan dalam bentuk pasangan kunci.

Dikarenakan redis mempunyai lokasi penyimpanan didalam memori, maka dipastikan kemampuan redis dalam menjalankan perintah sangatlah cepat. Redis dapat menjalankan seratus operasi perintah per detiknya, sehingga redis sangatlah cocok digunakan dalam system yang membutuhkan pemantauan secara *real-time*. Berikut adalah perbandingan performansi redis dengan database lainnya yang dapat dilihat pada **Gambar 2.6** :



Gambar 2. 5 Perbandingan Performa Redis dengan Database Lain

[Sumber: Redis.io]

2.2.9 Quality of Services

Salah satu fitur utama dari MQTT adalah level kualitas layanan atau *Quality of Service* (QoS). Pesan-pesan yang di *publish* pasti memiliki salah satu dari 3 level QoS. Level-level ini memberikan garansi akan konsistensi (*reliability*) dari pengiriman pesan. *Client* dan broker menyediakan mekanisme penyimpanan dan pengiriman kembali dari pesan sehingga meningkatkan konsistensi akibat kegagalan jaringan, aplikasi berhenti atau sebab-sebab lainnya.

a. Level 0

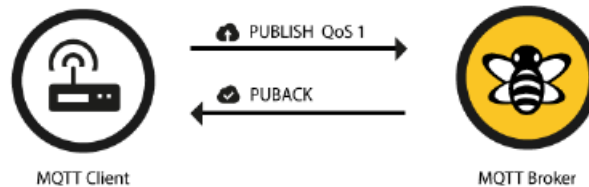
Pesan hanya akan dikirimkan sekali. Pesan yang dikirim tergantung dari *reliability* TCP atau tergantung pada ketersediaan jaringan dan tidak ada usaha untuk mengirim pesan kembali apabila terjadi kegagalan.



Gambar 2. 6 QoS 0 pada MQTT (HiveMQ, 2015)

b. Level 1

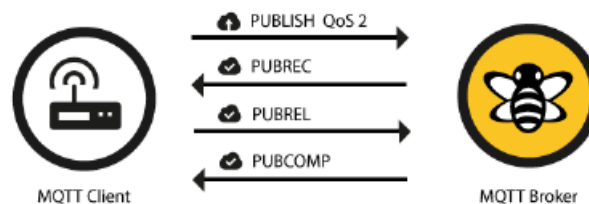
Pesan akan dikirimkan setidaknya satu kali. Jadi *client* akan menerima pesan setidaknya sekali (dapat terjadi duplikasi). Apabila *subscriber* tidak mengirimkan ACK, maka broker akan mengirimkan kembali sampai ia menerima ACK dari *client*.



Gambar 2. 7 QoS 1 pada MQTT (HiveMQ, 2015)

c. Level 2

Pesan pasti diterima satu kali. Protokol dengan level ini memastikan bahwa pesan tersampaikan dan tidak terjadi duplikasi pesan terkirim. Hal yang perlu diperhatikan adalah semakin tinggi level QoS maka semakin tinggi pula *overhead* yang terjadi.



Gambar 2. 8 QoS 2 pada MQTT (HiveMQ, 2015)

2.2.10 NodeMCU (ESP32)

NodeMCU (ESP32) adalah sebuah mikrokontroler yang bisa berfungsi sebagai *slave* ataupun *host* untuk sebuah perangkat. ESP32 dapat berinteraksi menggunakan dua transmisi sekaligus, yaitu WiFi dan *Bluetooth Low Energy*. ESP32 merupakan MCU yang paling sedikit dalam konsumsi daya dikarenakan ESP32 merupakan generasi terbaru dari NodeMCU.

2.2.11 Pengujian Interoperability

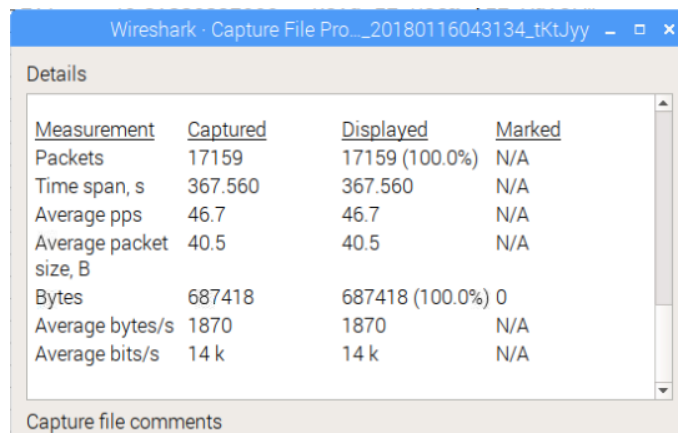
Interoperabilitas didefinisikan sebagai kemampuan dua (atau lebih) sistem atau komponen untuk saling bertukar data dan memproses data tersebut. Dengan berkembangnya IoT, kebutuhan akan sebuah sistem yang mendukung interoperabilitas semakin meningkat. Berawal dari kebutuhan ini, beberapa penelitian sudah dilakukan untuk merumuskan metode pengujian tentang interoperabilitas sebuah sistem. Hasil dari metode pengujian ini harus mampu

menunjukkan tingkat interoperabilitas sebuah sistem serta dapat digunakan sebagai bahan untuk meningkatkan interoperabilitas yang sudah ada.

(Reza Rezaei, 2014) telah melakukan penelitian terkait metode pengujian interoperabilitas sebuah sistem. Penelitian ini membahas semua metode yang sudah ada dan membandingkannya dilihat dari seberapa luas cakupan interoperabilitas yang diuji. Untuk mengevaluasi metode pengujian yang ada, pertama peneliti menyebutkan cakupan interoperabilitas berdasarkan hasil penelitian oleh S. Koussouris (2011). Cakupan interoperabilitas ini dibagi menjadi 4 level. Level pertama meliputi *Data interoperability, Process interoperability, Rules interoperability, Object interoperability, Software system interoperability* dan *Cultural interoperability*. Level kedua meliputi *Knowledge interoperability, Services interoperability, Social network interoperability, Electornic interoperability*. Level ketiga yakni *Cloud interoperability* sedangkan level terakhir yakni *Ecosystem interoperability*.

2.2.11.1 Menghitung Bandwidth Consumption

Pada penelitian ini akan dilakukan pengambilan data untuk mendapatkan data terkait konsumsi *Bandwidth* pada saat proses pengiriman data. Untuk mendapatkan data Konsumsi *bandwidth* akan digunakan program wireshark untuk meng *capture* datanya. Contoh untuk melihat *bandwidth consumption* pada wireshark adalah seperti berikut:



The screenshot shows the 'Statistics' window in Wireshark. It displays a table with four columns: Measurement, Captured, Displayed, and Marked. The data is as follows:

Measurement	Captured	Displayed	Marked
Packets	17159	17159 (100.0%)	N/A
Time span, s	367.560	367.560	N/A
Average pps	46.7	46.7	N/A
Average packet size, B	40.5	40.5	N/A
Bytes	687418	687418 (100.0%)	0
Average bytes/s	1870	1870	N/A
Average bits/s	14 k	14 k	N/A

Gambar 2. 9 Statistik Paket Wireshark

Pada gambar 2.10 dapat dilihat informasi apa saja yang dapat dilihat pada menu statistik didalam wireshark, untuk mengambil data tentang *bandwidth consumption* dapat dilihat pada *Average bits/s* atau rata – rata bits/detik dari proses pengiriman.

2.2.11.2 Menghitung CPU dan Memory Usage

Pada penelitian ini akan dilakukan pengambilan data guna mendapatkan data terkait dengan CPU dan memori usage didalam middleware saat terjadi proses pengiriman data. Untuk mendapatkan data CPU dan memori dijalankan

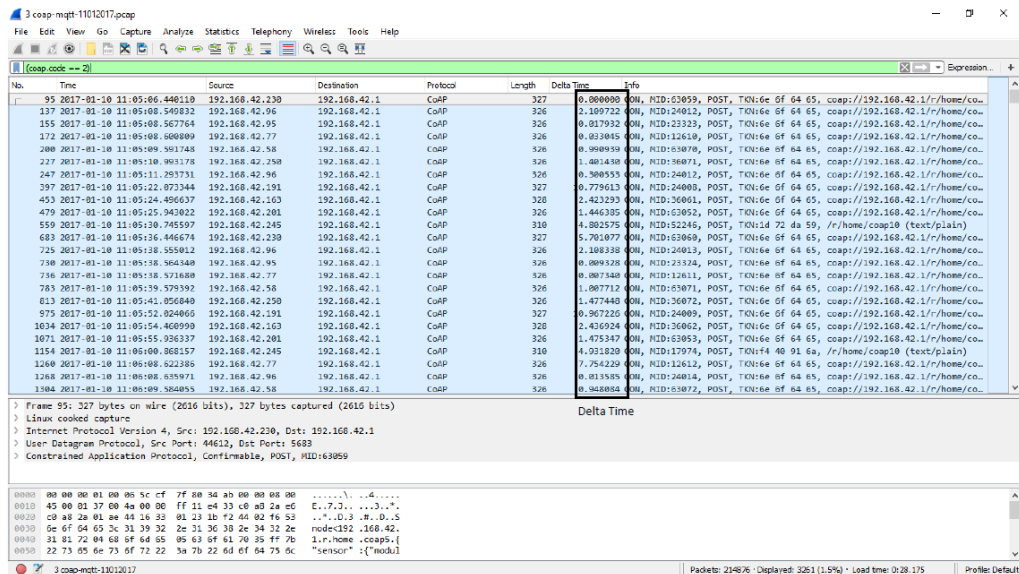
program yang dapat mencatat penggunaan CPU dan memori berdasarkan PID (*process ID*) dari middleware setiap 15 detik sekali (Kode Program 2.2).

```
var usage = require('usage');
var pid = 1057
setInterval(function() {
  var options = { keepHistory:true};
  usage.lookup(pid, options, function(err, stat) {
    console.log(err,stat);
    console.log(new Date().toISOString());
  });
},15000);
```

Kode Program 2.1 Penggunaan CPU dan memori

2.2.11.3 Menghitung Delay Menggunakan Wireshark

Terdapat banyak parameter untuk menentukan performansi dari sebuah jaringan salah satunya *delay*. Delay sendiri juga dapat dipengaruhi oleh banyak hal diantara lain jarak, *latency*, dan fisik dari perangkat tersebut. Penghitungan delay pada sebuah jaringan dapat dilakukan dengan melakukan capture data dengan bantuan aplikasi Wireshark, dengan cara melihat delta time yang tersedia didalam wireshark (rozi, 2017). Contoh melihat *delay* dengan menggunakan wireshark adalah seperti berikut:



Gambar 2. 10 Delta Time Wireshark

[Sumber: Rozi, 2017]