PROPOSAL SKRIPSI

PENERAPAN MOSQUITTO SEBAGAI SERVER MQTT LOKAL PADA KOMUNIKASI ANTARA DATABASE MONGODB DAN MESIN ABSENSI *MULTI-NODE*

PENGUSUL:
KHIARUL ARHAM (1903423002)

PROGRAM STUDI BROADBAND MULTIMEDIA
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI JAKARTA
FEBRUARI 2020

LEMBAR PERSETUJUAN
CALON PEMBIMBING SKRIPSI

1. Judul	: Penerapan Mosquitto sebagai Server MQTT Lokal pada Komunikasi Antara Database MongoDB dan Mesin Absensi *Multi-Node*
2. Bentuk Tugas Akhir	: Rancang Bangun
3. Personalia Tugas Akhir	
a. Nama Mahasiswa	: Khiarul Arham
NIM	: 1903423002
IPK	: -
Judul	: Penerapan Mosquitto sebagai Server MQTT Lokal pada Komunikasi Antara Database MongoDB dan Mesin Absensi *Multi-Node*
4. Perkiraan Biaya	: Rp 832.000
5. Alokasi Waktu Pelaksanaan	: 5 Bulan

Calon Pembimbing 1	Calon Pembimbing 2
Agus Wagyana, S.T., M.T.	M. Fathurahman, S.T., M.T.
NIP. 196808241999031002	NIP. 19710824200312100l

PENILAIAN PROPOSAL SKRIPSI

JURUSAN TEKNIK ELEKTRO

JUDUL : Penerapan Mosquitto sebagai Server MQTT Lokal pada Komunikasi Antara Database MongoDB dan Mesin Absensi *Multi-Node*

KRITERIA SKRIPSI

NO	KRITERIA	INDIKATOR PENILAIAN	BOBOT	SKOR	NILAI
1	Orientasi Permasalahan Dan Pustaka	a. Latar Belakang b. Perumusan Masalah c. Tujuan d. Tinjauan Pustaka	25		
2	Pola Penyelesaian Masalah	a. Metode Pelaksanaan Tugas Akhir	35		
3	Fisibilitas Sumber Daya	a. Jadwal Pelaksanaan b. Personalia TA c. Perkiraan Biaya	15		
4	Kebahasaan	a. Bahasa Proposal b. Daftar Pustaka (keserasian dan substansi kemutakhiran)	25		
Nilai Total					

1. Masing-masing kriteria diberi skor 1, 2, 4, dan 5 (1=sangat kurang, 2=kurang, 4=baik, 5=sangat baik) yang mencerminkan skor seluruh butir yang dinilai dalam masing-masing kriteria.
2. Nilai = Skor x Bobot; Nilai Total = N1+N2+N3+N4+N5
3. Hasil Penilaian : Nilai Total ≥ 400 (Diterima) ; Nilai Total < 400 (Ditolak)

Depok,

Penilai

( )

NIP.

Saran untuk Pengusul :

I. PENDAHULUAN

1.1 Latar Belakang

Sistem absensi seringkali menjadi perhatian sebuah instansi dalam pengelolaan puluhan atau bahkan ratusan sumber daya manusia (karyawan), sehingga banyak perusahaan telah menerapkan sistem absensi digital. Kebanyakan instansi atau perusahaan memiliki gedung yang berdekatan atau hanya satu gedung, sehingga cukup meletakkan 1 atau 2 mesin absensi dengan pengumpulan data manual menggunakan *flashdisk* tanpa harus mengintegrasikannya ke jaringan lokal instansi tersebut. Beda halnya dengan instansi dengan area yang luas dan jarak antar gedung yang cukup jauh, sehingga mengharuskan instansi tersebut memasang mesin absensi di masing-masing gedung tempat karyawan bekerja yang jumlahnya tidak sedikit. Tentunya sangat merepotkan untuk mengumpulkan data di tempat-tempat yang berjauhan secara manual untuk masing-masing mesin. Kalaupun masing-masing gedung memiliki seorang perwakilan untuk mengumpulkan datanya, maka untuk menggabungkan semua data tersebut akan merepotkan bagian administrasi bila dilakukan setiap hari atau setiap minggu.

Untuk mesin yang memiliki fitur *logging* ke jaringan sendiri masih terbilang cukup mahal dan rata-rata memiliki database yang terpisah (*standalone*), yang beredar di pasaran misalnya ZT1200 atau ZT1100 yang menggunakan MySQL sebagai *standalone* database. Bagi instansi sejenis pabrik, manufaktur, atau bisnis dengan kemampuan *budget* tinggi, itu bukanlah masalah besar. Namun untuk instansi non-bisnis dengan kemampuan *budget* menengah atau instansi pada umumnya, dan ditambah dengan area luas, hal itu benar-benar harus dipertimbangkan. Oleh karena itu masih banyak yang menggunakan cara manual untuk perekapan data absensi.

Sekarang ini dengan adanya perkembangan sistem integrasi antar perangkat dengan jaringan seperti *Internet of Things* (IoT) ditambah dengan penerapannya yang tidak hanya pada komputer atau perangkat jaringan pada umumnya. Begitu mudahnya sekarang ini dalam menemukan *source code* atau *binary resource* untuk berbagai macam protokol pendukung IoT yang disediakan untuk para *hobby* dan *developer*, salah satunya protokol *Message Queiuing Telemetry Transport* (MQTT). MQTT

menjadi salah satu protokol yang banyak digunakan pada aplikasi IoT karena memiliki ukuran paket data yang kecil sehingga cocok digunakan pada aplikasi *real-time* atau aplikasi yang membutuhkan kemampuan *data streaming* yang baik. Dari beberapa info yang penulis dapatkan dari dosen dan referensi di internet, protokol MQTT sudah beberapa kali dijadikan bahan pembahasan untuk keperluan tugas akhir atau skripsi, namun masih memanfaatkan *cloud* MQTT *broker* yang sudah siap pakai. Dalam penerapannya *cloud* MQTT *broker* harus menggunakan koneksi internet walaupun hanya untuk perangkat yang letaknya tidak jauh atau berdekatan. Tentu akan menghambat komunikasi antar *client* dengan *cloud broker* bila saat koneksi internet atau *Internet Service Provider* (ISP) yang sering mengalami gangguan terutama instansi yang berada di daerah tepencil atau masih belum dijangkau banyak ISP lain yang handal. Sehingga untuk mengatasinya agar lebih efisien perlu adanya penerapan MQTT yang cukup memanfaatkan jaringan lokal yang sudah tersedia di suatu instansi tanpa harus memakan *bandwidth* internet.

Salah satu penyedia *open-source* MQTT *client & broker* yaitu Eclipse Mosquitto. Dengan menggunakan Mosquitto kita dapat menerapkan protokol MQTT pada jaringan lokal. Mosquitto juga menyediakan *binary* yang mendukung berbagai macam sistem operasi seperti Windows, Linux, dan Mac. Bahkan Mosquitto juga dapat dipasang pada *single board computer* (SBC) seperti Raspberry Pi. Dengan memanfaatkan *low power single board computer* sebagai MQTT *broker* maka kita tidak perlu menggunakan sebuah *Personal Computer* (PC) yang harus *standby* selama koneksi dengan *broker* diperlukan terkait pertimbangan penggunaan daya. Selain itu output data dari *client* juga dapat ditampilkan pada sebuah database dengan merancang GUI pada SBC tersebut. Salah satu *platform* GUI yang dapat digunakan, yaitu Qt yang juga menyediakan *tools designer* untuk mempermudah *developer* saat merancang GUI dengan fitur konversi hasil desain ke bahasa pemrograman seperti C/C++ ataupun Python (PyQt). Maka dari pemaparan di atas penulis akan membuat sebuah skripsi dengan judul **“Penerapan Mosquitto Sebagai Server MQTT Lokal pada Komunikasi Antara Database MongoDB dan Mesin Absensi Multi-node”**.

1.2 Perumusan Masalah

Berdasarkan uraian latar belakang, maka dapat dirumuskan masalah yang akan dibahas:

1. Cara menerapkan MQTT pada mesin absensi menggunakan jaringan lokal.
2. Proses implementasi *library* Mosquitto pada *Single Board Computer* (SBC) dalam hal ini Raspberry Pi.
3. Perancangan algoritma komunikasi antara beberapa *Node* atau *Client* (mesin absensi) dengan *Broker* MQTT sekaligus mengakses database.

1.3 Tujuan

Adapun tujuan dari perancangan yang akan dilakukan adalah sebagai berikut:

1. Menerapkan protokol MQTT pada mesin absensi di *local area network* dengan mengandalkan *access point* terdekat untuk komunikasi lebih handal dan aman.
2. Menginstalasi Mosquitto pada *Single Board Computer* (SBC) salah satunya Raspberry Pi.
3. Menerapkan algoritma komunikasi *bidirectional* dari protokol MQTT pada perangkat-perangkat yang terhubung pada jaringan lokal atau jaringan publik.
4. Menggunakan aplikasi GUI *designer* PyQt5 dan penerapannya untuk menginput, mengedit, dan menampilkan data dari suatu database dengan konsep *NoSQL* salah satunya MongoDB dengan *library* Pymongo pada Python.

1.4 Luaran

Beberapa manfaat yang dapat diperoleh dari perancangan di bawah ini adalah:

1. Diterapkannya teknologi mesin absensi secara *wireless* dengan memanfaatkan jaringan lokal untuk *logging* kehadiran karyawan pada umumnya bagi instansi dengan area yang luas dan yang sering mengalami gangguan koneksi ke ISP.
2. Sebagai referensi baru mengenai penerapan *open-source* MQTT *broker* untuk modul pembelajaran dan praktikum (*jobsheet*).

3. Publikasi untuk proyek *Small and Cheap Attendance Node* (SCAN) yang *open-source* dalam forum komunitas elektronika, mikrokontroler, dan otomasi sehingga dapat dipelejari, diterapkan, dan dikembangkan secara umum.

II. TINJAUAN PUSTAKA

2.1 Protokol MQTT

Protokol MQTT menggunakan konsep *publish/subscribe* yang sangat berbeda dengan konsep protokol yang lain seperti HTTP yang menggunakan paradigma *request/response*. Titik pusat komunikasi adalah MQTT *broker* yang bertanggung jawab untuk mengirim semua pesan di antara pengirim dan penerima yang sah. [2]

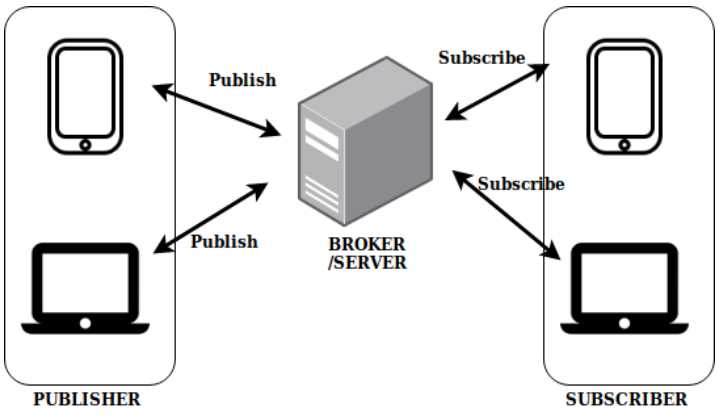
Setiap *client* yang mengirim pesan ke broker harus menyisipkan informasi nama topik yang dituju. Topik adalah informasi routing untuk *broker* yang bertugas mengarahkan pesan yang masuk kedalam *broker*. *Client* yang ingin menerima pesan harus melakukan *subscribe* ke topik yang dituju dan selanjutnya *broker* akan mengirimkan semua pesan yang diarahkan pada topik tersebut. Oleh karena itu *client* tidak perlu saling mengenal, mereka dapat berkomunikasi melalui topik ini. [2]

Perbedaan mendasar dengan protokol HTTP adalah *client* tidak perlu menarik informasi yang dibutuhkannya, melainkan broker akan mengirimkan informasi ke client jika ada pesan baru yang masuk ke sebuah topik. Jika koneksi mengalami gangguan maka MQTT *broker* dapat menyimpan semua pesan yang masuk dan mengirimkannya kembali ke *client* ketika koneksi kembali normal. [2]

MQTT *Client* dapat bertindak sebagai *publisher* ataupun *subscriber*. MQTT *Client* dapat berupa peralatan mikrokontroler maupun aplikasi komputer yang ditanamkan *library* untuk dapat melakukan koneksi ke MQTT *broker*. Hardware bisa sangat simpel (*constrained device*) yang terhubung secara wireless menggunakan WiFi, maupun kabel ethernet. Implementasi Protokol MQTT pada *client* sangatlah mudah, hal ini merupakan salah satu alasan mengapa MQTT ideal dan cocok untuk *small devices*. *Library* MQTT client tersedia untuk banyak bahasa pemrograman seperti Android, Arduino, C, C++, C#, Go, iOS, Java, JavaScript, .NET. MQTT *client* dapat berupa objek yang mengirim dan menerima data. Tipe MQTT *client* tergantung

fungsi kerjanya apakah sebagai *publisher* ataukah *subscriber*. MQTT *Broker* adalah *central device* yang berfungsi untuk memproses komunikasi antara MQTT *client* dan mendistribusikan pesan dari mereka pada topik-topik yang dituju. MQTT *broker* dapat menangani ribuan *devices* dalam satu waktu. [2]

Arsitektur MQTT terdiri dari tiga komponen yaitu *publisher*, *broker*, dan *subscriber*. *Device* yang menginginkan untuk terhubung pada suatu topik harus diregistrasikan pada topik tersebut sebagai *subscriber* sehingga ketika *publisher* mempublish informasi pada topik tersebut *broker* akan meneruskan informasi ke *subscriber*. Otorisasi *subscriber* dan *publisher* dicek oleh *broker* untuk memastikan keamanan terjaga sehingga bisa dipastikan bahwa *publisher* dan *subscriber* telah diregristasikan pada *broker*. [2]



Gambar 2. 1 Arsitektur MQTT
(Sumber: Dasar Implementasi Protokol MQTT Menggunakan Python dan NodeMCU, 2019)

2.2 Mosquitto

Mosquitto menyediakan server yang sesuai standar dan implementasi *client* dari protokol MQTT. MQTT menggunakan model *publish/subscribe*, memiliki overhead jaringan yang rendah dan dapat diimplementasikan pada perangkat berdaya rendah seperti mikrokontroler yang mungkin digunakan secara *remote* pada sensor Internet of Things. Dengan demikian, Mosquitto dimaksudkan untuk digunakan dalam semua situasi di mana ada kebutuhan untuk pengiriman pesan yang ringan, terutama pada perangkat dengan *resource* terbatas. [6]

Proyek Mosquitto merupakan bagian dari Eclipse Foundation. Ada 3 bagian pada proyek:

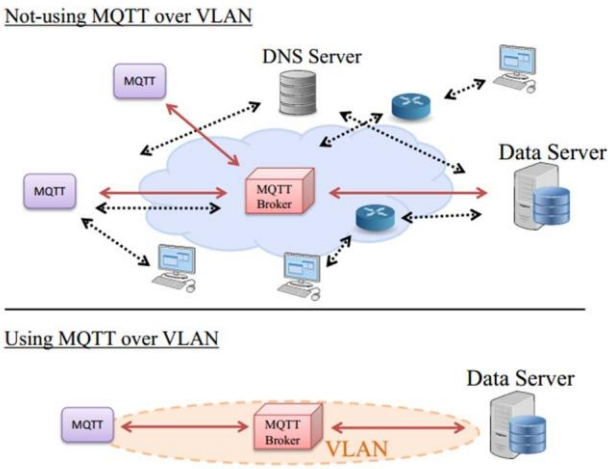
- Server utama Mosquitto
- mosquitto_pub dan mosquitto_sub *client utilities* yang merupakan salah satu metode untuk berkomunikasi dengan server MQTT.
- *Library* MQTT *client* yang ditulis dalam C/C++ *wrapper*.

Mosquitto memungkinkan penelitian yang terkait langsung dengan protokol MQTT itu sendiri, seperti membandingkan kinerja MQTT dan *Constrained Application Protocol* (CoAP) [11] atau menyelidiki penggunaan OAuth di MQTT [4]. Mosquitto mendukung kegiatan penelitian lainnya sebagai blok yang berguna untuk membangun sistem berskala besar dan telah digunakan untuk mengevaluasi MQTT untuk digunakan dalam *Smart City Services* [1] dan dalam pengembangan sistem pemantauan lingkungan [3]. Mosquitto juga telah digunakan untuk mendukung penelitian yang langsung digunakan sebagai bagian dari skema untuk remote control dari suatu eksperimen [9].

Di luar akademisi, Mosquitto digunakan dalam proyek *open-source* lainnya seperti proyek otomasi openHAB dan OwnTracks, proyek pelacakan lokasi pribadi, dan telah terintegrasi ke dalam produk komersial. [6]

2.3 MQTT pada VLAN

Komunikasi MQTT pada *Wide Area Network* (WAN) dapat dibagi menjadi jaringan yang lebih kecil menggunakan VLAN. *Overhead* pada DNS dari jaringan lain tidak berpengaruh ke jaringan yang lebih kecil. Sehingga *delay* dan antrian yang terjadi dapat dikurangi. [8]

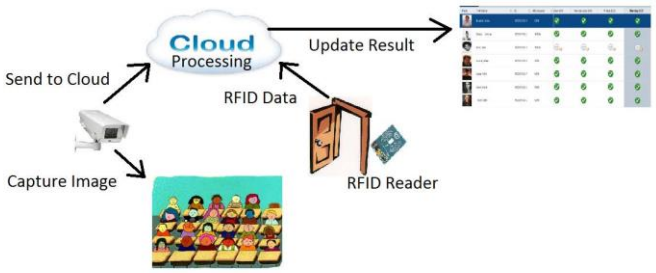


Gambar 2. 2 Arsitektur Jaringan MQTT dengan DNS Server dan VLAN

(Sumber: Sasaki. MQTT over VLAN for Reduction of Overhead on Information Discovery, 2019)

2.4 *Attendance Based IoT and RFID System*

RFID dan sistem IoT banyak diterapkan di berbagai bidang yang memerlukan proses identifikasi seperti transportasi, medikal, perusahaan, dan keamanan. *Tracking* kendaraan dan juga autentikasi pintu dikarenakan bersifat *low cost device* [10]. Dengan IoT sistem absensi dapat diterapkan secara *cloud* sehingga dapat menjadi fleksibel dan dapat diakses di manapun selama ada koneksi ke internet. Prinsipnya dengan membandingkan UID yang terbaca dengan UID yang ada pada database lalu meng-update ke database tersebut. Selain itu RFID juga dapat dikombinasikan dengan metode *Capture Image* menggunakan kamera.



Gambar 2. 3 Arsitektur *Attendance based RFID, Image, & cloud*

(Sumber: Sharma. An automatic attendance monitoring system using RFID and IOT using Cloud, 2016)

2.5 MongoDB

Database NoSQL adalah database yang tidak menggunakan relasi antar tabel dan tidak menyimpan data dalam format tabel baku seperti layaknya *Relational Database* seperti SQL atau Oracle, PostgreSQL, MySQL dan SQL server [5].

Kelebihannya NoSQL tidak mengenal skema tabel yang kaku dengan format data yang kaku. NoSQL cocok untuk data yang tidak terstruktur, istilah singkat untuk fitur ini adalah *Dynamic Schema*. NoSQL juga menggunakan OOP dalam pengaksesan atau manipulasi datanya. NoSQL bisa menampung data yang terstruktur, semi terstruktur dan tidak terstuktur secara efesien dalam skala besar. [5]

MongoDB merupakan database *open-source* berbasis dokumen (*Document-Oriented Database*) yang awalnya dibuat dengan bahasa C++. MongoDB sendiri sudah dikembangkan oleh 10gen sejak Oktober 2007, namun baru dipublikasikan pada Februari 2009. Selain karena performanya 4 kali lebih cepat dibandingkan MySQL serta mudah diaplikasikan, karena telah tergabung juga sebagai modul PHP. [5]

Kelebihannya adalah performa yang ditawarkan MongoDB lebih cepat dibandingkan MySQL ini disebabkan oleh *mem-cached* dan format dokumennya yang berbentuk seperti JSON, dan masih banyak lagi kelebihan lainnya. MongoDB tersedia untuk platform Linux, Windows dan Mac. [5]

2.6 PyQt

PyQt menyatukan *framework cross-platform* Qt C++ dan Python *cross-platform interpreted*. Qt lebih dari sekadar toolkit GUI. Ini termasuk abstraksi soket jaringan, utas, Unicode, ekspresi reguler, basis data SQL, SVG, OpenGL, XML, peramban web yang berfungsi penuh, sistem bantuan, kerangka multimedia, serta koleksi kaya widget GUI. [7]

Kelas-kelas Qt menggunakan mekanisme sinyal/slot untuk berkomunikasi antara objek-objek yang bertipe aman tetapi longgar digabungkan sehingga mudah untuk membuat komponen perangkat lunak yang dapat digunakan kembali [7].

Qt juga mencakup Qt Designer, perancang GUI. PyQt dapat menghasilkan kode Python dari Qt Designer. Dimungkinkan juga untuk menambahkan kontrol GUI baru yang ditulis dengan Python ke Qt Designer. [7]

Python adalah bahasa berorientasi objek yang sederhana namun kuat. Kesederhanaannya membuatnya mudah dipelajari, tetapi kekuatannya berarti aplikasi besar dan kompleks dapat dibuat. Sifatnya yang ditafsirkan berarti bahwa programmer Python sangat produktif karena tidak ada siklus pengembangan kompilasi. [7]

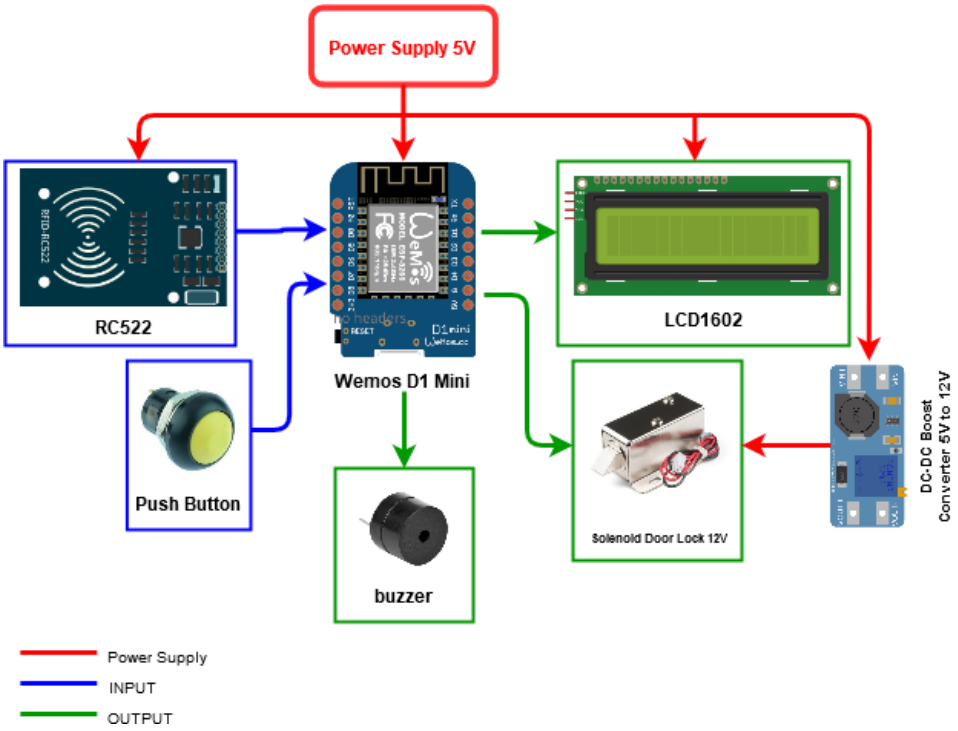
Sebagian besar kekuatan Python berasal dari rangkaian modul ekstensi yang komprehensif yang menyediakan berbagai fungsi termasuk server HTTP, parser XML, akses basis data, alat kompresi data dan, tentu saja, antarmuka pengguna grafis. Modul ekstensi biasanya diimplementasikan dalam Python, C atau C ++. Menggunakan alat-alat seperti SIP membuat modul ekstensi yang merangkum *library* C atau C ++ yang ada. Dengan cara ini, Python kemudian dapat menjadi lem untuk membuat aplikasi baru dari *library* yang sudah ada. [7]

PyQt menggabungkan semua keunggulan Qt dan Python. Seorang programmer memiliki semua kekuatan Qt, tetapi mampu mengeksploitasinya dengan kesederhanaan Python. [7]

III. METODOLOGI DAN BENTUK TUGAS AKHIR

3.1 Perancangan Hardware

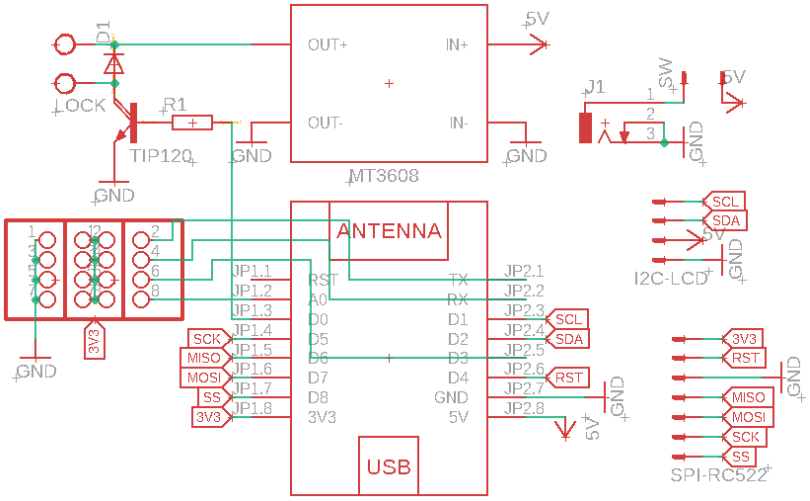
3.1.1 Blok Diagram Hardware



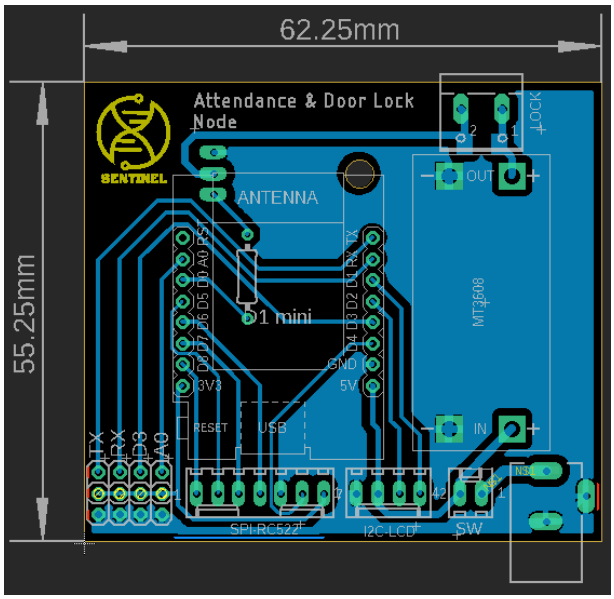
Gambar 3. 1 Blok Diagram Hardware *Node* Absensi

Pada hardware terdapat *push button* yang berfungsi untuk mengubah mode apakah ingin membuka pintu atau melakukan absensi. Untuk fitur tambahan yaitu pengunci pintu menggunakan solenoid dengan tegangan kerja 12 volt yang didapat dari modul DC-DC *boost converter* untuk menaikkan tegangan dari 5 volt menjadi 12 volt.

3.1.2 Skema dan Tata Letak Rangkaian

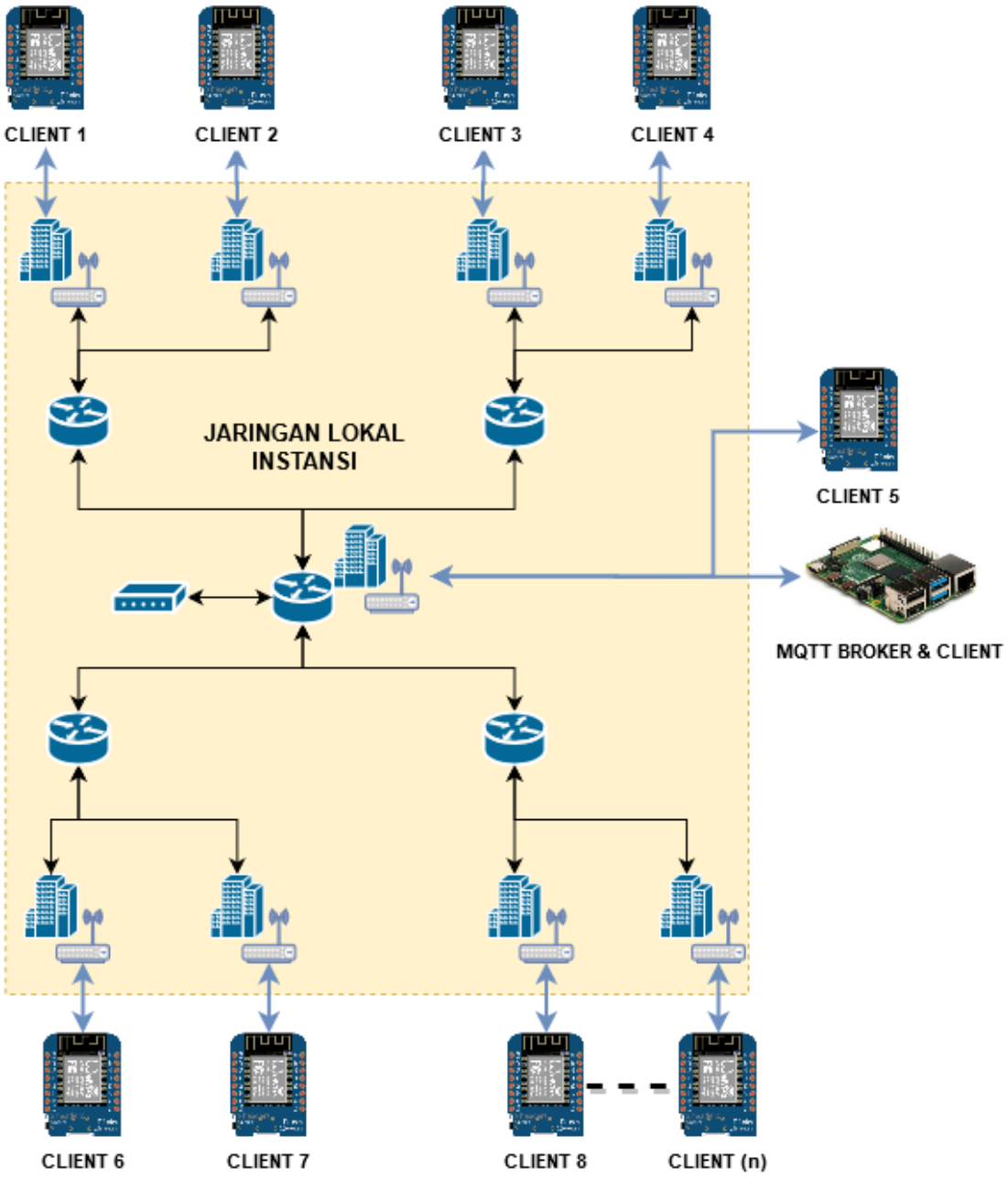


Gambar 3. 2 Skema Rangkaian *Node* Absensi



Gambar 3. 3 Tata Letak Rangkaian *Node* Absensi

3.2 Konfigurasi Jaringan



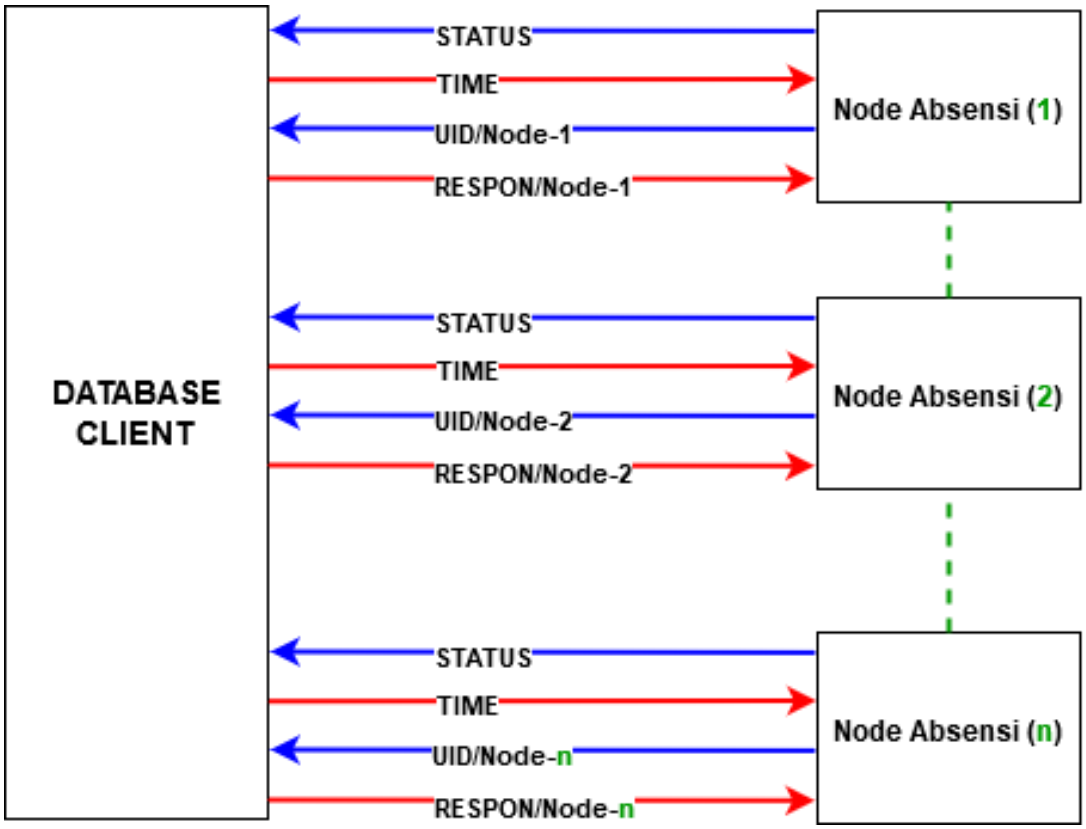
Gambar 3. 4 Konfigurasi jaringan sistem absensi

Pada Gambar 3.4 Raspberry Pi berfungsi sebagai MQTT *broker* atau *server* dan sekaligus sebagai *client* yang sudah terinstall *library* MOSQUITTO. Untuk masing-masing *node* absensi menggunakan Wemos D1 Mini sebagai MQTT *client*. Perangkat

broker maupun *client* akan dihubungkan ke *access point* terdekat. Setiap *client* dapat berada pada posisi yang jauh tergantung di mana karyawan akan melakukan absensi. Yang terpenting semuanya terkoneksi ke 1 jaringan yang sama. Untuk penelitian ini hanya digunakan 2 *node* absensi.

3.3 Perancangan Software dan Algoritma

3.2.1 Blok Diagram Algoritma



Gambar 3. 5 Blok Diagram Algoritma

Dari blok diagram 3.4 dapat dilihat bahwa tanda panah biru merupakan topik yang di-*subscribe* oleh *database client* atau di-*publish* oleh *node* absensi. Sedangkan tanda panah merah merupakan topik yang di-*publish* oleh *database client* atau di-*publish* oleh *node* absensi.

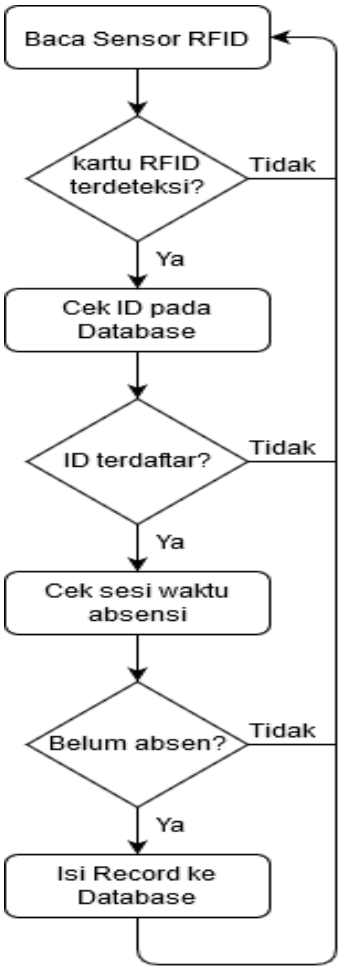
Client terdiri dari *database* berupa komputer yang akan digunakan untuk menyimpan dan mengolah *database* karyawan dan beberapa *node* absensi di banyak titik. (n) merepresentasikan ID *node* yang akan digunakan sebagai *client* ID dalam

protokol MQTT di setiap *node*. Pada *database client* juga terdapat GUI untuk mempermudah administrator mengecek status koneksi masing-masing *node*, menambahkan data karyawan baru, dan menghapus data dalam *database*.

Masing-masing *node* absensi akan mem-*publish* status koneksi ke *database client* dengan periode waktu tertentu pada topik "STATUS". Setiap *node* absensi menggunakan fitur MQTT, yaitu "*Last Will Message*" jika suatu waktu koneksi terputus. *Message* ini akan disimpan oleh *broker* dan akan diteruskan ke *subscriber* topik "STATUS" jika salah satu *publisher* dalam hal ini setiap *node absensi* tiba-tiba terputus koneksi dengan *broker*.

Bila *node* absensi mendeteksi kartu RFID seorang karyawan, UID dari kartu akan di-*publish* melalui topik "UID/Node-n". UID tersebut akan dicek oleh *database client* apakah UID tersebut terdaftar dalam *database* atau tidak. Respon akan di-*publish* oleh *databse client* ke topik "RESPON/Node-n".

3.2.2 *Flowchart* Sistem Absensi



Gambar 3. 6 *Flowchart* Sistem Absensi

Cara kerja sistem digambarkan pada Gambar 3.5. *Node* absensi akan mengirim setiap UID yang terbaca untuk selanjutnya dicek oleh server apakah UID terdaftar atau tidak. Selanjutnya server akan memverifikasi apakah UID yang terdaftar sudah melakukan absensi berdasarkan sesi waktu absensi saat ini. *Database* akan di-*update* bila setelah dilakukan verifikasi karyawan belum melakukan absensi. *Node* absensi akan menampilkan pesan ke LCD setelah menerima respon dari server. Setiap aktivitas yang terjadi pada *node* absensi akan dikirim dan disimpan di *log file* pada server.

IV. JADWAL PELAKSANAAN

No.	Kegiatan	Waktu Pelaksanaan (Bulan ke-)																			
		1				2				3				4				5			
1	Perancangan proposal dan konsep sistem/alat yang akan dibuat																				
2	Persiapan komponen dan perancangan, dan pembuatan hardware																				
3	Pembuatan GUI, integrasi database dan algoritma																				
4	Pengujian *hardware* dan *software*																				
5	Analisis, perekapan, dan pelaporan hasil pengujian																				

V. ANGGARAN BIAYA

No.	Nama Komponen	Justifikasi Pemakaian	Jumlah	Harga	Harga Keseluruhan
1	Modul Raspberry Pi 3B (second)	Sebagai *broker* dan *database*	1	Rp 335,000	Rp 335,000
2	MicroSD Sandisk Ultra A1 16GB	*Drive* penyimpanan pada Raspberry Pi	1	Rp 58,000	Rp 58,000
3	Wemos D1 Mini	Kontroler masing-masing *node* (mesin absensi)	2	Rp 35,000	Rp 70,000
4	Modul MFRC522	*Scanner kartu* RFID	2	Rp 20,000	Rp 40,000
5	Modul LCD1602 + I2C adaptor	Display untuk masing-masing *node*	2	Rp 25,000	Rp 50,000
6	Adaptor 5V 3A	Catu daya untuk modul Wemos dan Raspberry Pi	3	Rp 30,000	Rp 90,000

7	Boost Converter MT3608 2A	Untuk output tegangan ke solenoid di setiap *node* (opsional)	2	Rp 7,000	Rp 14,000
8	PCB (cutting, masking, & overlay)	Board sistem minimum setiap *node*	2	Rp 25,000	Rp 50,000
9	komponen lain	komponen elektronika pendukung lainnya	1	Rp 75,000	Rp 75,000
10	Cutting Acrylic	Casing penutup untuk setiap *node*	2	Rp 25,000	Rp 50,000
Total Biaya					Rp 832,000

DAFTAR PUSTAKA

[1]Antonić, A., M. Marjanović, P. Skočir, and I. P. Žarko. (2015). Comparison of the CUPUS Middleware and MQTT Protocol for Smart City Services. In 2015 13th International Conference on Telecommunications (ConTEL), 1–8. doi:10.1109/ConTEL.2015.7231225.

[2]Atmoko, Rachmad Andri. (2019). Dasar Implementasi Protokol MQTT Menggunakan Python dan NodeMCU. Blitar: Mokosoft Media.

[3]Bellavista, Paolo, Carlo Giannelli, and Riccardo Zamagna. (2017). The Pervasive Environment Sensing and Sharing Solution. Sustainability 9 (4): 585. doi:10.3390/su9040585.

[4]Fremantle, P., B. Aziz, J. Kopecký, and P. Scott. (2014). Federated Identity and Access Management for the Internet of Things. In 2014 International Workshop on Secure Internet of Things, 10–17. doi:10.1109/SIoT.2014.8.

[5]IdCloudHost. (2017, Juli 7). Mengenal Lebih Dekat Tentang Database MongoDB. Januari 23, 2019. https://idcloudhost.com/mengenal-lebih-dekat-tentang-database-mongodb/

[6]Light, R.A. (2017). Mosquitto: server and client implementation of the MQTT protocol. Journal of Open Source Software, 2(13), 265, doi:10.21105/joss.00265.

[7]Riverbank Computing Limited. What is Pyqt?. Januari 23, 2019. https://www.riverbankcomputing.com/software/pyqt/intro

[8]Sasaki, Y., Yokotani, T., & Mukai, H. (2019). MQTT over VLAN for Reduction of Overhead on Information Discovery. 2019 International Conference on Information Networking (ICOIN). doi:10.1109/icoin.2019.8718125.

[9]Schulz, M., F. Chen, and L. Payne. (2014). Real-Time Animation of Equipment in a Remote Laboratory. In 2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV), 172–76. doi:10.1109/REV.2014.6784247.

[10]Sharma, T., & Aarthy, S. L. (2016). An automatic attendance monitoring system using RFID and IOT using Cloud. 2016 Online International Conference on Green Engineering and Technologies (IC-GET). doi:10.1109/get.2016.7916851.

[11]Thangavel, D., X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan. (2014). Performance Evaluation of MQTT and CoAP via a Common Middleware. In 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 1–6. doi:10.1109/ISSNIP.2014.6827678.