PENGEMBANGAN SISTEM MONITORING KUALITAS UDARA TERINTEGRASI IOT CLOUD UNTUK DITERAPKAN PADA SMART CITY

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Hermawan Heri Wijaya
NIM: 125150300111014

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

PENGESAHAN

PENGEMBANGAN SISTEM MONITORING KUALITAS UDARA TERINTEGRASI CLOUD IOT UNTUK DITERAPKAN PADA SMART CITY

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Hermawan Heri Wijaya
NIM: 125150300111014

Skripsi ini telah diuji dan dinyatakan lulus pada

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing 1	Dosen Pembimbing 2
Sabriansyah Rizqika Akbar, S.T, M.Eng	Mochammad Hannats Hanafi I., S.ST, M.T
NIP. 19820809 201212 1 004	NIK. 201405 881229 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 13 juli 2016

Hermawan Heri Wijaya

NIM: 125150300111014

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Pengembangan sistem monitoring kualitas udara terintegrasi IoT *Cloud* untuk diterapkan pada *Smart City*" ini dapat terselesaikan.

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

1. Kedua orang tua, yang selalu memberikan dukungan doa dan motivasi sehingga penulis dapat menyelesaikan skripsi ini.
2. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku dosen pembimbing 1 dan Bapak Mochammad Hannats Hanafi I., S.ST, M.T. selaku dosen pembimbing 2 skripsi yang selalu sabar dalam membimbing penulis dalam menyelesaikan skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku ketua Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Bapak Ir. Sutrisno, M.T. selaku Ketua Program Teknologi Informasi dan Ilmu Komputer.
5. Segenap Bapak dan Ibu dosen Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas segenap ilmu pengetahuan dan perhatian yang diberikan.
6. Kepada teman kosan "DPK7" Rusyadi, Tama, Rizki Eka, Ricky Prasetyo, Meidy, Cahya, Mukhlis yang selalu nge-*bully* agar lebih termotivasi dalam penyelesaian skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 27 juni 2016

Penulis

Hermawan Heri Wijaya

ABSTRAK

Pada penelitian ini, penulis mengembangkan sistem monitoring kualitas udara dimana sistem ini menginformasikan kualitas udara disekitarnya dan memberitahukan hasil pembacaan ke masyarakat melalui sosial media. Sistem secara umum terdiri atas 2 buah sensor udara, yaitu *Air Quality* dan *Carbon Dioxide*, yang terhubung ke jaringan internet melalui transmitter ESP8266 dengan mikrokontroler Arduino Mega 2560 yang bertujuan untuk mengupload data ke IoT *Cloud* dan update status berupa *twitter feed* sebagai sarana informasi. Pengujian dilakukan dengan 3 cara yaitu konektivitas, fungsionalitas dan *delay*. Uji konektivitas membuktikan bahwa *transmitter* dapat terhubung ke *Access Point* dan jaringan internet. Uji fungsionalitas membuktikan bahwa sensor dapat membaca udara dan mengambil data. Dan yang terakhir adalah pengujian *delay*, dimana data mulai dari *booting* alat hingga data sampai di IoT *Cloud* dan sosial media akan diperhitungkan lama proses dan pengiriman. Dari hasil perhitungan dalam pengujian yang dilakukan didapatkan bahwa data akan terkirim sebanyak 24 data dalam 1 jam, namun dari 5 percobaan 4 diantaranya hanya mengirim rata-rata 22 data dalam 1 jam, walaupun selisih jumlah data lumayan besar, namun tujuan penelitian sudah sesuai dengan apa yang diharapkan.

Kata kunci : Monitoring, Kualitas Udara, IOT, Sosial Media.

ABSTRACT

In this research, the authors developed air quaily monitoring where the system informs the surrounding air quality readings and notify the results to the public through social media. The system generally consists of two sensors, namely Air Quality and Carbon Dioxide, which is connected to the Internet network via the transmitter ESP8266 with microcontroller Arduino Mega 2560 that aims to upload data to the IoT Cloud and tweeter feed in the form of status updates as a means of information. Testing is done in 3 ways, namely connectivity, functionality and delay. Connectivity test proved that the transmitter can be connected to the Access Point and Internet networks. Functionality test proved that the air sensor can read and retrieve data. And The Third way is the test delay, where data from booting tool to IoT *data up in the Cloud and social media would be taken into account long process and delivery. From results of calculations in tests conducted found that the data will be sent as many as 24 data in 1 hour, but from 5 trials four of them are only sent an average of 22 data in 1 hour, although the difference between the amount of data is quite large, but the purpose of the study is in conformity with what which are expected.*

Keywords: Monitoring, Air Quality, IoT, Social Media.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Smart City merupakan pengembangan dan pengelolaan kota dengan pemanfaatan teknologi informasi dan komunikasi (TIK) untuk mengetahui (*sensing*), memahami (*understanding*) dan mengendalikan (*controlling*) berbagai sumber daya yang ada di dalam kota dengan lebih efektif dan efisien untuk memaksimalkan pelayanan kepada warganya serta mendukung pembangunan berkelanjutan. atau lebih mudahnya, Kota Cerdas adalah integrasi informasi secara langsung dengan masyarakat perkotaan (Aditya, et al., 2015).

Saat ini kota-kota besar di Indonesia sudah tidak memberikan tempat yang nyaman dan sehat bagi penduduknya. Industri yang semakin berkembang di Indonesia diiringi dengan meningkatnya jumlah kendaraan serta meningkatnya jumlah penduduk membuat kota-kota tersebut berada dalam kondisi pencemaran lingkungan yang mengkhawatirkan. Data WHO memasukkan kota-kota besar di Indonesia dalam pemantauan tingkat polusi udara yang melebihi ambang batas maksimal kadar partikel dalam udara yang di rekomendasikan WHO (Kompas, 2013). Tetapi hal ini tidak didukung dengan adanya sistem monitoring kondisi polusi sehingga kualitas hidup penduduk perkotaan maupun kondisi kota itu sendiri semakin buruk dari waktu ke waktu (Yusniwarti, 2003). Menurut buku *Climate ChangeMyths & Realistic* karya Dr.S. Jeevananda Reedy polutan primer yang dihasilkan oleh manusia berupa sulfur oksida (SOx), nitrogen oksida (NOx), Karbon Dioksida (CO_2), materi Partikulat (PM) seperti asap dan debu, logam beracun seperti timah, kadmium dan tembaga, Chlorofluorocarbon (CFC), Amoniak (NH3) dari proses pertanian, bau, seperti dari sampah, limbah dan proses industri, radioaktif oleh ledakan nuklir atau radon. Melihat fakta tersebut, masyarakat perlu mengetahui informasi seberapa layak suatu daerah ditinggali sehingga meskipun mereka hidup diperkotaan mereka tetap bisa menjaga lingkungannya agar dapat terbebas dari segala macam polusi.

Dalam mengimplementasikan teknologi *Smart City*, kita dapat memanfaatkan Jaringan sensor nirkabel / WSN untuk melakukan monitoring kualitas udara. WSN terbentuk dari sekumpulan node yang berukuran kecil yang tersebar secara kolektif membentuk *Wireless Sensor Network* (WSN) dan memberikan informasi dari data, teknologi yang dibutuhkan untuk merealisasikan era baru bagi teknologi penginderaan di mana-mana. WSN dapat digunakan dengan perangkat yang bekerja melalui frekuensi gelombang seperti NrF ataupun Wi-fi namun kebanyakan WSN saat ini masih menggunakan NrF atau gelombang radio dimana komunikasi nantinya menjadi kurang stabil apabila diterapkan di daerah perkotaan yang terdapat banyak bangunan tinggi dan tebal (Gregory O'Hare, 2015). Perangkat yang mendukung WSN dan dapat bekerja dengan menggunakan jaringan wifi salah satunya adalah ESP, ESP adalah perangkat *transceiver* berukuran kecil yang dipergunakan untuk komunikasi dari mikrokontroler ke perangkat jaringan wifi seperti *Access Point*. ESP bekerja pada frekuensi standar

sesuai dengan ketetapan IEEE802.11bgn, dengan *Radio Frequency* 2.4GHz pada *transmitter* dan *receiver*.

Pada penelitian yang dilakukan oleh Aditya, 2015 adalah mendefinisikan secara sistematis bagaimana *Smart City* beroperasi dan mengambil data yang dihasilkan dari sensor. Layanan komunikasi yang digunakan dalam infrastruktur *Smart City*, yaitu 3G (*3rd generation*), LTE (*Longterm evolution*), Wi-Fi (*Wireless fidelity*), WiMAX (*worldwide interoperability for microwave access*), ZigBee, CATV (*cable television*) dan komunikasi satelit. Setiap komunikasi selalu melalui perantara, baik berupa tulisan, suara, video, maupun Sosial Media. Berdasarkan data statistik bulan agustus 2015, didapatkan bahwa 31,7 miliar penduduk di dunia menggunakan internet dan 80% dari mereka terhubung ke Sosial Media (Kemp & Simon, 2015). Berdasarkan data statistik tersebut dapat diperkirakan apabila Smary City dioperasikan, maka akan sangat terlihat manfaatnya bagi masyarakat luas.

Selain dari WSN, IoT juga dapat dimanfaatkan sebagai salah satu cara untuk mewujudkan *Smart City.* Internet *of Things* (IoT) adalah sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus-menerus, Seperti kemampuan *Remote Control*, *File Sharing*, dan sebagainya, termasuk pada benda-benda di dunia fisik. Seperti Bahan pangan, elektronik, peralatan apa saja, koleksi, termasuk benda hidup, yang semuanya tersambung ke jaringan lokal dan global melalui *Embedded System* dan selalu "on". Pada hakekatnya, benda Internet atau *Internet of Things* mengacu pada benda yang dapat di identifikasikan secara unik sebagai representasi virtual dalam struktur berbasis Internet. Bahkan sekarang IoT sudah berkembang dengan munculnya IoT *Cloud*, yaitu *webserver* khusus untuk Embedded System berbasis Internet.

Pada penelitian ini, akan dilakukan pengembangan sistem monitoring kualitas udara yang dapat membaca kualitas udara dan karbon dioksida yang ada di udara serta meng-upload hasil baca sensor ke IoT *Cloud* yaitu *Thingspeak* dan sekaligus melakukan publikasi ke Sosial Media Twitter. Sistem secara umum terdiri atas 2 buah sensor udara, yaitu Air Quality dan Carbon Dioxide, yang terhubung ke jaringan internet melalui transmitter ESP8266 dengan mikrokontroler Arduino Mega 2560 yang bertujuan untuk mengupload data ke IoT *Cloud* dan update status berupa twitter *feed* sebagai sarana informasi

1.2 Rumusan masalah

Perumusan masalah yang akan dibahas pada penelitian ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan sensor monitoring kualitas udara dan sensor gas CO_2 sebagai sensor node?
2. Bagaimana mengirimkan data dari sensor node menuju ke IoT *Cloud* melalui media komunikasi wifi?

3. Bagaimana mengirimkan informasi dari IoT *Cloud*/Node ke sosial media Twitter?
4. Bagaimana waktu respon yang diberikan sistem dari awal hingga informasi *ter-publish* melalui sosial media?

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Menyusun Sensor Monitoring Kualitas Udara dan Sensor gas CO_2 dengan metode IoT agar membentuk sebuah Sistem Monitoring Kualitas Udara.
2. Sistem ini menerapkan fungsi API *Key* yang tersedia pada IoT *Cloud* yang terintegrasi sosial media twitter sebagai penghubung antara Mikrokontroler ke IoT *Cloud* dan Sosial Media.
3. Melihat hasil baca data pada Sistem Monitoring Kualitas Udara melalui IoT *Cloud* dan Sosial Media.
4. Melakukan analisis berapa lama waktu respon yang diberikan oleh sistem dari awal hingga *ter-publish* di IoT *Cloud* dan Sosial Media.

1.4 Manfaat

Manfaat yang diharapkan dari hasil penelitian ini adalah

1. Bagi bidang pelayanan masyarakat penelitian ini dapat membantu menyebarkan informasi bagi masyarakat yang ingin mengetahui kondisi udara disekitarnya.
2. Bagi bidang pemerintah penelitian ini dapat membantu realisasi *Smart City* di era canggih sekarangi ini.
3. Sebagai gambaran atau acuan bagi pemerintah, masyarakat dan pengembang apabila ingin mengembangkan sistem ini.

1.5 Batasan masalah

Adapun batasan masalah pada penelitian ini agar tidak menyimpang dari perumusan masalah yang sudah ditulis adalah sebagai berikut:

1. Pengiriman data yang dilakukan adalah dari 1 buah node yang terhubung dengan internet melalui jaringan Wi-Fi.
2. Senyawa yang diukur pada Udara adalah CO_2 (Karbon Dioksida/Zat Asam Arang) dan Zat Berbahaya.
3. Metode pengkoneksian Mikrokontroller dengan Sosial media adalah dengan menggunakan fitur yang ada pada *Cloud* IoT.
4. Tidak membahas mengenai akurasi data yang didapat sensor.
5. Tidak melakukan *Proses Device Control* dan *Alerts.*
6. Sosial Media yang digunakan dalam penelitian ini adalah Twitter karena twitter merupakan situs *text based* yang terhitung ringan dan memiliki banyak pengguna.

7. *Cloud* IoT yang digunakan adalah Thingspeak karena lebih mudah dan simple, selain itu memiliki banyak fitur seperti matlab, tweethings dan masih banyak lainnya.
8. Sistem ini masih dalam tahapan PURWARUPA, sehingga belum dapat didistribusikan secara umum.

1.6 Sistematika pembahasan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut.

BAB I : **Pendahuluan**

Mengeruakan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

BAB II : **Kajian Pustaka Dan Dasar Teori**

Menguraikan kajian pustaka dan dasar teori yang mendasari teknologi *Smart City*, *Wireless Sensor Network*, dan beberapa aspek yang digunakan dalam penelitian.

BAB III : **Metode Penelitian dan Perancangan**

Menguraikan tentang metode dan langkah kerja yang terdiri dari studi literatur, analisis kebutuhan simulasi, perancangan sistem, implementasi dan analisis serta pengambilan kesimpulan.

BAB IV : **Implementasi**

Menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

BAB V : **Pengujian dan Analisis**

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan

BAB VI : **Penutup**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian, serta saran-saran untuk pengembangan lebih lanjut.

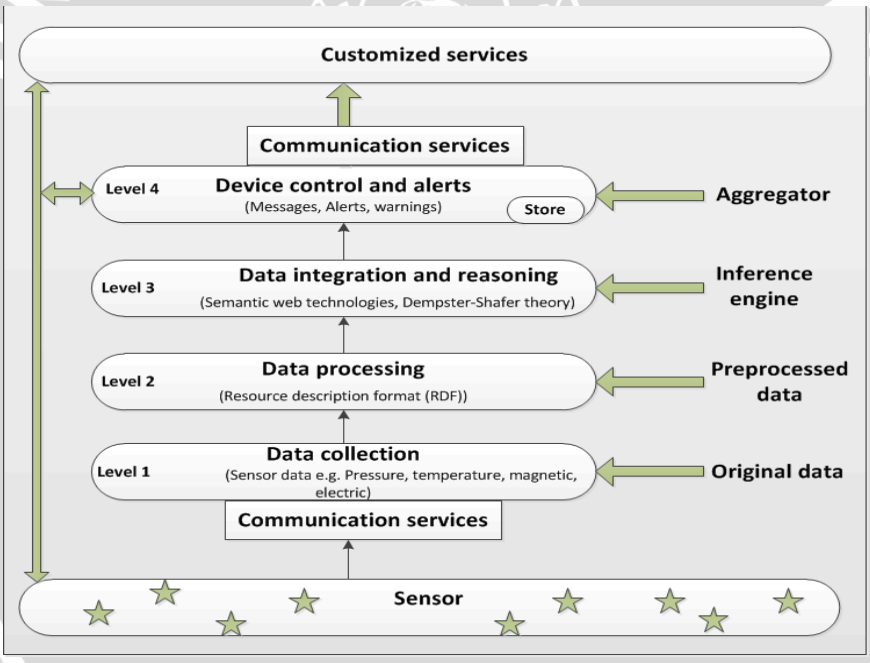
BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan menjelaskan tentang dasar teori metode dan algoritma yang akan digunakan dalam penelitian. Tinjauan pustaka membahas penelitian yang telah ada dan diusulkan yaitu Monita dan *Smart City Architecture*. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan berupa *Smart City*, WSN, Mikrokontroler, Sensor dan API.

2.1 Tinjauan Pustaka

Tinjauan pustaka berisi tentang penelitian yang sudah dilakukan sebelumnya dan berkaitan dengan penelitian yang akan dilakukan. Tujuan dengan adanya tinjauan pustaka yang sudah ada agar dapat dijadikan perbandingan dalam pelaksanaan penelitian.

Penelitian yang dilakukan Aditya Gaur, 2015 adalah bagaimana IoT diterapkan dalam *Smart City* serta arsitektur dari *Smart City* yang dikategorikan dalam Multi-level seperti pada Gambar 2.1



Gambar 2.1 Multi-level *Smart City* Architecture

Sumber: (Aditya, et al., 2015)

Level 1 – Data Collection

Mengambil data dengan menggunakan sensor dan disimpan untuk langkah selanjutnya

Level 2 – Data Processing

Mengubah format data yang terkumpul, contohnya kedalam bentuk RDF atau Resourde Description Framework untuk bertukar informasi melalui web

Level 3 – Data Integration and Reasoning

Melakukan operasi penggabungan, perbandingan, dan lainnnya dari keseluruhan data yang terkumpul dan disajikan dalam bentuk Obyek dan Data

Level 4 – Device Control and Alerts

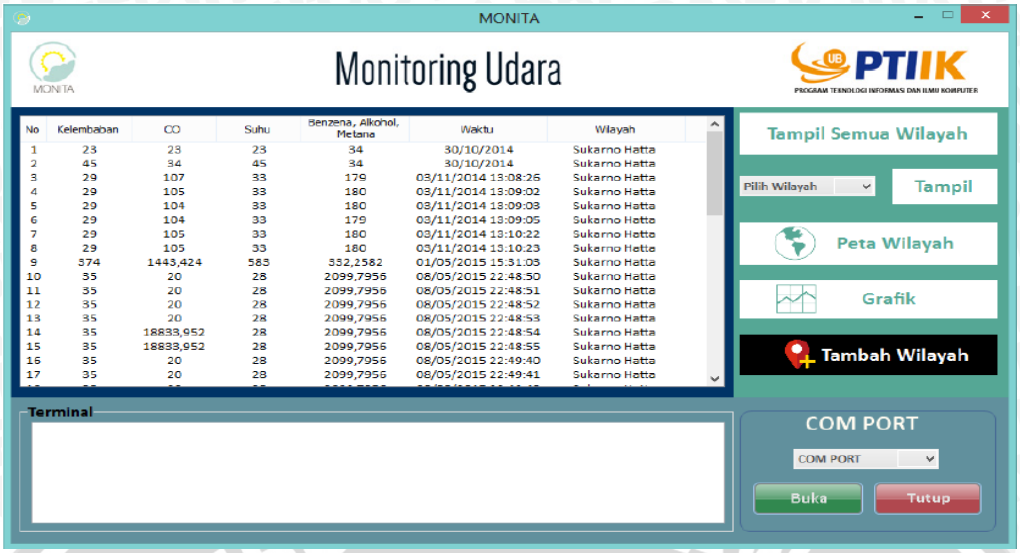
Data dari Level 3 dapat diolah menjadi aksi berupa Input/Output, Message, Alert, or Warning.

Penelitian yang dilakukan oleh Murat adalah pengembangan teknologi WSN sebagai Smart Environtment, seperti monitoring cuaca, besar medan magnetis, besar radiasi, lampu otomatis, mendeteksi tempat padat penduduk, pembuangan limbah, bercocok tanam, dan masih banyak lainnya.

WSN dapat diimplementasikan untuk mengetahui banyaknya Karbon Monoksida, Karbon Dioksida, Oksigen, Hidrogen, Ozon, karbon Hydro, Sulphur Dioxide, Nitric Oxide, Nitrogen, Partikel, Suhu, Kelembaban, Radiasi (Beta, Gama), Radiasi (UV), Bidang elektromagnetik, Brightness, Penyesuaian Kecerahan, Gerak Deteksi, Microphone, Range Finder, Sensor Metana dalam konteks Smart Environtment.

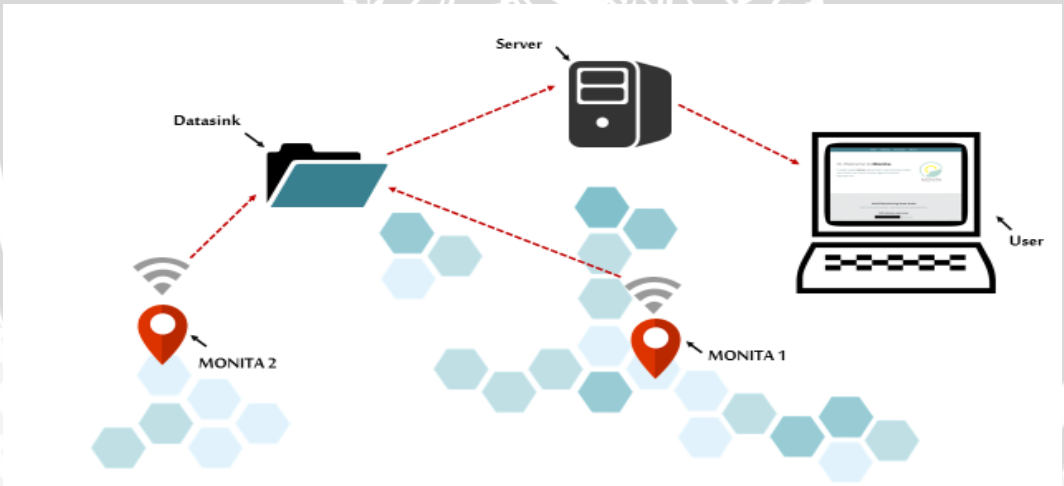
Suhu udara, udara Kelembaban, Suhu Tanah, Kelembaban Tanah, Tekanan Udara, Radiasi Matahari, Ultra violet Radiasi, Kecepatan Angin, Arah Angin, Light, Nosie, Nitrogen, Fosfor, salinitas, Listrik Konduktivitas, fotosintesis Aktif Radiasi, Gerak dan Tinggalkan sensor basah dalam konteks Smart Agricultural dan RS-232, RS-485, RS-422, konverter CAN-BUS untuk Integrated Electricity (O'Hare, et al., 2015)

Monitoring Udara (Monita) adalah projek milik penulis yang sebelumya sudah pernah dikerjaan, yaitu berupa Monitoring gas CO_2 dan Kualitas Udara dengan teknologi WSN dan database MySql dengan GUI berupa aplikasi computer hasil pemrograman menggunakan Delphi7, data yang dihasilkan masih data mentah yang belum diapa-apakan dan hanya disimpan dalam database untuk ditampilkan pada aplikasi (Heri, et al., 2014).



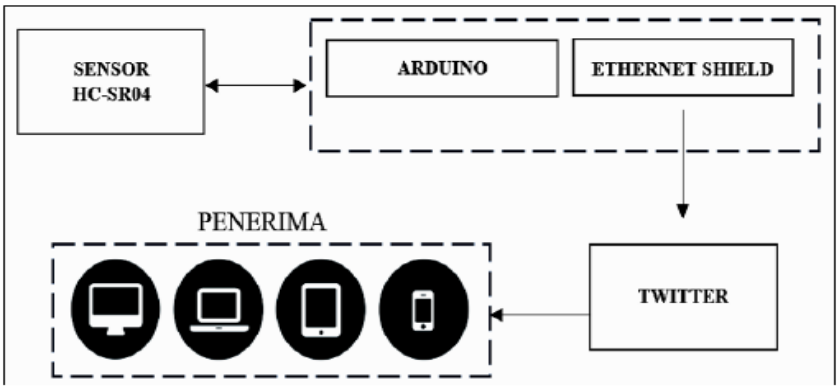
Gambar 2.2 Monitoring udara dengan aplikasi Delphi7

Monita menggunakan teknologi WSN, yaitu node sensor yang terpisah jarak namun masih dapat berkomunikasi demi mengirimkan data ke dalam datasink, yang kemudian datasink menyimpan database tersebut kedalam MySql pada server, dan menampilkan hasilnya melalui GUI aplikasi hasil dari pemrograman pada Delphi7



Gambar 2.3 Cara Kerja MONITA dengan Metode WSN

Penelitian yang dilakukan oleh Devin adalah perancangan prototype sistem informasi ketinggian air melalui twitter sebagai sistem peringatan dini akan bahaya banjir, Sehingga masyarakat dapat menyelamatkan perabotan dan barang berharga lainnya lebih cepat sebelum wabah banjir melanda pemukiman (Davin,2015).



Gambar 2.4 Cara Kerja Sistem Informasi Ketinggian Air menggunakan Twitter

Dengan menggunakan sensor HC-SR04 atau sensor ultrasonik, tinggi air dapat dipantau dan hasil baca ketinggian tersebut dikirimkan melalui Ethernet shield dengan TOKEN agar dapat menggunan OAuth (*Open Authentication*), penerima informasi adalah masyarakat yang menggunakan twitter dan mem-follow akun twitter sistem tersebut.

Penelitian yang penulis lakukan adalah membangun sistem monitoring kualitas udara berdasarkan Zat Berbahaya di udara (*NH3*, *NOx*, *alcohol*, *Benzene*, *smoke*, *etc*.) dan CO_2 yang terkandung di udara dan membagikan informasi tersebut melalui Sosial Media berupa twitter serta menganalisa lama waktu yang diperlukan untuk mengakuisisi data hingga publikasi data.

Berbeda dari penetilian sebelumnya yang datanya hanya dapat diliha secara personal, disini data dipublikasikan agar seluruh masyarakat dapat melihat data tersebut.

2.2 Dasar Teori

Beberapa dasar teori diperlukan agar penelitian yang dilakukan dapat membuahkan hasil yang sesuai dengan tujuan peneltian

2.2.1 Smart City

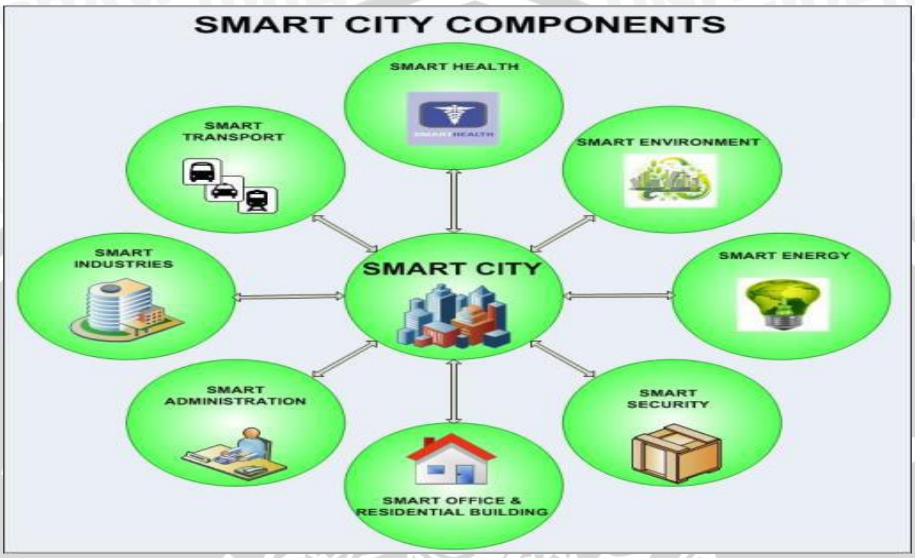
Smart City adalah sebutan untuk daerah perkotaan yang hampir seluruh benda di kota tersebut terhubung dengan Mikrokomputer dengan konsep *Internet of Things* (IoT), seperti lemari pendingin yang dipasangi internet agar jika isi lemari pendingin sudah mulai menipis, dengan sendirinya lemari pendingin mengirim permohonan stock barang ke produsen yang tersedia, atau contoh lainnya seperti *Smart Home* dimana lampu akan menyala secara otomatis apabila matahari sudah tenggelam (Aditya, et al., 2015).

Smart city direalisasikan dengan berbagai cara, salah satunya dengan metode *Wireless Sensor Network* (WSN), WSN adalah sumber utama informasi yang heterogen bagi *Smart City*, dengan bantuan teknologi jaringan nirkabel sekarang

ini bukanlah hal yang tidak mungkin untuk merealisasikan *Smart City* di daerah perkotaan.

2.2.1.1 Penyusun Utama *Smart City*

Secara *Verbal*, gambaran *Smart City* adalah kota pintar, namun jika dibayangkan dan digambarkan dalam bentuk *Visual*, *Smart City* dengan pengecualian kendaraan bermotor dapat dilihat pada Gambar 2.5



Gambar 2.5 Komponen *Smart City*

Seperti ditunjukkan dalam Gambar 2.5, kita membayangkan unsur-unsur utama dari arsitektur Kota Ceras untuk menjadi kesehatan pintar, lingkungan cerdas, energi pintar, keamanan cerdas, kantor cerdas dan bangunan tempat tinggal, administrasi cerdas, transportasi cerdas dan industri pintar. Node sensor dikerahkan di setiap domain Pintar Kota menyediakan sumber data primer untuk pembangkit informasi heterogen. Informasi yang dihasilkan melalui node sensor dikumpulkan menggunakan layanan komunikasi yang ada. Sebagai contoh, penggunaan jaringan satelit untuk perangkat GPS, layanan seluler seperti GSM / 3G / 4G untuk ponsel pintar dan penggunaan internet untuk PC dan perangkat navigasi lainnya untuk pengumpulan data mentah. Data yang terkumpul kemudian diolah dan dianalisis menggunakan teknologi web semantik dan aturan kombinasi Dempster-Shafer. Fokusnya adalah pada penggelaran arsitektur pada platform awan untuk digunakan sebagai *Software as a Services* (SaaS) atau Perangkat lunak sebagai layanan (Aditya, et al., 2015).

2.2.2 *Wireless Sensor Network* (WSN)

Wireless Sensor Network (WSN) merupakan kumpulan dari beberapa device yang digunakan untuk merubah besaran fisis menjadi besaran elektrik, yang saling terhubung satu dengan yang lain sehingga mampu saling berkomunikasi dan bertukar data, dan membentuk sebuah jaringan yang terhubung tanpa menggunakan kabel / nirkabel. Pada WSN ada istilah mote, yang merujuk ke sebuah node pada WSN. Tiap mote pada WSN sendiri nantinya akan terhubung ke

mote yang lain, maupun terhubung ke Base Station ataupun Data Sink, sehingga data bisa dimonitoring maupun diolah oleh User.

Mote pada WSN sendiri mempunyai beberapa bagian, yaitu:

1. Sensing Unit

   Merupakan bagian yang digunakan untuk mengambil data dari lingkungan. Jika data berupa data analog, maka data akan dirubah dahulu ke data digital menggunakan ADC (Analog to Digital Converter).

2. Processing Unit

   Merupakan bagian yang digunakan untuk mengolah data yang diambil oleh sensing unit, lalu meneruskanya ke actuator ataupun ke mote lain. Pada processing unit juga terdapat storage yang bisa digunakan untuk menyimpan data sementara sebelum diolah atau diteruskan lagi.

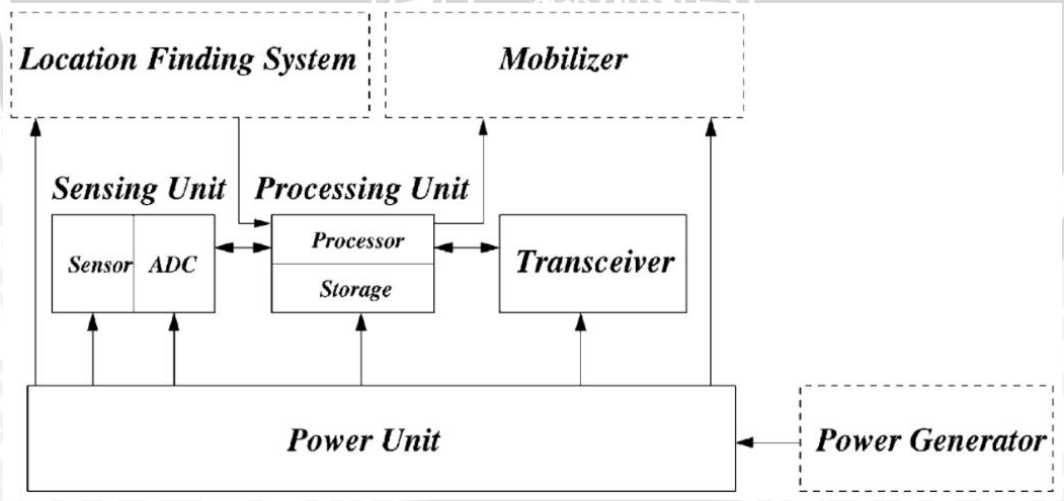
3. Transceiver

   Bagian yang bertugas untuk menerima data dari node lain, dan mengirimkan data dari processing unit ke mote yang lain.

4. Power Unit

   Bagian yang akan mendistribusikan energy yang akan digunakan oleh Sensing Unit, Processing Unit, maupun oleh Transceiver.

Selain 4 komponen utama diatas, pada mote juga bisa ditambahkan beberapa komponen tambahan yang bersifat optional, misalnya *power generator* yang digunakan untuk sumber energi, lalu *mobilizer* untuk mobilisasi mote, dan lain lain. Berikut ini diagram mote WSN secara umum (Ali & Shahzad, 2012).



Gambar 2.6 Struktur node pada WSN

2.2.3 Internet of Things

Dengan semakin berkembangnya infrastruktur internet, maka kita menuju babak berikutnya, di mana bukan hanya smartphone atau komputer saja yang dapat terkoneksi dengan internet. Namun berbagai macam benda nyata akan

terkoneksi dengan internet. Sebagai contohnya dapat berupa mesin produksi, mobil, peralatan elektronik, peralatan yang dapat dikenakan manusia (*wearables*), dan termasuk benda nyata apa saja yang semuanya tersambung ke jaringan lokal dan global menggunakan sensor dan atau aktuator yang tertanam. Di dunia bidang IT, konsep ini telah dikenal dengan istilah "*Internet of Things*" atau dikenal dengan singkatan IOT. Suatu perangkat keras biasanya tertanam dalam berbagai macam benda nyata tersebut sehingga benda tersebut dapat tersambung dengan internet.

2.2.4 Internet of Things Cloud

IoT *Cloud* adalah sebuah layanan pada webserver tertentu yang menyediakan database dan memisahnya kedalam bentuk channel/kanal sehingga data pada suatu sistem dapat disimpan secara khusus di channel/kanal tertentu dan terpisah, sehingga data tidak rancu dan juga lebih menghemat biaya dan waktu karena tidak harus mengutak atik database dan juga mengkoneksikan database local ke internet, karena semuanya sudah tersedia di IoT *Cloud* (ThingSpeak, 2016).

2.2.5 Mikrokontroler

Mikrokontroler merupakan komputer didalam chip yang digunakan untuk mengontrol peralatan elektronik, yang menekankan efisiensi dan efektifitas biaya. Secara harfiah dapat disebut sebagai "pengendali kecil" dimana sebuah sistem elektronik yang sebelumnya banyak memerlukan komponen-komponen pendukung seperti IC TTL dan CMOS dapat direduksi/diperkecil dan akhirnya terpusat serta dikendalikan oleh mikrokontroler ini.

Mikrokonktroler digunakan dalam produk dan alat yang dikendalikan secara automatis, seperti sistem kontrol mesin, *remote control*, mesin kantor, peralatan rumah tangga, alat berat, dan mainan. Dengan mengurangi ukuran, biaya, dan konsumsi tenaga dibandingkan desain menggunakan mikroprosesor memori dan alat *input output* yang terpisah, kehadiran mikrokontroler membuat kontrol elektrik untuk berbagai proses menjadi lebih ekonomis. (Arduino, 2016)

2.2.5.1 Arduino Mega 2560

Arduino Mega 2560 adalah board Arduino yang merupakan perbaikan dari board Arduino Mega sebelumnya. Arduino Mega awalnya memakai chip ATmega1280 dan kemudian diganti dengan chip ATmega2560, oleh karena itu namanya diganti menjadi Arduino Mega 2560. Pada saat tulisan ini dibuat, Arduino Mega 2560 sudah sampai pada revisinya yang ke 3 (R3). Berikut spesifikasi Arduino Mega 2560 R3.

Gambar 2.7 Arduino Mega

Spesifikasi Arduino Mega dapat dilihat pada tabel 2.1 berdasarkan (Arduino, 2016).

Tabel 2.1 Spesifikasi Arduino Uno

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB

2.2.5.2 ESP8266

Modul WiFi ini dapat menyambungkan rangkaian elektronika nirkabel karena modul elektronika ini menyediakan akses ke jaringan WiFi secara transparan dengan mudah melalui interkoneksi serial (UART RX/TX) modul WiFi ini bekerja dalam jaringan 802.11b/g/n.

Keunggulan utama modul ini adalah tersedianya mikrokontroler RISC (Tensilica 106µ Diamond Standard Core LX3) dan Flash Memory SPI 4 Mbit Winbond W2540BVNIG terpadu.

Gambar 2.8 ESP 8266

Spesifikasi konsumsi daya ESP 8266 dapat dilihat pada Tabel 2.2 berdasarkan (NURDs, 2015).

Tabel 2.2 Spesifikasi Konsumsi Daya ESP 8266

Mode	Typ	Unit
Mengirimkan 802.11b, CCK 1Mbps, Pout = + 19.5dBm	215	mA
Mengirimkan 802.11b, CCK 11Mbps, Pout = + 18.5dBm	197	mA
Mengirimkan 802.11g, OFDM 54Mbps, Pout = + 16dBm	145	mA
Mengirimkan 802.11n, MCS7, Pout = + 14dBm	135	mA
Menerima 802.11b, paket panjang = 1.024 byte, 80dBm	60	mA
Menerima 802.11g, paket panjang = 1.024 byte, 70dBm	60	mA
Menerima 802.11n, paket panjang = 1.024 byte, -65dBm	62	mA
Siaga	0.9	mA
Tidur nyenyak	10	uA
Modus hemat daya DTIM 1	1.2	mA
Modus hemat daya DTIM 3	0.86	mA
Jumlah penutupan	0.5	uA

2.2.6 Sensor

Sensor yang akan digunakan dalam pengembangan sistem *Smart City* ini antara lain sebagai berikut.

2.2.6.1 *MG-811 CO_2 Gas Sensor*

CO_2 *Gas Sensor* Modul ini dirancang untuk memungkinkan mikrokontroler untuk menentukan kapan tingkat gas Karbon Dioksida ditetapkan telah tercapai atau terlampaui. Modul ini dapat digunakan untuk menentukan kadar karbon dioksida yang terdapat pada udara. Modul ini berbasiskan sensor MG-811 yang

mampu melakukan pendeteksian gas karbon dioksida dengan range 1 - 10000 ppm.

Gambar 2.9 MG-811 CO_2 *Gas Sensor*

Spesifikasi dari MG-811 dapat dilihat pada Tabel 2.3 berdasarkan (Digiware, 2015).

Tabel 2.3 Spesifikasi MG-811

Tegangan kerja	5 VDC.
Target gas	karbon dioksida (CO_2).
Range deteksi	350 - 10000 ppm.
Antarmuka	UART TTL, I2C.
Lainnya	- Menggunakan ADC 10-bit untuk konversi data analog dari sensor. - Memiliki output berupa data digital dengan nilai 0 - 1023 (hasil konversi ADC). - Terdapat 1 buah variable resistor untuk pengaturan nilai threshold secara manual. - Disediakan beberapa jumper untuk konfigurasi pull-up I2C, resistor beban, serta variable resistor threshold.

2.2.6.2 *MQ-135 Air Quality*

MQ-135 Air Quality sensor adalah modul sensor gas yang dapat digunakan untuk menentukan kadar konsentrasi gas-gas berbahaya dalam udara. Modul ini berbasiskan sensor MQ-135, yaitu sensor yang dapat mendeteksi gas amonia,

bensol, alkohol, serta gas berbahaya lainnya. Modul ini cocok digunakan pada proses penentuan kualitas udara (*air quality control*).

Gambar 2.10 MQ-135 *Air Quality*

Spesifikasi dari sensor TGS2602 *Air Quality* dapat dilihat pada Tabel 2.4 berdasarkan (Digiware, 2015)

Tabel 2.4 Spesifikasi TGS2602 *Air Quality*

Tegangan kerja	5 VDC
Target gas	amonia (NH3), nitrogen oksida (NOx), alkohol, bensol, asap , dll.
Range deteksi	10 ppm - 300 ppm amonia, 10 ppm - 1000 ppm bensol, 10 ppm - 300 ppm alkohol.
Antarmuka	UART TTL. I2C
Lainnya	- Terdapat 1 buah variable resistor untuk pengaturan nilai threshold secara manual. - Disediakan beberapa jumper untuk konfigurasi pull-up I2C, resistor beban, serta variable resistor threshold. - Memiliki fitur kendali on/off dengan 2 mode kerja pilihan yaitu hysterisis dan window.

2.2.7 Gas / Zat berbahaya di udara

MQ-135 mendeteksi gas berbahaya di udara, sedangkan untuk MG-811 mendeteksi CO_2, kenapa gas ini dikatakan berbahaya, penjelasan dapat dilihat pada Tabel 2.5 berdasarkan (Iskandar & M.Si, 2011).

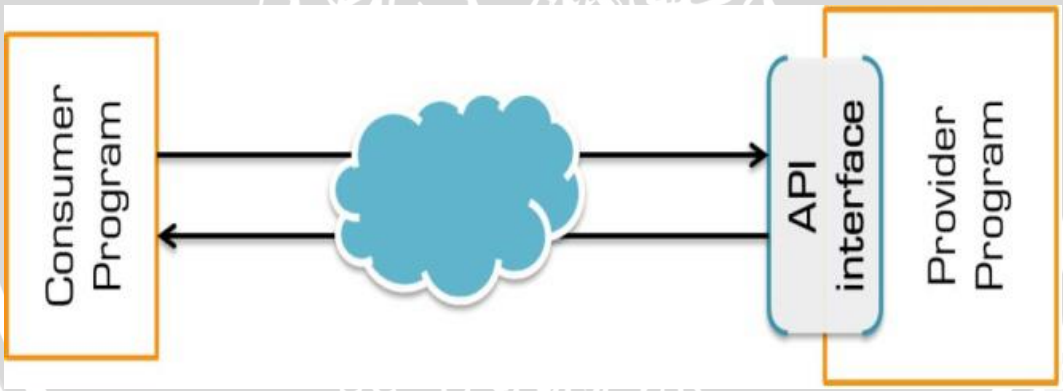
Tabel 2.5 Zat di Udara dan Bahayanya

No	Zat Di udara	Efek yang terjadi akibat Zat berbahaya yang berlebihan
1	Karbondioksida (CO_2)	• Melubangi lapisan Ozon • Efek rumah kaca, cahaya & panas matahari yang masuk kebumi tidak dapat di lepas ke luar angkasa secara kosmis. • Meningkatkan suhu bumi secara global beberapa derajat • Mencairkan es kutub sehingga meningkatkan permukaan air laut
2	Amonia (NH_3)	• Kontak dengan gas amonia berkonsentrasi tinggi dapat menyebabkan kerusakan paru-paru dan bahkan kematian.
3	Nitrogen oksida (NO_x)	• menyebabkan timbulnya Peroxy Acetil Nitrates (PAN). PAN ini menyebabkan iritasi pada mata yang menyebabkan mata terasa pedih dan berair. • Paru-paru yang terkontaminasi oleh gas NO_2 akan membengkak sehingga penderita sulit bernafas yang dapat mengakibatkan kematian.
4	Methanol (Metil Alkohol) CH_3OH	• Pada “keadaan atmosfer” ia berbentuk cairan yang ringan, mudah menguap, tidak berwarna, mudah terbakar, dan beracun dengan bau yang khas (berbau lebih ringan daripada etanol). Ia digunakan sebagai bahan pendingin anti beku, pelarut, bahan bakar dan sebagai bahan additif bagi etanol industri.

No	Zat Di udara	Efek yang terjadi akibat Zat berbahaya yang berlebihan
5	Benzena (Bensol) C_6H_6	• Menyebabkan sakit kepala berkelanjutan
6	asap	• penyebab utama penyakit kardiovaskuler (penyakit jantung) • mengandung Carbon Monoksida arsenik, benzena, benzo[a]pirena, logam berat (timbel, kadmium), hidrogen sianida, dan nitrosamina khusus tembakau

2.2.8 Application Programming Interfaces

Sebuah Application Programming Interface (API) adalah satu set tertentu dari aturan dan spesifikasi yang program perangkat lunak dapat mengikuti untuk mengakses dan memanfaatkan layanan dan sumber daya yang disediakan oleh program lain perangkat lunak tertentu yang mengimplementasikan API. Ini berfungsi sebagai interface antara program perangkat lunak yang berbeda dan memfasilitasi interaksi mereka, mirip dengan cara user interface memfasilitasi interaksi antara manusia dan komputer.



Gambar 2.11 API Inteface

Gambar 2.11 API Antarmuka API dapat diklasifikasikan dalam beberapa kategori tergantung apa abstraksi yang sedang dijelaskan. Deskripsi ini mungkin tampak sangat berbeda, tetapi mereka umumnya ikuti panduan dari definisi. Tabel 2.6 menunjukkan kategori API khas, dan bersama-sama dengan contoh masing-masing dari API tersebut.

Tabel 2.6 Kategori API Beserta Contohnya

API Category	Example	Timeline
Operating System	API for MS Windows API for Apple Mac OS X (Cocoa)	1985- 2001-
Programming Languages	Java API	1995-
Application Services	API for SAP (BAPI)	1990s-
Infrastructure Services	Amazon Web Services API	2002-
Web Services	Twitter API	2006-

2.2.8.1 *Thingspeak API*

ThingSpeak merupakan aplikasi "*Internet of Things*" *open source* dan API untuk menyimpan dan mengambil data dari hal-hal yang menggunakan HTTP melalui Internet atau melalui Local Area Network. Pada tahun 2010, ioBridge meluncurkan ThingSpeak.com, sebuah platform data yang terbuka untuk *Internet of Things*. Dengan ThingSpeak, Anda dapat membuat aplikasi sensor logging, aplikasi pelacakan lokasi, dan jaringan sosial dengan update status. Dari awal, ThingSpeak dirancang untuk menjadi hardware agnostik dan ThingSpeak.com sekarang digunakan oleh ribuan orang untuk agregat data dari sensor dan untuk mengontrol berbagai macam barang lainnya (ThingSpeak, 2016).

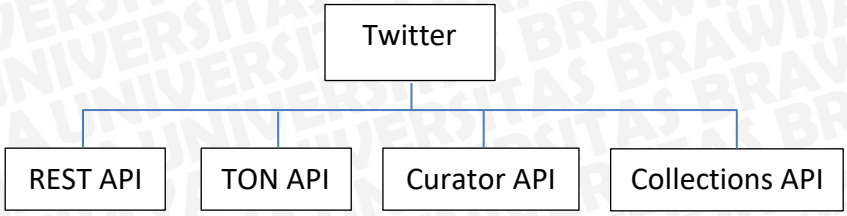
Thingspeak Juga mendukung beberapa fitur, yaitu:

1. API terbuka
2. Pengumpulan data real-time
3. Data geolokasi
4. Pengolahan data
5. Visualisasi data
6. Pesan status perangkat
7. plugin

ThingSpeak menyediakan platform di mana data dikirim dengan menggunakan permintaan HTTP sederhana (melalui HTTP normal atau transportasi HTTPS aman melalui Internet atau LAN). Daftar resmi didukung perangkat termasuk misalnya Arduino, Raspberry oi, ioBridge / RealTime.io dan Listrik Imp.

2.2.8.2 *Twitter API*

Twitter memiliki 4 Jenis Api, yaitu REST API / Public API, TON API, Curator API, dan Collections API seperti pada Gambar 2.12 berdasar pada website official twitter pada menu developer (Twitter, 2015)



Gambar 2.12 Kategori Twitter API

1. REST API menyediakan akses program untuk membaca dan menulis data Twitter. Penulis Tweet baru, membaca profil penulis dan data follower, dan banyak lagi. REST API mengidentifikasi aplikasi Twitter dan pengguna menggunakan OAuth; tanggapan yang tersedia di JSON. OAuth adalah Protokol terbuka untuk memungkinkan otorisasi aman dalam metode yang sederhana dan standar dari aplikasi web, mobile dan desktop.
2. TON (*Twitter Obyek Nest*) API memungkinkan pelaksana untuk meng-upload media dan berbagai aset ke Twitter.
3. Kurator Penyiar API adalah API swasta yang menyediakan penyiar aliran Kurator-dibuat mereka untuk on-air sistem grafis (atau display digital lainnya). Kurator Penyiar API memerlukan izin khusus untuk mengakses dan hanya tersedia untuk digunakan siaran TV kasus.
4. Koleksi adalah kelompok dapat diedit dari Tweet tangan-dipilih oleh pengguna Twitter atau pemrograman dikelola melalui koleksi API. Setiap koleksi publik dan memiliki halaman sendiri di twitter.com, sehingga mudah untuk berbagi dan menanamkan dalam situs web Anda dan aplikasi.

BAB 3 METODOLOGI

Pengembangan sistem pada penelitian ini diawali dengan studi literatur yang terkait dengan tinjauan pustaka dan dasar teori mengenai riset-riset sebelumnya yang berhubungan dengan teknologi *Smart City* dan Sosial Media. Berdasarkan arsitektur *Smart City* pada Gambar 2.1, pada penelitian ini akan dibuat dengan rancangan arsitektur seperti pada Gambar 3.1

Gambar 3.1 Rancangan Arsitektur *Smart City*

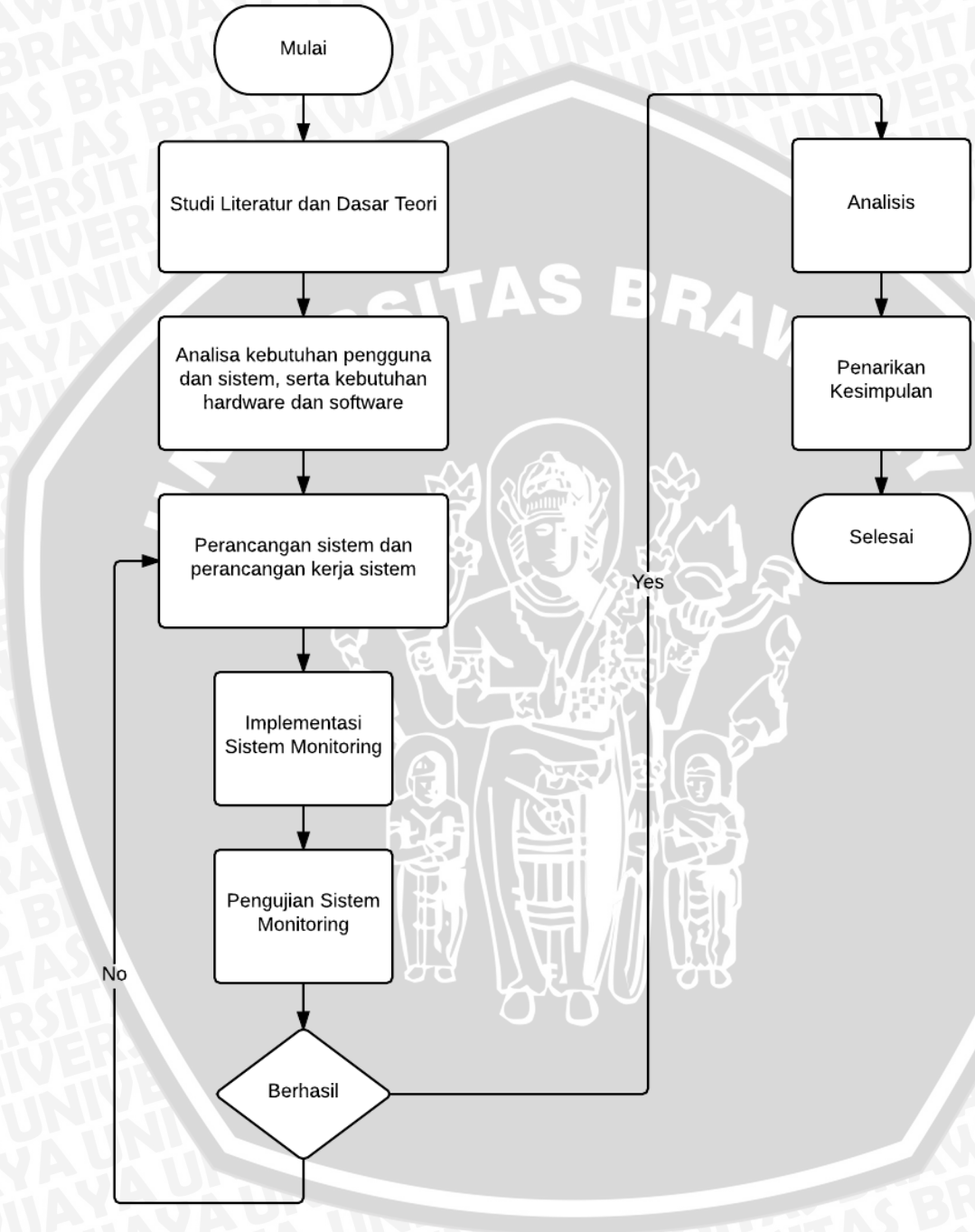
Data Collection: adalah proses pengambilan data oleh sensor untuk mengetahui nilai yang terbaca dalam ruang lingkup sensor.

Data Processing: adalah pengolahan data yang terbaca pada proses sebelumnya, data dibentuk dan diolah agar dapat disimpan dan atau dikirimkan.

Data Collection: adalah pengumpulan dan publikasi data yang sudah diproses atau data matang yang berasal dari proses sebelumnya dan merupakan proses terakhir dari sistem.

Seperti yang dapat dilihat pada Gambar 3.1, konteksnya tidak sebanyak dan seluas yang terdapat pada Gambar 2.1. Dalam penelitian ini bagian device control and alert tidak diikutsertakan karena nantinya penelitian ini akan menjadi terlalu luas dan tidak dapat terfokus ke satu topik.

Berikut merupakan tahapan-tahapan metodologi penelitian untuk pembuatan sistem monitoring kualitas udara yang digambarkan dengan diagram alir pada Gambar 3.2



Gambar 3.2 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Pada bagian ini dibahas mengenai dasar teori yang mendukung penelitian Pengembangan sistem monitoring kualitas udara terintegrasi dengan Sosial Media. Berikut merupkan dasar teori yang digunakan sebagai bahan studi :

1. Penerapan teknologi *Smart City*

   Mempelajari lebih lanjut bagaimana *Smart City* Beroperasi, media perantara apa saja yang dapat digunakan serta komponen pendukung lainnya yang mendorong *Smart City* dapat berjalan.

2. *Wireless Sensor Network* (WSN)

   Penjelasan dasar tentang WSN, apa saja komponen penyusun pembentuk WSN serta bagaimana sebuah sistem dengan metode WSN dapat beroperasi.

3. Sensor, Mikrokontroler dan modul

   Mempelajari pengkabelan Sensor dengan Mikrokontroler dan *module* serta penyusunan kode program agar sensor dapat memberikan hasil yang sesuai dengan fungsinya.

   a. Arduino Uno
      Mempelajari berbagai fungsi serta pengaplikasian sebagai mikrokontroler atau otak utama dari sistem yang digunakan.

   b. TGS2602 *Air Quality*
      Penjelasan tentang susunan pin, fungsi dari sensor, serta mengetahui input apa yang diperlukan sehingga didapat sensor yang dapat menghasilkan output berupa kualitas udara yaitu kadar H2S dan NH3 di udara.

   c. MG-811 CO_2 *Gas Sensor*
      Penjelasan tentang susunan pin, fungsi dari sensor, serta mengetahui input apa yang diperlukan sehingga didapat sensor yang dapat menghasilkan output berupa kualitas udara yaitu kadar H2S dan NH3 di udara.

   d. ESP 8266
      Mempelajari cara kerja serta konfigurasi dari ESP 8266 sebagai module internet agar mikrokontroler dapat terhubung kedalam jaringan WiFi.

4. Sosial media API

   Penjelasan tentang apa itu API, apa saja yang dibutuhkan untuk menggunakan API serta bagaimana API dari sebuah Sosial Media bekerja dan bagaimana pengaturannya.

a. Facebook API

Penjelasan tentang bagaimana cara mengatur dan membuat agar sebuah sistem dapat melakukan update status dengan akun facebook melalui facebook API.

b. Twitter API

Penjelasan tentang bagaimana cara mengatur dan membuat agar sebuah sistem dapat membagikan tweet dengan akun twitter melalui twitter API.

5. Analisa QoS

Bagaimana cara mengetahui *Quality of Service (QOS)* mulai dari akuisisi data hingga data bisa sampai di Sosial Media.

3.2 Analisis Kebutuhan Sistem

Analisa kebutuhan diperlukan untuk menganalisa apa saja yang dibutuhkan oleh sistem pada penelitian yang dilakukan, sehingga hasil penelitian dapat dicapai dan sesuai dengan yang diharapkan. Berikut beberapa kebutuhan yang akan digunakan dan dibutuhkan dalam penelitian ini:

1. Sistem dapat membaca kualitas udara dan CO_2 yang berada disekitarnya.
2. Sistem dapat terhubung ke jaringan Internet menggunakan Wi-Fi.
3. Nilai sensor diolah dalam mikrokontroler dan dikirimkan ke Cloid IOT setiap satuan waktu tertentu serta update status twitter apabila data telah terkirim ke IoT *Cloud*.

3.2.2 Kebutuhan Hardware

Berikut beberapa kebutuhan Hardware yang digunakan dalam membuat sistem dalam penelitian ini sebagai berikut.

1. Mikrokontroler

   Mikrokontroler merupakan bagian utama dari penelitian sistem ini, mikrokontroler ini digunakan sebagai pengolah, pengirim dan penerima data. Mikrokontroler yang digunakan dalam penelitian ini adalah Arduino Uno.

2. Sensor MQ-135

   Sensor MQ-135 digunakan sebagai sensor yang berfungsi membaca Kualitas Udara yang tercover oleh sensor.

3. Sensor MG-811

   Sensor MG-811 digunakan sebagai sensor ynag berfungsi membaca kadar CO_2 di udara pada area yang tercover oleh sensor.

4. ESP 8266

Modul ESP 8266 digunakan sebagai modul penghubung antara mikrokontroler dengan jaringan WiFi dan juga sebagai pengirim data ke media sosial.

5. AC Adapter/Power Supply

   AC Adapter/Power Supply sebagai catu daya untuk menghidupkan mikrokontroler yang menghidupkan sensor serta module yang terhubung dengan mikrokontroler.

3.2.3 Kebutuhan Software

1. IoT *Cloud*

   IoT *Cloud* digunakan sebagai wadah penampung data, dimana data disimpan dan dapat dilihat secara grafis.

2. Akun Sosial Media Dan Twitter

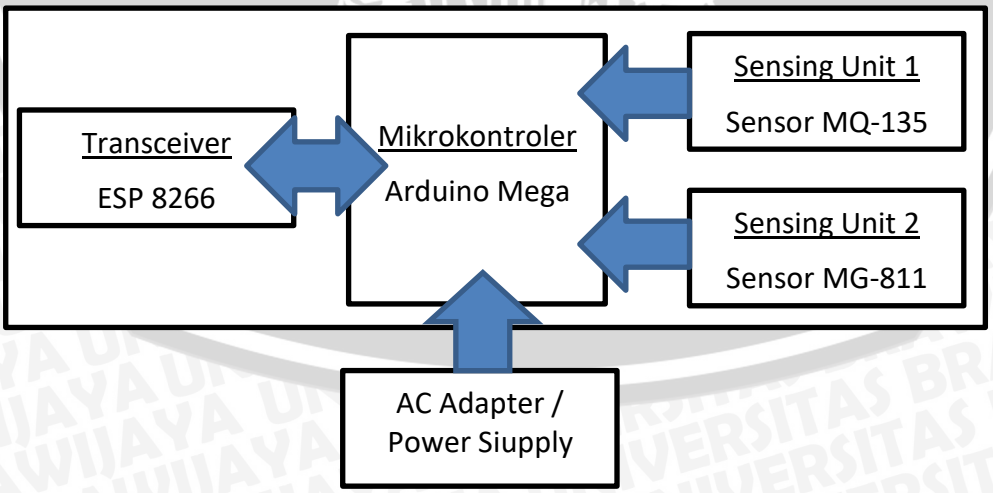
   Akun Sosial Media sebagai antarmuka atau perwujudan hasil pembacaan data yang dilakukan oleh sensor, sehingga seluruh masyarakat dapat melihat hasilnya melalui Sosial Media.

3. Jaringan WiFi

   Jaringan WiFi digunakan sebaga Media perantara antara mikrokontroler dengan jaringan internet.

3.3 Perancangan Sistem

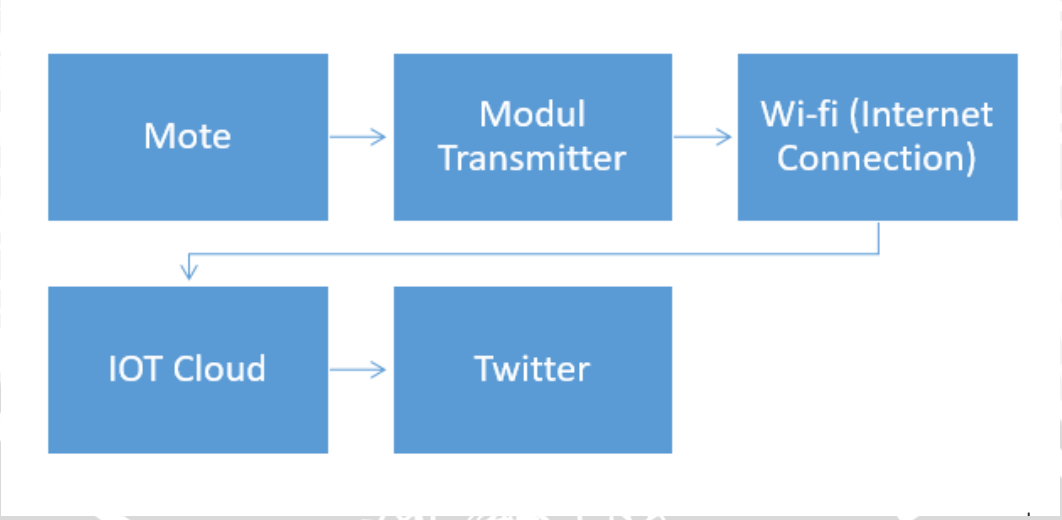
Tahap ini adalah tahap setelah melakukan studi literatur dan analisa kebutuhan. Perancangan sistem perangkat keras pada node dapat dilihat pada Gambar 3.2



Gambar 3.3 Diagram Blok Sistem Node

Gambar 3.2 merupakan diagram blok dari perancangan sistem node pada percobaan ini, dimana ESP 8266 sebagai *transceiver*, mikrokontroler sebaga

processing unit, Sensor MQ-135 sebaga Sensing Unit 1 dan Sensor MG-811 sebagai Sensing Unit 2. Untuk penelitian ini akan lebih difokuskan untuk pengiriman data dari node hingga dapat ditampilkan di IoT *Cloud* dan sosial media serta analisa lama waktu akuisisi data hingga dapat ditampilkan di Sosial Media.



Gambar 3.4 Diagram Blok Kerja Sistem

Gambar 3.3 merupakan diagram blok kerja sistem yang mana nanti setelah sensor berhasil membaca Kualitas Udara dan CO_2 pada udara disekitarnya, informasi akan diolah oleh mikrokontroler dan dikirim oleh Modul ESP8266 melalui jaringan WiFi disekitarnya dengan tujuan IoT *Cloud* sehingga data yang terkirim akan muncul pada IoT *Cloud* yang telah didaftarkan dan disetting agar dapat membuat postingan pada sosial media setiap kali mote mengirimkan data.

3.4 Implementasi

Pada tahap ini akan dilakukan impelentasi dari Sistem Monitoring Kualitas Udara sesuai dengan yang sudah dijelaskan pada tahap perancangan sistem sehingga dapat dilakukan tahap pengujian untuk mengetahui apakah sistem sesuai dengan yang direncanakan dari penelitian ini. Implementasi penelitian ini meliputi:

1. Pemasangan mote pada sebuah ruangan berukuran 4x4 meter.
2. Impelemtasi pengiriman data dari tiap mote ke IoT *Cloud*.

3.5 Pengujian dan analisis

Pada tahap ini terbagi menjadi dua bagian yaitu tahap pengujian dan analisis kesalahan, serta analisis kualitas layanan dimana analisis kualitas layaan akan dilakukan apabila pengujian telah berhasil dilakukan. Tahap pengujian dilakukan untuk mengetahui apakah sistem monitoring kualitas udara yang sudah dibuat berjalan sesuai dengan yang diinginkan. Apabila belum maka analisa kesalahan akan dilakukan, baik pada sisi komponen penyusun ataupun kode program. Terdapat beberapa skenario pengujian yang akan dilakukan terhadap sistem yang

akan dirancang dan disesuaikan dengan kebutuhan fungsional dan non fungsional dari sistem ini adalah sebagai berikut

1. Konektivitas dari ESP8266, baik dari ESP8266 ke *Access Point*, ataupun dari ESP8266 ke thingspeak dan thingtweet.
2. Fungsionalitas dari sensor dan pengolahan data yang dilakukan oleh sensor dan Arduino.
3. *Delay* yang dihasilkan atau waktu yang dibutuhkan oleh sensor untuk mengakuisisi data hingga terkirim ke thingspeak dalam waktu 1 jam.

Analisis kesalahan berdasarkan konektivitas adalah dikarenakan banyaknya traffic data pada *Access Point*, tidak kebagian IP *Address* dan terdapat Halaman Login sebelum mendapatkan Access. Fungsionalitas dari sensor adalah apakah sensor tersebut bekerja dan memberikan data.

Sedangkan analisis *delay* adalah berapa banyak data yang dapat dihasilkan yang dibandingkan dengan perhitungan *delay* yang sudahditetapkan dalam program.

3.6 Penarikan Kesimpulan

Tahap ini merupakan tahap yang dilakukan setelah tahap-tahap sebelumnya terpenuhi yaitu studi literatur, analisa kebutuhan sistem, perancangan, impelentasi serta tahap pengujian dan analisis. Penarikan kesimpulan diambil dari hasil tahap pengujian dan analisis sistem, selain itu pada tahap ini ditambahkan saran sebagai hasil akhir yang diharapkan dapat dipergunakan sebagai batu loncatan untuk penelitian selanjutnya.

BAB 4 REKAYASA PERSYARATAN

Pada bab ini akan menjelaskan tentang deskripsi umum dan kebutuhan fungsional non fungsional dari sistem yang akan dibuat

4.1 Deskripsi Umum

Pada sub bab ini akan dijelaskan tentang prespektif sistem, kegunaan, karakteristik pengguna, batas perancangan dan implementasi serta asumsi dan ketergantungan dari sistem yang akan dibuat

4.1.1 Prespektif Sistem

Sistem dalam penelitian ini dapat bekerja normal apabila setiap mote terdapat jaringan wifi yang SSID nya sudah diketahui dan dituliskan kedalam program, apabila jaringan mote terhubung ke jaringan internet, mote akan membaca kualitas udara dan CO_2 disekitarnya. Hasil baca data tersebut diolah dari bentuk ADC ke dalam bentuk variable dengan besaran PPM yang kemudian data ini dikirim ke IoT *Cloud* untuk disimpan dan dipublikasi ke twitter sesuai dengan tujuan penelitian.

4.1.2 Kegunaan

Kegunaan dari penelitian ini adalah agar dapat menghasilkan sistem monitoring kualitas udara yang menggunakan jaringan wifi agar dapat digunakan di daerah perkotaan, data yang dikirimkan berupa hasil baca sensor yang telah diolah dalam mikrokontroler yang disimpan ke dalam IoT *Cloud* melalui jaringan wifi, sistem ini bertujuan untuk memberikan informasi kualitas udara disekitar sistem melalui IoT *Cloud* dan sosial media, sistem ini dibangun dengan tujuan membangun salah satu dari sekian banyak kemungkinan untuk merealisasikan *Smart City*, dimana hasil penelitian ini dapat digunakan untuk penelitian selanjutnya dengan melakukan penambahan fitur ataupun melakukan penyempurnaan sistem dari kekurangan-kekurangan yang ada sesuai kebutuhan.

4.1.3 Karakteristik Pengguna

Pengguna dalam penelitian ini hanya berperan dalam pengamatan data yang terakuisisi pada IoT *Cloud* dan sosial media, karena penelitian ini berfokus pada smart city dimana pengguna selalu dapat melihat informasi dari sistem melalui IoT *Cloud* dan sosial media.

4.1.4 Batas Perancangan dan Implementasi

Ada beberapa batasan masalah yang terdapat dalam perancangan implementasi dari sistem yang dibuat, yaitu

1. Konfigurasi alat harus dilakukan terlebih dahulu, yaitu memasukkan SSID dan password agar dapat terhubung dengan wifi.

2. Sistem yang dibuat hanya dapat melakukan pembacaan data sensor dan pengiriman data sesuai waktu yang ditentukan, sistem ini tidak dapat mengirimkan data melalui permintaan ataupun *interrupt* lain.
3. Dikarenakan sensor selalu mengalami *overheat* dan pembacaan data menjadi tidak berfungsi semestinya, alat ini hanya dianjurkan untuk menyala selama 1 jam dan dimatikan, kurang lebih selama 1 jam untuk *cooldown*.
4. Apabila jaringan wifi terputus dikarenakan *out of range* atau tidak kebagian *IP Address*, alat harus direset secara manual, dengan cara mencabut adapter dan memasangnya kembali.
5. Apabila password pada wifi diganti atau SSID diganti/dihapus dari router, maka konfigurasi ulang sistem harus dilakukan.

4.1.5 Asumsi dan ketergantungan

Ada beberapa asumsi dan ketergantungan dari sistem yang dibuat untuk penelitian ini, yaitu

1. Alat harus disetting terlebih dahulu apabila pindah wifi/SSID.
2. Alat tidak dapat bekerja apabila tidak terhubung dengan wifi.
3. Alat ini hanya bekerja sesuai dengan program yang tertulis, tidak dapat diberikan perintah atau instruksi dari luar.

4.2 Kebutuhan fungsional

Kebutuhan fungsional merupakan kebutuhan yang digunakan agar sistem yang dibuat untuk penelitian ini dapat berjalan dan memberikan keluaran seperti yang diharapkan, adapun kebutuhan fungsional yang harus dipenuhi akan dijelaskan pada sub bab berikut.

4.2.1 Fungsi Transceiver terhubung dengan internet secara nirkabel

1. Penjelasan dan prioritas

   Fungsi ini sama pentingnya dengan fungsi sebelumnya dengan prioritas yang tinggi, fungsi ini berperan untuk melakukan komunikasi pengiriman dari mote ke jaringan internet sebelum melakukan pengiriman.

2. Stimulus/respon sistem

   Mote akan terlebih dahulu melakukan konfigurasi konektivitas ke jaringan wifi dengan meminta IP *Address* dan akses HTTP, ketika IP address dan akses telah didapatkan maka dapat dilanjutkan ketahap pembacaan data sensor dan pengiriman.

3. Kebutuhan fungsional

   Fungsi ini haruslah terpenuhi, jika tidak maka mote tidak akan bisa mengirimkan paket dan menjadikan sistem tidak berjalan dengan semestinya.

4.2.2 Fungsi Pembacaan data sensor dan pengolahan data

1. Penjelasan dan Prioritas

   Fungsi ini adalah fungsi lain yang harus dipenuhi dan memiliki prioritas yang tinggi dalam penelitian ini, fungsi ini melakukan tahap selanjutnya dari fungsi sebelumnya yang membuat sistem bisa membaca Kadar CO_2 dan zat berbahaya di sekitar sistem setelah *transceiver* terhubung.

2. Stimulus/respon Sistem

   Data yang telah diambil diolah dengan menggunakan ADC dan kemudian dikirimkan dengan *transceiver* menuju IoT *Cloud*.

3. Kebutuhan fungsional

   Fungsi ini dibutuhkan agar informasi yang di olah dapat dikirim dari mote dan diterima oleh IoT *Cloud*, bila salah satu tidak memenuhi baik itu fungsi pembacaan sensor ataupun pengiriman data akan membuat sistem tidak berjalan dengan yang diharapkan

4.3 Kebutuhan Non Fungsional

Adapun keutuhan non fungsional dari sistem yang dibuat, kebutuhan ini adalah kebutuhan tidak tertulis yang harus dipenuhi agar sistem dapat berjalan sesuai dengan yang diharapkan.

4.3.1 Kebutuhan Performa

Sistem yang dibuat untuk penelitian ini akan bekerja dengan baik apabila mote mendapatkan IP *Address* dan dalam cakupan area WiFi dan suhu udara yang tidak terlalu ekstrim, estimasi dari area cakupan sensor kurang lebih 1 Meter, dan juga dengan *traffic bandwidth* yang tidak terlalu padat agar dapat mengirim data tanpa terjadi *packet loss*.

4.3.2 *Delay*

Sistem ini hanya memiliki satu jenis Mote, pada Mote yang menyala selama 1 jam, dimana dalam waktu tersebut digunakan untuk *booting, connecting, dan scanning*, dan 40 detik digunakan untuk akuisisi pengiriman data ditambah dengan stand by mode selama 20 detik dan kemudian restart.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

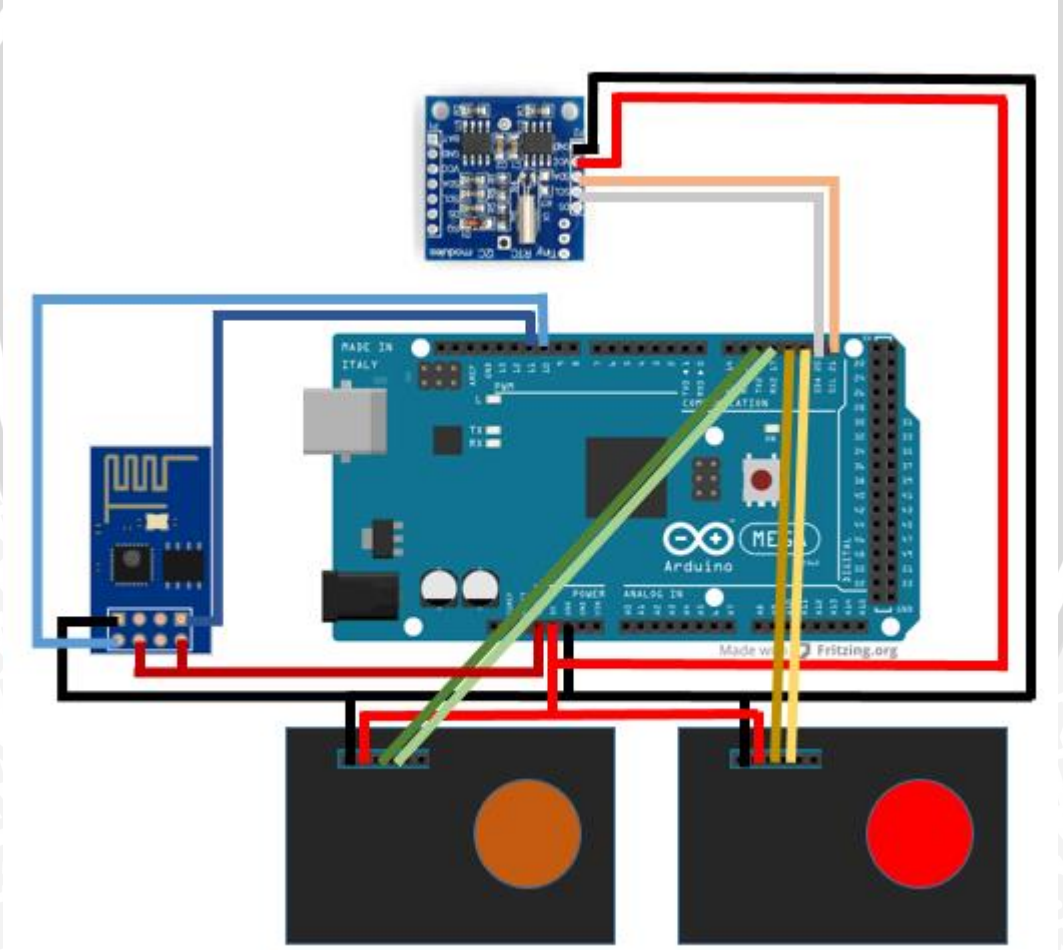
Pada bab ini akan dijelaskan tentang perancangan, cara kerja dan implementasi sistem dari pengembangan sistem monitoring kualitas udara terintegrasi IoT *Cloud* untuk diterapkan pada *smart city* secara lebih mendetail lagi.

5.1 Perancangan Sistem

Pada bagian perancangan sistem ini terdapat penjelasan mengenai perancangan perangkat keras dan perangkat lunak.

5.1.1 Perancangan Perangkat Keras

Pada penelitian ini dilakukan perancangan perangkat keras yaitu perancangan mote / node sensor untuk sistem monitoring kualitas udara. Adapun skematik dari perancangan rangkaian sensor dapat dilihat pada Gambar 5.1.



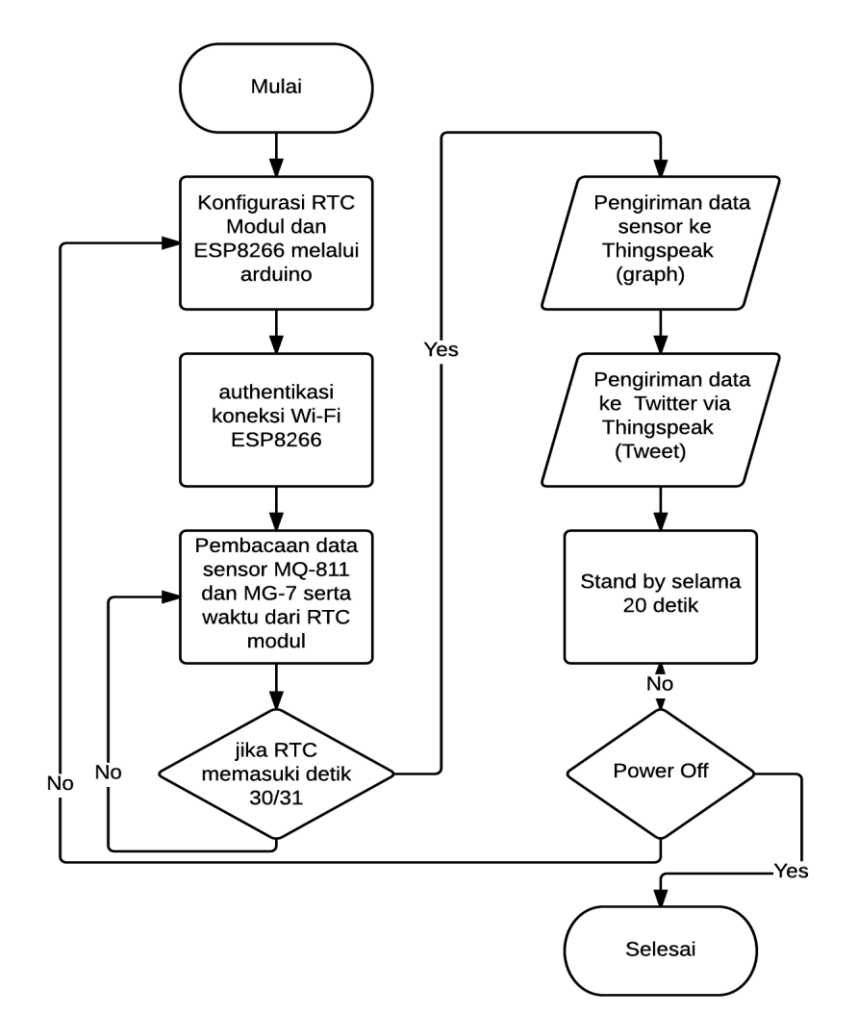
Gambar 5.1 Skematik perancangan rangkaian sensor

Bisa dilihat pada Gambar 5.1 yang digunakan sebagai mikrokontroler adalah Arduino Mega 2560 dan sensor pendeteksi adalah menggunakan sensor MQ-

135(Kadar zat berbahaya) dan sensor MG-811(Karbon Dioksida) yang setiap pin VCC sensor dihubungkan ke pin 5V Arduino karena daya yang dibutuhkan untuk beroperasi adalah 5V, untuk MQ-135 pin Tx Rx dihubungkan ke pin Rx1 Tx1 Arduino dan untuk MG-811 pin Tx Rx dihubungkan ke pin Rx2 Tx2 Arduino. kemudian juga terdapat ESP8266 sebagai *Transceiver*, ESP8266 menggunakan VCC 3.3V, dan Tx Rx dari ESP8266 dihubungkan dengan pin 10 dan 11 Arduino. Selain itu juga ditambahkan DS1307RTC sebagai *Real Time Clock*, yang menggunakan VCC 5V, dimana pin SDA SCL dihubungkan dengan SDA SCL Arduino.

5.1.2 Perancangan Perangkat Lunak

Pada bagian perancangan perangkat lunak untuk sistem rangkaian sensor menggunakan Arduino IDE. Adapun penjelasan algoritma yang digunakan untuk monitoring kualitas udara pada sistem rangkaian sensor dapat dilihat pada Gambar 5.2.



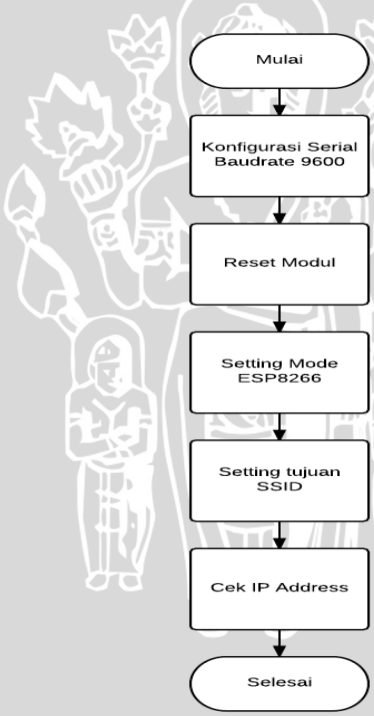
Gambar 5.2 Diagram Alir Algoritma Monitoring Udara

Gambar 5.2 merupakan diagram alir dari sistem monitoring udara, sebelum sistem bekerja, ESP8266 dan RTC Modul harus dikonfigurasi terlebih dahulu dan

mencoba untuk terhubung dengan jaringan wifi yang ditentukan, setelah terhubung ESP8266 mengecek autentikasi dengan cara melihat *IP Address* yang dimillikinya. Setelah IP *Address* didapatkan dan terhubung ke internet, pembacaan data mulai dilakukan, setelah data dibaca dan waktu memasuki detik 30 atau 31 dalam jam yang digunakan oleh RTC maka data akan dikirim ke thingspeak, thingspeak digunakan sebagai IoT *Cloud* dan menyimpan data sensor dalam bentuk grafik. Tidak hanya itu, sistem juga mengirim sebuah kalimat ke twitter atau dapat disebut *Tweet* setelah mengirimkan data ke thingspeak, kemudian sistem memasuki cooling down selama 20 detik, apabila dalam 20 detik alat tidak dimatikan, maka program akan berjalan kembali dari awal. Untuk memperjelas bagian perangkat lunak ini akan dicerca sebagai berikut

5.1.2.1 Perancangan Perangkat Lunak ESP8266

ESP8266 dapat deprogram dengan 2 cara, yaitu AT *Command* dan LUA, dikarenakan pada penelitian ini menggunakan Arduino maka Bahasa Pemprograman yang digunakan adalah Arduino AT *Command*, algoritma program ESP8266 terdapat pada diagram alir pada Gambar 5.3.



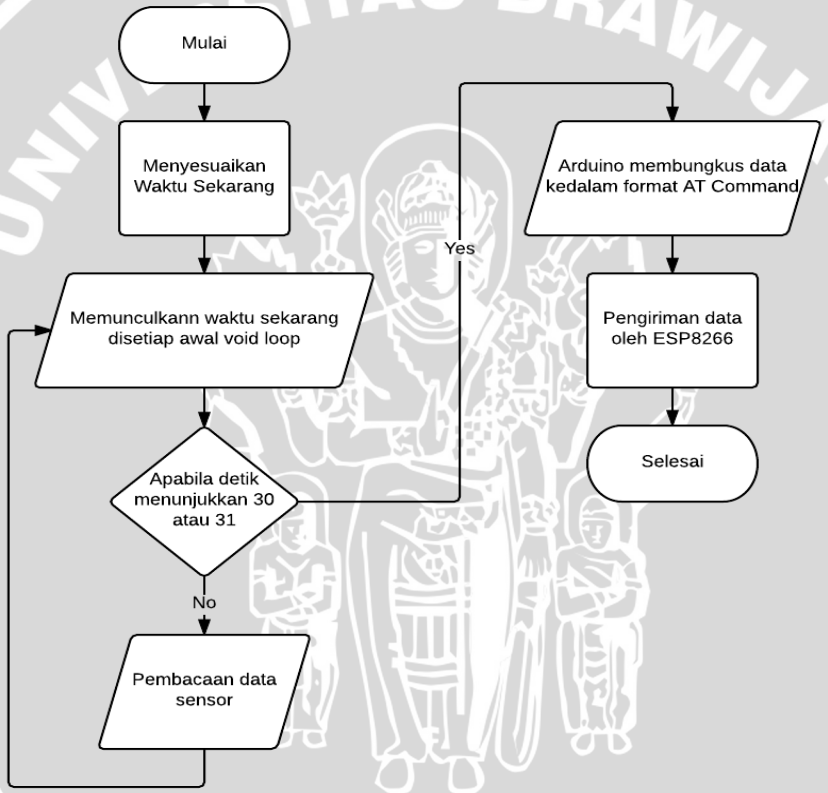
Gambar 5.3 Diagram Alir Perangkat Lunak ESP8266

Seperti yang tergambarkan pada Gambar 5.3 urutan cara kerja ESP8266 yaitu dengan menyamakan baudrate terlebih dahulu yaitu 9600, kemudian memasukkan perintah untuk melakukan soft reset agar konfigurasi sebelumnya tidak tertumpuk dengan konfigurasi yang baru, konfigurasi ESP8266 agar dapat terhubung ke wifi dan dapat melakukan upload data dengan perintah CWMODE=1 yaitu *Station Mode*, kemudian mencari wifi dengan SSID yang sesuai dengan input, dan memasukkan password apabila dibutuhkan.

ESP8266 memerlukan waktu sekitar 40 detik untuk terhubung ke wifi dikarenakan ESP menggunakan TCP sehingga memakan waktu yang cukup lama untuk melakukan *Discover, Over, Request* dan ACK. Setelah 40 detik ESP8266 diperintahkan untuk mengecek IP *Address*, apabila tidak 0.0.0.0 maka ESP8266 sudah terhubung ke jaringan wifi. Kapan dan dalam kondisi apa ESP8266 mengirim data akan dijelaskan pada Gambar 5.4.

5.1.2.2 Perancangan Perangkat Lunak RTC Modul

RTC Modul terhubung ke Arduino melalui Pin SDA dan SCL, dimana pin ini tidak memerlukan baudrate, sehingga data yang ada pada modul dapat langsung dibaca apabila diminta untuk diproses oleh Arduino, Algoritma mendetail mengenai RTC *Module* dapat dilihat pada Gambar 5.4.

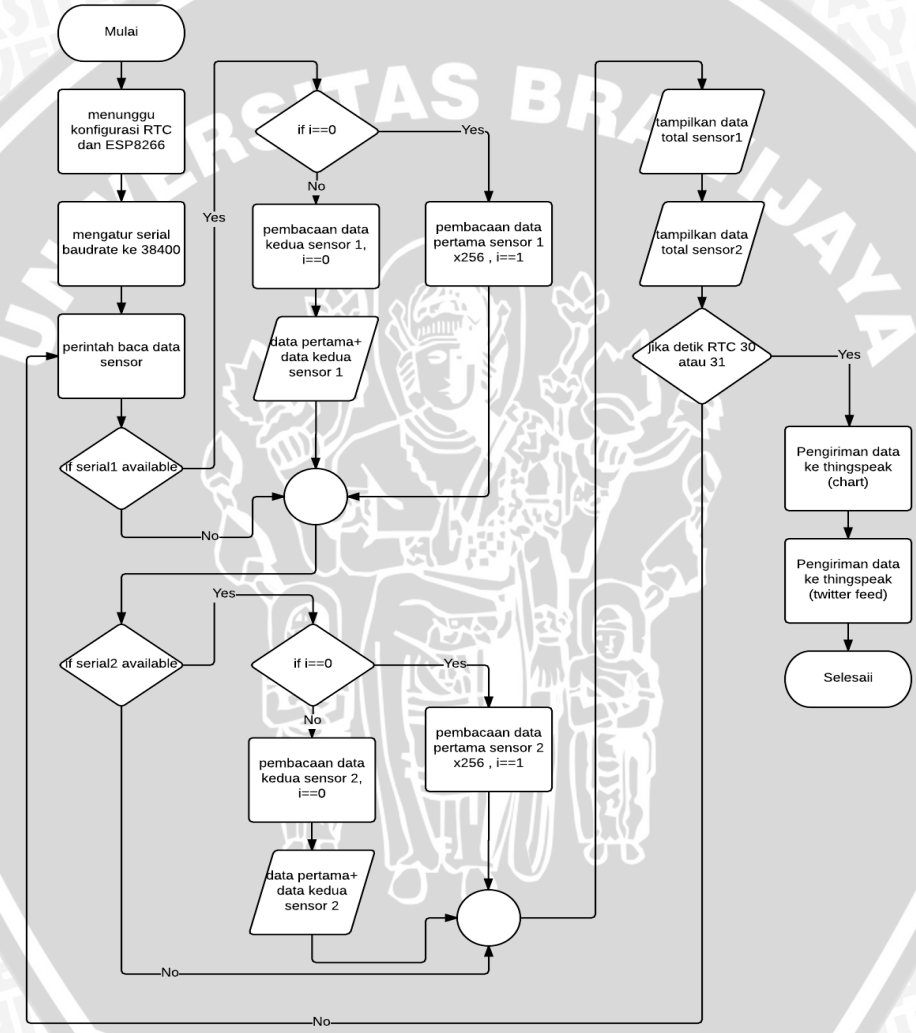


Gambar 5.4 Diagram Alir Perangkat Lunak RTC Modul

Seperti yang tergambarkan pada Gambar 5.4 RTC perlu dikonfigurasi terlebih dahulu agar waktu yang ditunjukkan oleh RTC sesuai dengan waktu sekarang, apabila sudah sesuai maka waktu akan dipanggil dan ditampilkan melalui serial monitor, apabila waktu yang ditunjukkan belum memasuki detik ke 30 atau 31 maka baca nilai sensor, apabila memasuki detik 30 atau 31 maka data yang tadi telah dibaca dan disimpan kedalam variabel akan dibentuk kedalam format pengiriman menggunakan AT *Command*, dan dikirimkan dengan ESP8266 melalui port 80. Untuk bagaimana pengambilan nilai sensor dan pembentukan data kedalam format AT *Command* akan dijelaskan pada Gambar 5.5.

5.1.2.3 Perancangan Perangkat Lunak Sensor

Sensor yang digunakan pada penelitian ini ada 2, yaitu MQ-135 sensor *Air Quality* dan MG-811 sensor Karbon Dioksida, MQ-135 menggunakan serial1 sedangkan MG-811 menggunakan serial2, pembacaan data untuk tiap sensor ini sama, yaitu data terbagi menjadi 2 bagian, bagian pertama adalah datanya bernilai n*256, dan data kedua adalah nilai yang tidak memenuhi hingga 256, atau 0-255, agar data tersebut dapat dibaca, data sensor yang terpisah tersebut harus digabungkan dengan perhitungan aritmatika, untuk lebih jelasnya akan dijelaskan melalui Gambar 5.5.



Gambar 5.5 Diagram Alir Perangkat Lunak Sensor

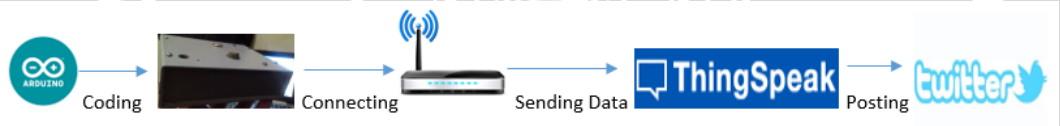
Setelah selesai mengkonfigurasi RTC dan ESP8266, *baudrate* serial1 dan serial2 diset menjadi 38400 untuk dapat berkomunikasi dengan sensor, kemudian dengan memasukkan perintah serial1.read(0x41), serial1 meminta data yang terbaca oleh sensor MQ-135, data pertama dikalikan dengan 256 dan disimpan kedalam variabel, kemudian data kedua langsung disimpan kedalam variabel pula. Variabel baru lain akan menampung hasil jumlah kedua variabel yang menampung data

sensor1, dimana data ini nantinya yang akan ditampilkan dan digunakan untuk dikirimkan ke IoT *Cloud*.

Sensor2 tidak jauh berbeda dengan pembacaan sensor1, juga menggunakan serial2.read(0x41) hanya saja hasil baca sensornya disimpan kedalam variabel yang berbeda dari variabel untuk sensor1, agar datanya tetap valid. Setelah kedua data terpenuhi, data ditampilkan ke serial monitor Arduino. Apabila waktu yang ditunjukkan oleh RTC memasuki detik ke 30 atau 31 maka pengiriman data yang telah disimpan dalam variabel tersebut diformat kedalam bentuk AT *Command* dengan fungsi HTTP GET agar dapat dikirimkan melalui port 80 menuju IP Public milik Thingspeak. Setiap pengiriman data memerlukan waktu 16 detik, sehingga data tidak bisa dikirim secara kontinyu dalam waktu singkat.

5.2 Gambaran Kerja Sistem

Pada bagian ini menjelaskan bagaimana sistem yang telah dirancang dapat bekerja, adapun gambaran kerja sistem yang dijelaskan adalah bagaimana sistem mengirimkan data yang didapatkan dari sensor sehingga data tersebut dapat *ter-publish* di thingspeak dan twitter, gambaran kerja sistem dapat dilihat pada Gambar 5.3.



Gambar 5.6 Gambaran Kerja Sistem

Agar sistem dapat berkerja pertama-tama lakukan konfigurasi pada pemrograman Arduino yaitu memasukkan SSID dan Password untuk Wi-Fi yang digunakan, hasil konfigurasi adalah IP *Address* yang didapatkan, kemudian pembacaan data sensor dilakukan, dan apabila memasuki detik tertentu, data akan dikirim ke Thingspeak untuk masuk kedalam database dan juga posting ke halaman twitter.

5.3 Implementasi

Implementasi sistem dapat dilakukan jika tahapan perancangan perangkat sudah terpenuhi dimana implementasi sistem ini dilakukan sesuai dengan perancangan, ada beberapa langkah konfigurasi yang harus dilakukan terlebih dahulu yaitu implementasi perangkat keras dimana disesuaikan dengan bab sebelumnya dan implementasi perangkat lunak yang terdiri dari konfigurasi thingspeak, konfigurasi twitter dan konfigurasi konfigurasi sensor.

5.3.1 Implementasi Perangkat Keras

Implementasi perangakat keras disesuaikan dengan bab sebelumnya, yaitu menghubungkan RTC *Module*, ESP 8266, Sensor MQ-135 dan MG-811 dengan Arduino Mega sesuai dengan perancangan sistem.

5.3.2 Implementasi Perangkat Lunak

Untuk implementasi perangkat lunak sistem disini penulis menggunakan Bahasa Pemprograman Arduino, selain dikarenakan penulis menggunakan Arduino Mega 2560 sebagai Mikrokontroler, namun juga karena Arduino mendukung AT *Command* yang digunakan oleh ESP 8266 sebagai Bahasa perintahnya. Adapun kode program yang digunakan akan dijelaskan pada sub bab berikut

5.3.2.1 Konfigurasi ESP 8266

ESP 8266 perlu di konfigurasi agar dapat terhubung ke jejaring internet dengan Arduino, penulisan Bahasa yang digunakan berupa AT *Command*, adapun beberapa perintah AT *Command* yang digunakan dapat dilihat pada Tabel 5.1.

Tabel 5.1 Perintah AT *Command*

Perintah	Fungsi	Set/Execute
AT+RST	restart the module	AT+RST
AT+CWMODE	wifi mode	AT+CWMODE=<mode>
AT+CIFSR	Get IP address	AT+CIFSR
AT+CWJAP	join the AP	AT+ CWJAP =<ssid>,<pwd>
AT+CIPSTART	set up TCP or UDP connection	AT+CIPSTART= <type>,<addr>,<port>
AT+CIPSEND	send data	AT+CIPSEND=<length>
AT+CIPCLOSE	close TCP or UDP connection	AT+CIPCLOSE

Perintah AT *Command* ini digunakan untuk menghubungkan ESP8266 dengan *Access Point*, dan juga menghubungkan ESP8266 ke IP tujuan untuk melakukan komunikasi atau pengiriman data, bentuk pemrograman Arduino dapat dilihat pada Gambar 5.7 sampai Gambar 5.9.

```
1  #include <SoftwareSerial.h>
2  #include <Wire.h>
   #define DEBUG true
3  .....
   SoftwareSerial esp8266(10,11);
4  .....
```

Gambar 5.7 *Script* Arduino untuk ESP8266 Part 1

Fungsi pada Gambar 5.7 untuk pemanggilan *library* dan pendefinisian fungsi yang digunakan dan berkaitan dengan *script* selanjutnya, yang dimana jika tidak diikutsertakan pada program, alat tidak akan berfungsi sebagaimana yang diinginkan.

Pada Gambar 5.7 dapat dilihat bahwa dalam *script* digunakan 2 *library*, yaitu wire.h dan SoftwareSerial.h dimana *library* ini berfungsi untuk memfungsikan pin analog pada Arduino sebagai pin serial untuk pertukaran data antara ESP8266 dan Arduino Mega, pin yang diubah hanya pin 10 dan pin 11, karena ESP8266 hanya menggunakan Tx dan Rx untuk berkomunikasi dengan Arduino.

```
5   ....
6   esp8266.begin(9600);
7   kirimPerintah("AT+RST\r\n", 2000, DEBUG);
8   kirimPerintah("AT+CWMODE=1\r\n", 1000, DEBUG);
9   kirimPerintah("AT+CWJAP=\"cocacola\",\"SadangWoy12312\"\r\n",
10  3000,DEBUG);
    delay(20000);
11  kirimPerintah("AT+CIFSR\r\n", 1000, DEBUG); //Mendapatkan IP adress
12  Serial.println("ESP8266 Ready!");
13  .....
```

Gambar 5.8 *Script* Arduino untuk ESP8266 Part 2

Gambar 5.8 terdapat fungsi untuk melakukan komunikasi dengan ESP8266 yaitu dengan cara menyamakan baudrate ke 9600 dan mengirimkan perintah AT *Command*, setiap perintah atau inputan yang dikirim ke ESP selalu diikuti oleh \r\n dimana fungsinya adalah sebagai *end command* dan *end line*.

Setiap perintah memiliki *delay*, untuk perintah AT+RST atau reset modul diperlukan waktu 2 detik, dan untuk mengubah mode atau perintah AT+CWMODE diperlukan waktu 1 detik, sedangkan untuk dapat terhubung ke *Access Point* dengan perintah AT+CWJAP diperlukan waktu sekurangnya 20 detik, karena ESP 8266 menggunaan mode DHCP dan memerlukan waktu untuk request IP. Setelah 20 detik, perintah untuk melihat IP atau AT+CIFSR dapat dieksekusi dan memunculkan IP address milik ESP8266.

```
14  String kirimPerintah(String perintah, const int timeout, boolean debug)
15  {
16     String response = "";
17     esp8266.print(perintah);
18     long int time = millis();
19     while( (time+timeout) > millis()) {
20     while(esp8266.available()) {
21     char c = esp8266.read();
22     response+=c;}}
23  if(debug) {
24   Serial.print(response)
25  ;}
26  return response;}
```

Gambar 5.9 *Script* Arduino untuk ESP8266 Part 3

Pada gambar 5.9 adalah fungsi untuk mengirimkan AT *Command* ke ESP8266, dimana balasan dari *AT* command disimpan pada variabel response yang selalu dikosongkan setiap kali perintah dikirimkan, perintah dikirimkan dalam bentuk

print, dan hasilnya dibaca dalam bentuk read. Ketika waktu sekarang ditambah dengan timeout atau waktu yang dibutuhkan untuk tiap eksekusi

5.3.2.2 Konfigurasi RTC *Module*

RTC module dihubungkan dengan Arduino Mega sesuai dengan perancangan sistem pada bab sebelumnya, RTC digunakan untuk menunjukkan waktu sekarang dan melakukan pengiriman data pada waktu yang ditentukan, adapun *script* untuk konfigurasi RTC module dapat dilihat pada Gambar 5.10 sampai dengan Gambar

```
1  #include <Wire.h>
2  #include <Time.h>
3  #include <DS1307RTC.h>
4
5  const char *monthName[12] = {
6    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
7    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
8  };
9
10 tmElements_t tm;
```

Gambar 5.10 *Script* Konfigurasi RTC pada Arduino Part 1

Pada Gambar 5.10 mendefinisikan *library* yang disertakan, Wire.h digunakan untuk menambahkan fungsi pembacaan, Time.h digunakan untuk fungsi waktu, dan *library* dari device tersebut, disana terdapat monthName atau nama bulan yang setiap angkanya mewakili nama-nama bulan dari januari sampai desember. tmElements_t diubah variabel menjadi tm agar mudah dipanggil dalam bentuk objek.

```
12 void setup() {
13   bool parse=false;
14   bool config=false;
16   // get the date and time the compiler was run
17   if (getDate(__DATE__) && getTime(__TIME__)) {
18     parse = true;
19     // and configure the RTC with this info
20     if (RTC.write(tm)) {
21       config = true;
22     } }
```

Gambar 5.11 *Script* Konfigurasi RTC pada Arduino Part 2

Gambar 5.11 mensetting untuk *parse* dan *config* agar berada dalam keadaan *false*, jika waktu dan tanggal sekarang diinginkan, maka *parse* diubah *true* dan RTC mengaktifkan fungsi penulisan dengan mengubah config pada tm menjadi *true*.

```
24  Serial.begin(9600);
25    while (!Serial) ; // wait for Arduino Serial Monitor
26    delay(200);
27    if (parse && config) {
28      Serial.print("DS1307 configured Time=");
29      Serial.print(__TIME__);
30      Serial.print(", Date=");
31      Serial.println(__DATE__);
32    } else if (parse) {
33      Serial.println("DS1307 Communication Error :-{");
34      Serial.println("Please check your circuitry");
35    } else {
36      Serial.print("Could not parse info from the compiler, Time=\"");
37      Serial.print(__TIME__);
38      Serial.print("\", Date=\"");
39      Serial.print(__DATE__);
40      Serial.println("\"");
41    }
42  }
```

Gambar 5.12 *Script* Konfigurasi RTC pada Arduino Part 3

Gambar 5.12 adalah *script* untuk bagian void setup, dimana RTC akan melakukan setup setelah serial monitor pada Arduino aktif, jika *parse* dan *config* nilainya didapatkan maka, waktu konfigurasi yang ada ditampilkan, jika hanya *parse*, maka muncul pesan alat tidak terhubung, dan jika keduanya tidak aktif, maka muncul pesan waktu tidak dapat diambil dari RTC.

Terkadang beberapa pesan error ditampilkan dikarenakan bahwa baterai dari RTC *error* atau karena memang sudah harus diganti, oleh karena itu untuk lebih aman pemakaian RTC menggunakan voltase input ke pin.

```
43  void loop() {
44    if (RTC.read(tm)) {
45    Serial.print("Ok, Time = ");
46    Serial.print(tm.Hour);
47    Serial.print(':');
48    Serial.print(tm.Minute);
49    ttemp1=tm.Minute;
50    Serial.print(':');
51    Serial.print(tm.Second);
52    ttemp=tm.Second;
53    Serial.print(", Date (D/M/Y) = ");
54    Serial.print(tm.Day);
55    Serial.print('/');
56    Serial.print(tm.Month);
57    Serial.print('/');
58    Serial.print(tmYearToCalendar(tm.Year));
59    Serial.println();
60   } else {
61    if (RTC.chipPresent()) {
62      Serial.println("The DS1307 is stopped.  Please run the SetTime");
63      Serial.println("example to initialize the time and begin running.");
64      Serial.println();
65    } else {
66      Serial.println("DS1307 read error!  Please check the circuitry.");
67      Serial.println();
68    }
69    delay(1000);
70   }
71   delay(1000);
72  }
```

Gambar 5.13 *Script* Konfigurasi RTC pada Arduino Part 4

Pada gambar 5.13 berisi proses menampilkan data dari RTC ke serial monitor dimana nilai waktu disimpan dalam variabel milik tm. Karena menit dan detik diperlukan dalam pengujian, kedua nilai ini juga disimpan dalam variabel lain sehingga setiap pembacaan waktu variabel akan selalu update. Dan apabila RTC

terpasang namun belum terkonfigurasi, maka diminta untuk menjalankan konfigurasi waktu terlebih dahulu. Dan jika bukan keduanya maka RTC memang belum terhubung.

```
73  bool getTime(const char *str)
74  {
75    int Hour, Min, Sec;
76
77    if (sscanf(str, "%d:%d:%d", &Hour, &Min, &Sec) != 3) return false;
78    tm.Hour = Hour;
79    tm.Minute = Min;
80    tm.Second = Sec;
81    return true;
82  }
```

Gambar 5.14 *Script* Konfigurasi RTC pada Arduino Part 5

Pada Gambar 5.14 konfigurasi Boolean untuk pengambilan waktu pada RTC, dimana nilai tm.Hour, tm.Minute dan tm.Second nilainya diambil dari hasil scan oleh RTC module.

```
83  bool getDate(const char *str)
84  {
85    char Month[12];
86    int Day, Year;
87    uint8_t monthIndex;
88
89    if (sscanf(str, "%s %d %d", Month, &Day, &Year) != 3) return false;
90    for (monthIndex = 0; monthIndex < 12; monthIndex++) {
91      if (strcmp(Month, monthName[monthIndex]) == 0) break; }
92    if (monthIndex >= 12) return false;
93    tm.Day = Day;
94    tm.Month = monthIndex + 1;
95    tm.Year = CalendarYrToTm(Year);
96    return true;}
```

Gambar 5.15 *Script* Konfigurasi RTC pada Arduino Part 6

Pada gambar 5.15 konfigurasi Boolean untuk pengambilan tanggal, dimana bulan terdefinisi menjadi 12 nama seperti yang sudah dijelaskan pada Gambar 5.10. dan kalender yang digunakan adalah kalender masehi.

5.3.2.3 Konfigurasi Sensor MQ-135 dan MG-811

Sensor terdiri dari 2 jenis yaitu MQ-135 dan MG-811 dimana sensor ini terhubung ke Arduino mega sesuai dengan perancangan sistem pada bab sebelumnya, sensor membaca kondisi udara disekitarnya dan membagi nilainya menjadi 2, yaitu dalam besaran lebih dari 256 dan kurang dari 256, apabila lebih dari 256 maka tiap nilai sensor melebihi 256 maka akan dikonversi menjadi 1 dan berlaku kelipatannya, sehingga diperlukan algoritma tambahan untuk membaca nilai kedua sensor ini. Adapun *script* dari sensor ini dapat dilihat pada Gambar 5.16 sampai dengan Gambar 5.19.

```
1   .....
2   unsigned int data1;
3   unsigned int data2;
4   unsigned int data6;
5   unsigned int data3;
6   unsigned int data4;
7   unsigned int data5;
8   .....
9   int i=0;
10  int j=0;
11  int k=0;
12  .....
```

Gambar 5.16 *Script* Konfigurasi MQ-135 dan MG-811 Part 1

Pada Gambar 5.16 terdapat 6 buah unsigned int, variabel ini digunakan untuk menyimpan data dari sensor, satu buah sensor memerlukan 2 tempat data sementara dan 1 tempat data final. Sedangkan int I,j,k adalah variabel yang digunakan untuk perulangan.

Dari *script* diatas digunakan 6 buah variable unsigned int untuk menyimpan nilai data yang dibaca oleh sensor, dimana data1 dan data3 untuk kelipatan 256 dari sensor, data2dan data4 untuk nilai satuan dari sensor dan data5 dan data6 untuk nilai total dari penjumlahan data1 dengan data2 dan data3 dengan data4. Data1, data2 dan data6 milik MG-811 dan data3, data4 dan data5 milik MQ-135, untuk penjelasan data dapat dilihat pada Gambar 5.17 dan Gambar 5.18.

```
13  Void loop(){
14  .....
15  Serial1.write(0x41);
16   if (Serial1.available()){
17     if (i == 0){
18       data1=256 * Serial1.read();
19       i = 1;
20     }
21     else{
22       data2=Serial1.read();
23       i = 0;
24       data6 = data1 + data2;
25     }
```

Gambar 5.17 *Script* Konfigurasi MQ-135 dan MG-811 Part 2

Gambar 5.17 adalah pengambilan data milik MG-811 dimana perintah mengambil datanya Serial1.write(0x41); karena menggunakan Serial1, data dikirim sebanyak 2 kali Melalui Serial1.read(); data pertama diberikan algortima dengan dikalikan 256, dan disimpan dalam variabel data1. Dan data2 menyimpan variabel ke 2 dari data yang diambil, setelah keduanya lengkap, data disatukan dalam variabel data6 seperti yang sudah dijelaskan sebelumnya.

```
25  Serial2.write(0x41);
26   if (Serial2.available()){
27     if (j == 0){
28       data3=256 * Serial2.read();
29       j = 1;
30     }
31     else{
32       data4=Serial2.read();
33       j = 0;
34       data5 = data3 + data4;
35     }
36   }
```

Gambar 5.18 *Script* Konfigurasi MQ-135 dan MG-811 Part 3

Gambar 5.18 kurang lebih sama seperti penjelasan Gambar 5.17, hanya saja disini yang digunakan adalah Serial2 dan variabel untuk data pertama yang dikali

dengan 256 adalah data3, variabel data ke 2 adalah data4, dan total untuk kedua data tersebut adalah data5.

```
38  if (Serial1.available() && Serial2.available()){
39    int a[2] = {data6, data5};
40    Serial.print("CO2 ");
41    Serial.println(a[0]);
42    Serial.print("N ");
43    Serial.println(a[1]);
44    delay (1000);
45    temp=a[0];
46    temp1=a[1];
47  .....
48  }
```

Gambar 5.19 *Script* Konfigurasi MQ-135 dan MG-811 Part 4

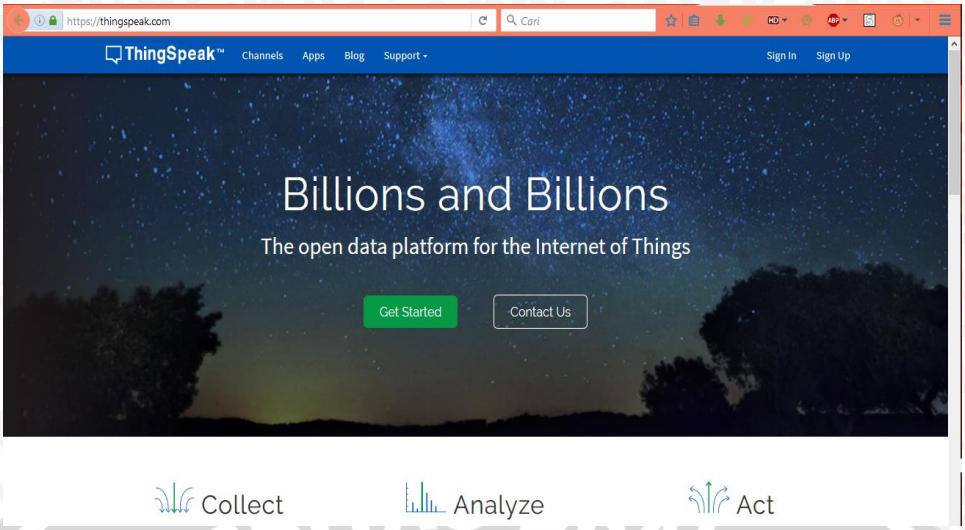
Gambar 5.19 adalah konfigurasi untuk menampilkan nilai value dari sensor ke serial monitor, dimana total dari nilai masing-masing sensor didapatkan dan disimpan kedalam sebuah variabel bertipe array dan akan ditampilkan pada serial monitor. Data yang ada dalam array itulah yang akan dimasukkan kedalam pengiriman nantinya.

5.3.2.4 Konfigurasi Thingspeak

Thingspeak adalah sebuah platform yang menyediakan layanan eksklusif yang ditargetkan untuk pembangunan aplikasi IOT, dimana data dikumpulkan secara real-time dan divisualisasikan dalam bentuk grafik, thingspeak juga memiliki kemampuan untuk membuat plugin dan aplikasi untuk berkolaborasi dengan layanan web, jaringan sosial dan API lainnya.

Untuk menggunakan thingspeak, "user" harus melakukan pendaftaran dan membuat channel untuk wadah penampungan data yang akan digunakan. Setiap channel memikili konfigurasi masing-masing dan tidak akan menggangu channel lainnya walaupun user menggunakan dua channel atau lebih secara berbarengan, adapun konfigurasi thingspeak adalah sebagai berikut

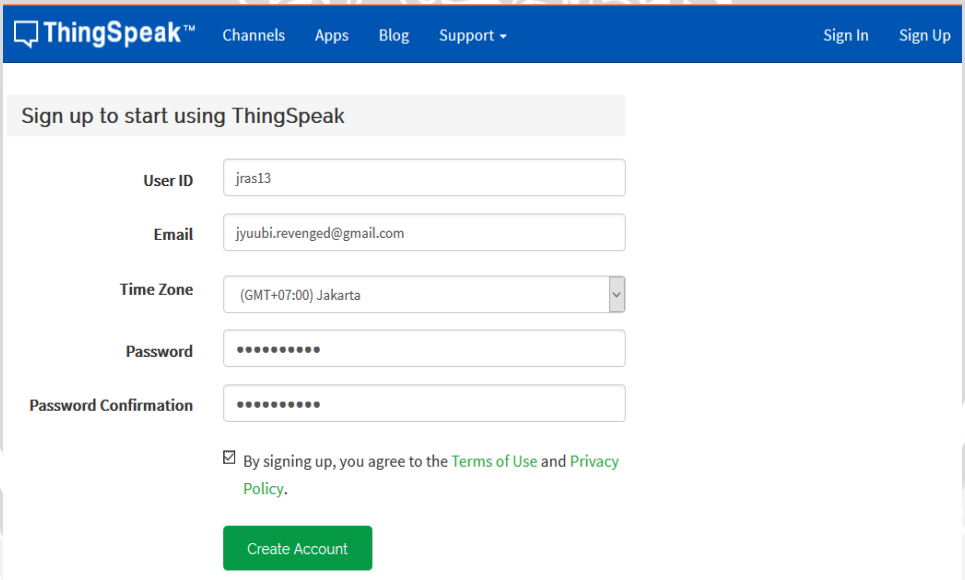
1. Buka http://www.thingspeak.com dan kemudian klik "*Sign Up*"



Gambar 5.20 Tampilan Thingspeak.com

Disini penulis lebih memilih menggunakan Thingspeak karena selain ringan, thingspeak juga mendukung pengembangan IOT dan pengembangan ESP.

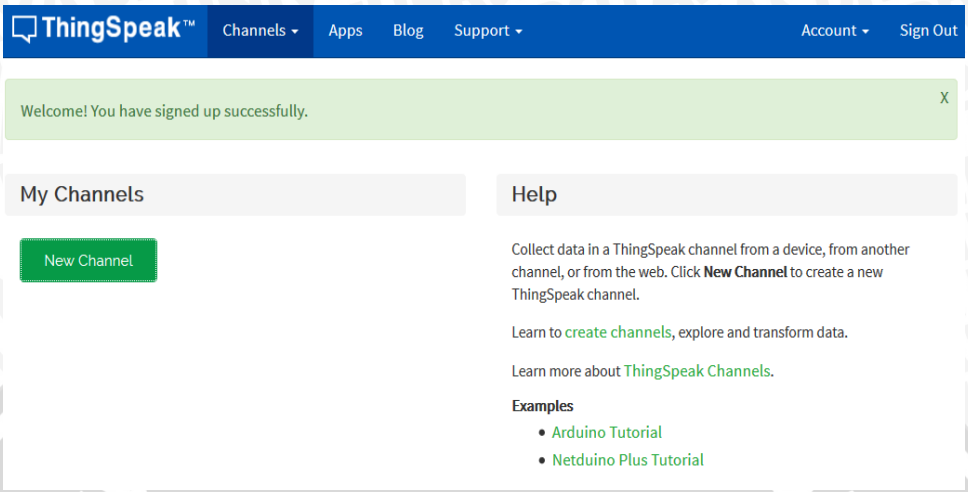
2. Isi *"User ID"*, *"Email"* dan *"password"*, untuk *"Time Zone"* disetting sesuai dengan wilayah dan *"accept Terms of Use and Privacy Policy"*, kemudian tekan tombol *"create account"*.



Gambar 5.21 Thingspeak *Sign Up*

"User id", *"email"* dan *"password"* digunakan sebagai pengamanan untuk akun dan data pengguna, dan merupakan hal umum yang wajib diisi, sedangkan untuk *"Time Zone"* disini kenapa harus diisi dikarenakan pengguna thingspeak tidak hanya dari satu daerah, namun banyak. dan juga dikarenakan untuk membuat kinerja web Thingspeak sendiri agar lebih ringan karena *"Time Zone"* tidak mengambil informasi dari user namun karena sudah tersedia.

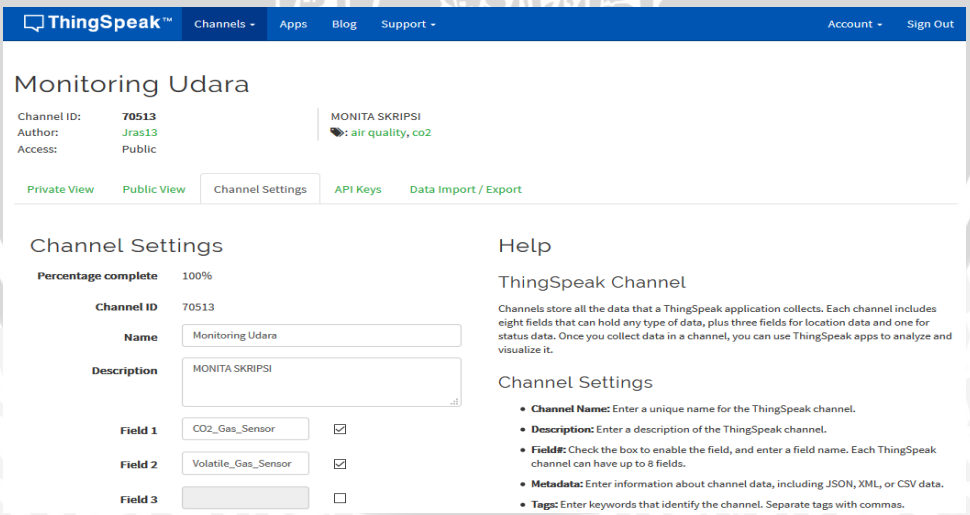
3. Setelah "*sign up*" berhasil, akan tampil list *channel* yang dimiliki, karena akun belum memiliki *channel* apapun maka tidak ada *channel* yang tampil, tekan tombol "*new channel*" untuk membuat *channel* baru.



Gambar 5.22 Penambahan *Channel* Baru Thingspeak

Channel disini digunakan untuk mendaftarkan saluran atau dalam gambaran kasarnya, satu orang dapat membuat banyak projek, setiap projek memiliki *channel*-nya masing-masing, dan untuk memudahkan *subscriber* untuk menemukan data.

4. isi "*Name*" untuk nama *channel*, Description untuk pendeskripsian *channel*, *field* 1 sampai *field* 8 adalah nama *field* untuk data yang akan diterima dan ditampilkan pada *channel*, *field* maksimal per *channel* adalah 8 *field*, karena pada penelitian ini memakai dua sensor maka hanya 2 *field* yang digunakan.

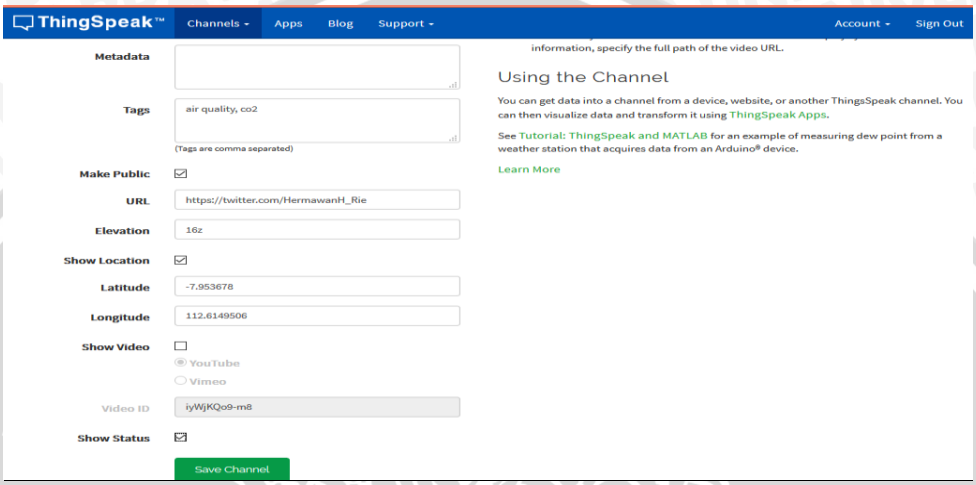


Gambar 5.23 Konfigurasi *Channel* Thingspeak 1

Ketiga data ini adalah informasi minimal dari sebuah *channel*, walaupun sisanya dikosongkan, *channel* dapat digunakan.

5. *"Metadata"* diisi untuk menginformasikan tentang data pada *channel* termasuk JSON, XML or CSV data, karena pada penelitian tidak menggunakan

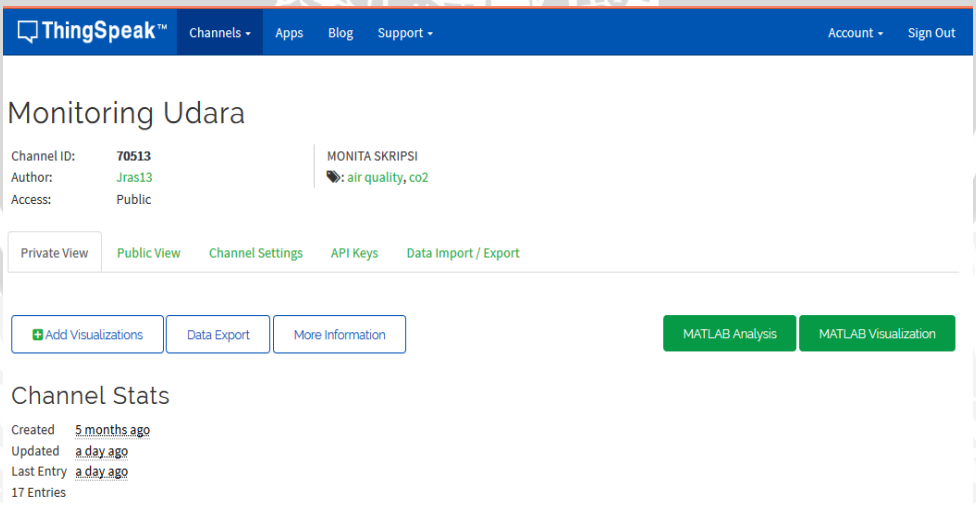
ketiganya, maka sengaja tidak diisi. "*Tags*" disini digunakan untuk memudahkan pencarian *channel* oleh publik, pada "*checkbox*" dibawah digunakan untuk mempublikasikan *channel* ini atau hanya untuk privat. "*Elevation*" adalah tinggi posisi sensor dari ketinggian tanah dalam ukuran meter, untuk "*show location*" terdapat "*latitude*" dan "*longitude*", koordinat 2D yang dapat diambil dengan cara melihat "*my position*" pada *Google Maps* untuk memposisikan dimana posisi alat dan agar mudah ditemukan. Selanjutnya tekan tombol *save channel*.



Gambar 5.24 Konfigurasi *Channel* Thingspeak 2

Gambar 5.23 dan 5.24 adalah konfigurasi *Channel* thingspeak, namun konfigurasi yang dilakukan disini hanyalah konfigurasi untuk bagian visual.

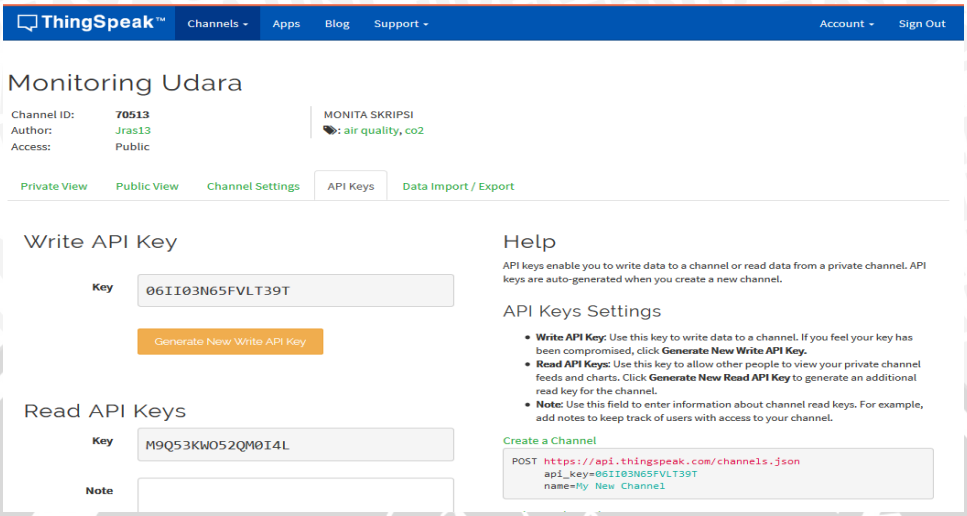
6. Setelah *channel* selesai dibuat, tampilannya sama seperti pada Gambar 5.12 dimana isinya masih kosong, kemudian masuk ke *tab* "API *Keys*".



Gambar 5.25 Tampilan Channel Thingspeak

Tab "API *keys*" adalah Tab yang berisi Api *key* yang diperlukan sebagai autentikasi pertukaran data tanpa harus *log in*.

7. Pada *tab* "API *Keys*" terdapat *Write* API dan *Read* API dimana *Write* API digunakan untuk menerima data dan *Read* API digunakan untuk meminta data dari *channel*, disini *Write* API lah yang digunakan sebagai API *Key* yang menghubungkan sistem dengan *web* thingspeak.



Gambar 5.26 Thingspeak API *Key*

Setelah didapatkan API *key* Thingspeak, salin API *key* tersebut untuk digunakan dalam *coding* Arduino, dan didapatkan *script* untuk pengiriman data ke ThingSpeak, adapun konfigurasi untuk penggunaan API *key* Thingspeak terdapat pada Gambar 5.18

```
1  #include <JeeLib.h>
2  #include <DS1307RTC.h>
3  #include <TimeLib.h>
4  #include <Time.h>
5  #include <SoftwareSerial.h>
6  #include <Wire.h>
7  #define DEBUG true
8  ISR(WDT_vect) { Sleepy::watchdogEvent(); }
9  ....
10 String apiKey = "06II03N65FVLT39T";
11 String twitKey = "LAJD1WPLYO61F8YD";
12 .....
```

Gambar 5.27 *Script* Konfigurasi Thingspeak pada Arduino Part 1

Gambar 5.27 adalah konfigurasi untuk *library* dan *key* dari Thingspeak

```
13  Void Loop (){
14  ....
15  if (Serial1.available() && Serial2.available()){
16   ....
17   if ((ttemp==30)||(ttemp==31)){
18    TestThingspeak();
19     TwitterTweet();
20    Sleepy::loseSomeTime(21000);
21    Serial.println("Autorestart Program!!");
22    delay (4000);
23    asm volatile ("  jmp 0");
24    }
25  }
26    }}
```

Gambar 5.28 *Script* Konfigurasi Thingspeak pada Arduino Part 2

Gambar 5.28 adalah potongan dari seluruh *void loop*, dimana isinya adalah pembacaan sensor, setelah dibaca dan dipastikan datanya asli dengan *serial available*, dan hitungan detik menunjukkan 30 atau 31 maka mulai pengirimkan data.

```
27  void TestThingspeak(){
28   kirimPerintah("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n",
29  16000, DEBUG);
30
31    if(esp8266.find("Error")){
32     Serial.println("AT+CIPSTART error");
33     return;
34    }
```

Gambar 5.29 *Script* Konfigurasi Thingspeak pada Arduino Part 3

Pengiriman data dijalankan dalam *void* TestThingspeak, disini perintah AT+CIPSTART atau perintah membuat jalur koneksi dilakukan, dengan jalur HTTP atau port 80 dengan *socket* TCP tujuan IP 184.106.153.149 atau IP thingspeak.com, dengan *delay* 16000 sebagai *timeout* untuk perintah AT+CIPSTART. Jika esp8266 tidak terhubung dengan mikrokontroler atau rusak, maka munculkan pesan *Error*.

```
35    String getStr = "GET /update?api_key=";
36    getStr += apiKey;
37    getStr +="&field1=";
38    getStr += String(temp);
39    getStr +="&field2=";
40    getStr += String(temp1);
41    getStr += "\r\n\r\n";
42
43    // send data length
44    String cmd = "AT+CIPSEND=";
45    cmd += String(getStr.length());
46    esp8266.println(cmd);
```

Gambar 5.30 *Script* Konfigurasi Thingspeak pada Arduino Part 4

Gambar 5.30 adalah pembungkusan data dan cara mengirim data melalui ESP8266, yaitu dengan perintah HTTP "GET /update?api_key=" yang ditambahkan dengan *key* sebelumnya,dan diikutkan dengan variabel php untuk pengisian *field* layaknya pengiriman ke *form* HTTP, dan diakhiri dengan \r\n\r\n, disini \r\n dibuat 2 karena sebaga penanda berakhirnya perintah HTTP dan perintah yang akan menampung pesan ini yaitu AT+CIPSEND. Perintah ini digunakan untuk mengirim data dengan cara ditambahkan dengan variabel yang mengambil data per karakter sesuai dengan panjang perintah HTTP, dan kemudian dikirimkan ke ESP8266.

```
48    if(esp8266.find(">")){
49      esp8266.print(getStr);
50    }
51    else{
52      esp8266.println("AT+CIPCLOSE");
53      // alert user
54      Serial.println("AT+CIPCLOSE");
55    }
56   delay (16000);
57  }
```

Gambar 5.31 *Script* Konfigurasi Thingspeak pada Arduino Part 5

Gambar 5.31 adalah lanjutan dari gambar 5.30 dimana jika esp8266 menemukan > atau \r\n maka cetak apa yang tertulis pada perintah AT+CUPSEND.

Jika tidak, tutup jalur pengiriman dengan AT+CIPCLOSE, dan kemudian diberikan *delay* 16 detik untuk akuisisi data ke thingspeak.

```
60  String kirimPerintah(String perintah, const int timeout, boolean debug)
61  {
62    String response = "";
63    esp8266.print(perintah);
64    long int time = millis();
65    while( (time+timeout) > millis()) {
66    while(esp8266.available()) {
67    char c = esp8266.read();
68    response+=c;
69  }
70  }
71  if(debug) {
72   Serial.print(response);
73  }
74  return response;
75  }
```

Gambar 5.32 *Script* Konfigurasi Thingspeak pada Arduino Part 6

Gambar 5.32 isi dan penjelasannya sama dengan Gambar 5.9 dan tidak ada yang berubah. Pada Gambar 5.27 sampai Gambar 5.32 dapat dilihat, API *key* digunakan untuk melakukan *update* atau *upload* data, jadi setelah data selesai dibaca oleh sensor dan nilai didapatkan kemudian disimpan kedalam variabel bertipe *array*, data akan dikemas dalam bentuk *char* agar dapat dikirim melalui ESP8266, ESP8266 membuka jalur koneksi ke Thingspeak melalui IP *Address*, bukan melalui nama atau DNS, jalur yang dibuka adalah port 80 atau HTTP, dan *socket*-nya adalah TCP.

Setelah jalur koneksi terbuka, data akan dikirim, agar dapat diterima oleh thingspeak, data harus sesuai dengan format yang ditentukan, yaitu GET /update?api_key=<apikey>&field1=<datasensor1>&field2<datasensor2>\r\n\r\n dimana dengan perintah HTTP yaitu GET melakukan *update* dengan API *key* yang ditentukan, dengan target field 1 yang akan diisi dengan data sensor1 dan target field 2 dengan data sensor2 dan \r\n\r\n adalah default dari perintah yang ada.

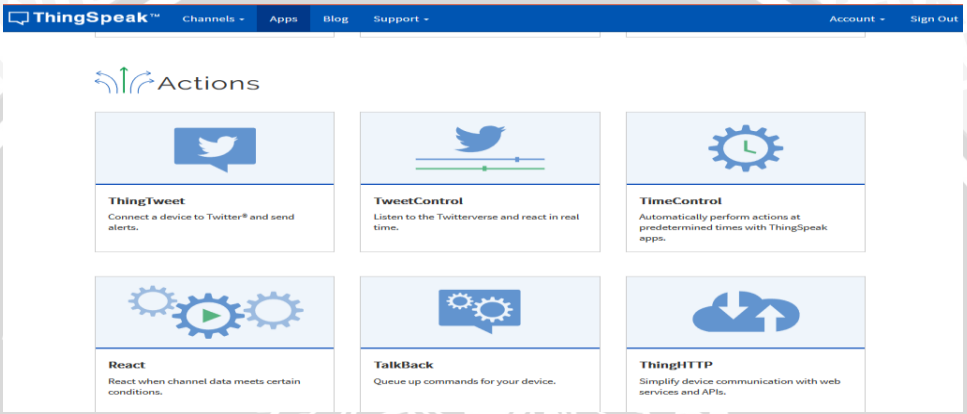
ESP8266 mengirimkan data dengan cara melihat length dari data yang akan dikirimkan, sehingga panjang data harus dihitung terlebih dahulu dengan perintah String(getStr.length()) dan kemudian dikirimkan dengan perintah AT+CIPSEND=. Jika perhitungan karakter sudah selesai maka data akan dikirimkan. Lama waktu

pengiriman data dari ESP8266 ke thingspeak adalah sekitar 16 detik, setelah data terkirim, koneksi akan ditutup oleh ESP8266 dengan perintah AT+CIPCLOSE.

5.3.2.5 Konfigurasi Twitter

Konfigurasi ini diperlukan untuk mengirimkan hasil pembacaan data yang didapatkan dari sistem agar dapat ter-*posting* di twitter dan dapat dibaca oleh para pengguna akun twitter yang mem-*follow* akun tersebut. Adapun konfigurasi yang dilakukan adalah sebagai berikut

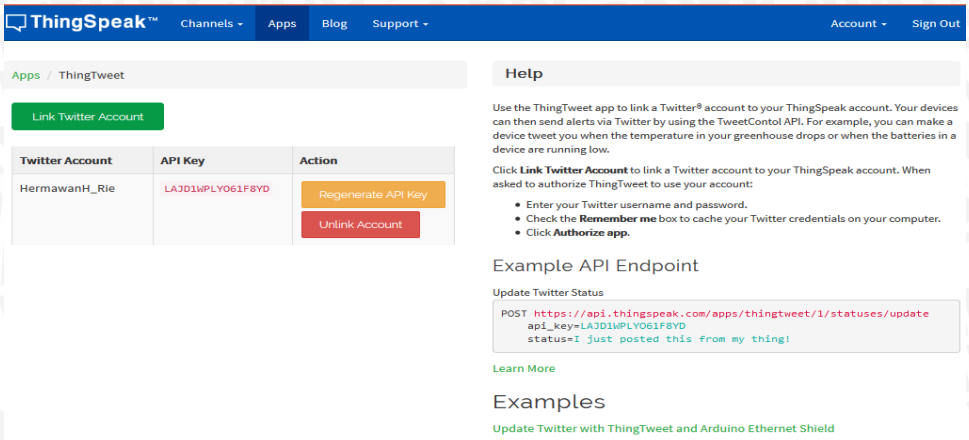
1. Pada tab Thingspeak, terdapat tab "*Apps*", masuk ke tab tersebut dan akan muncul pilihan menu "*Actions*", pilih "ThingTweet".



Gambar 5.33 Apps Menu pada Thingspeak

Pada Gambar 5.33 terdapat beberapa pilihan menu seperti "*ThingTweet*" untuk menghubungkan Thingspeak dengan Akun Twitter, "*TweetControl*" untuk melakukan update status ke twitter di waktu yang ditentukan, "*TimeControl*" untuk mengeksekusi sebuah perintah apabila waktu yang ditentukan telah tiba, "*React*" untuk mengeksekusi sebuah proses apabila data memenuhi sebuah kondisi, "*Talkback*" untuk memerintahkan instruksi ke sistem melalui thingspeak, dan "*ThingHTTP*" untuk memudahkan komunikasi dari sistem menuju Web Service atau API tertentu dari sebuah website HTTP. Namun pada penelitian ini yang diperlukan adalah sistem terhubung ke sosial media twitter, sehingga menu yang dipilih adalah ThingTweet.

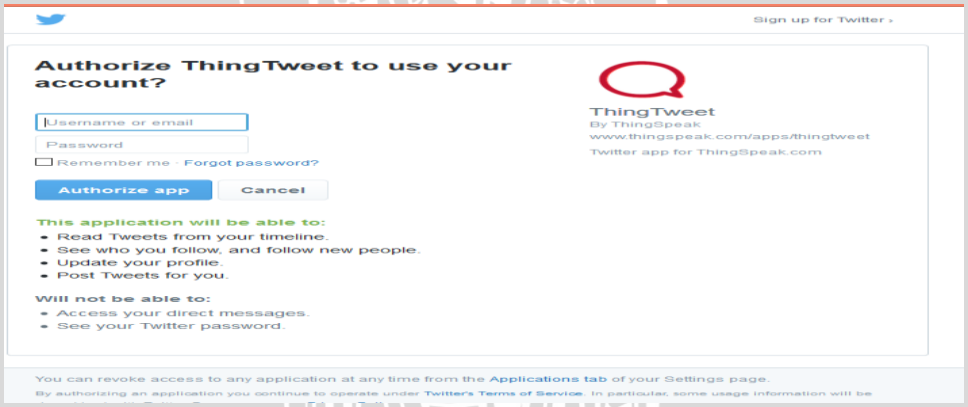
2. Seletah masuk ke *menu Apps* ThingTweet, terdapat tombol hijau yang bertuliskan "*Link Twitter Account*", tekan tombol tersebut dan akan terbuka jendela tab baru di *browser*.



Gambar 5.34 Menu Apps Thingspeak

Link bertujuan untuk menghubungkan twitter dengan thingspeak, dan dapat mengirimkan autentikasi atas user melalui thingspeak untuk memposting ke twitter.

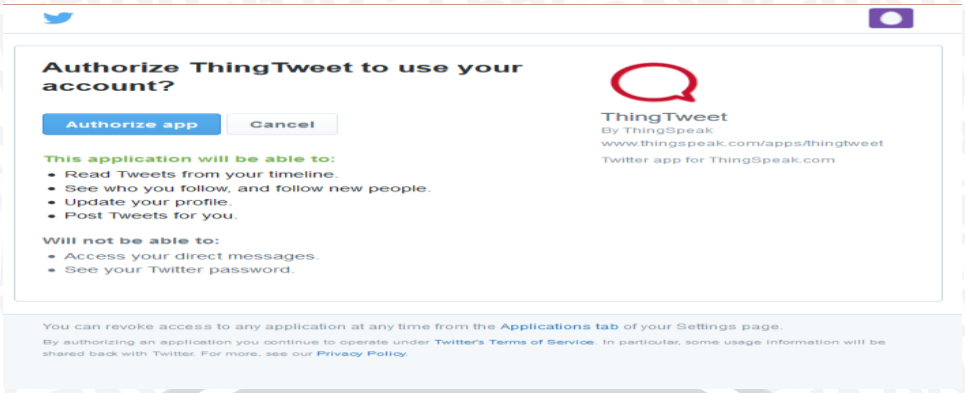
3. Pada laman ini diminta untuk login ke halaman twitter yang sudah dimiliki atau dibuat sebelumnya, dan meminta autentikasi untuk mengizinkan aplikasi ThingTweet melakukan posting selaku pengguna asli.



Gambar 5.35 Autentikasi ThingTweet

Autentikasi aplikasi thingspeak tidak berbeda dari autentikasi aplikasi twiter lainnya. Syaratnya adalah akun yang ingin di akui harus akun yang sudah terdaftar.

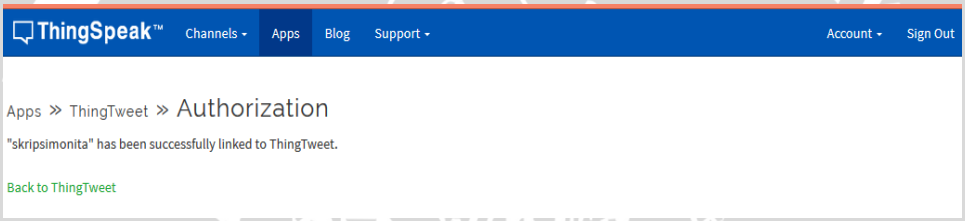
4. Apabila sudah login, autentikasi yang diminta hanya cukup dengan menekan tombol biru yang tersedia setelah melakukan login



Gambar 5.36 Konfirmasi Autentikasi ThingTweet

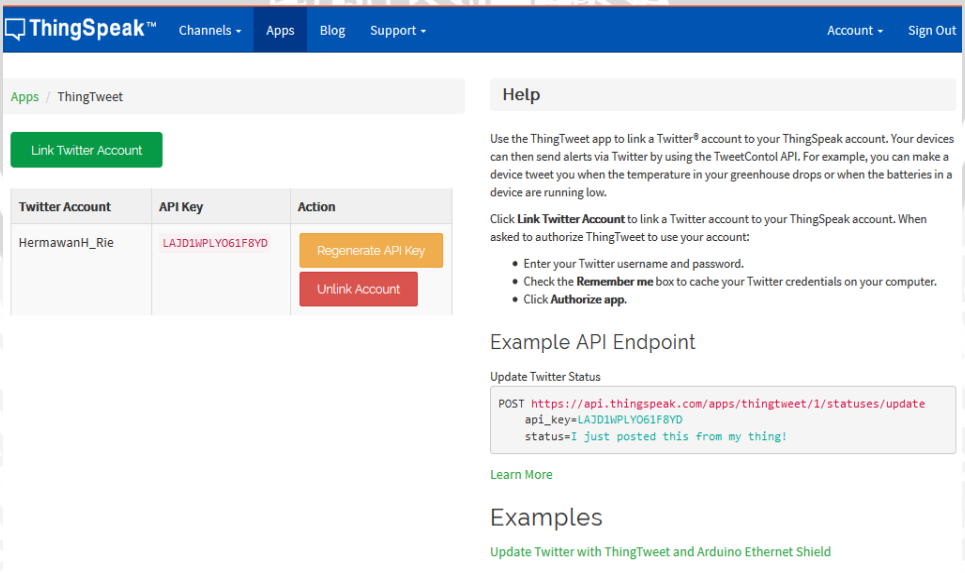
Apabila sudah terdaftar dan login ke twitter, autentikasi dapat dilakukan. Apabila sudah autentikasi, halaman twitter dapat ditutup.

5. Apabila autentikasi selesai, pada thingspeak akan muncul pesan bahwa autentikasi akun twitter telah berhasil, kemudian kembali ke ThingTweet apps untuk melihat apakah akun sudah tersedia.



Gambar 5.37 Autentikasi ThingTweet berhasil

6. Akun twitter yang sudah terhubung akan terlist ada ThingTweet, dan setiap akun memiliki API *key* tersendiri, api *key* ini digunakan untuk mengirimkan status pesan ke twitter melalui thingspeak.



Gambar 5.38 *List Linked Account* untuk ThingTweet

Untuk konfigurasi ThingTweet *script*-nya hampir sama dengan konfigurasi thingspeak pada Gambar 5.27 sampai Gambar 5.32, hanya saja *key* yang digunakan adalah *key* yang ada pada ThingTweet, isi dari tweet dan perintah untuk melakukan updatenya lah yang berbeda. Perbedaan tersebut dapat dilihat pada Gambar 5.39 sampai 5.41

```
1  void TwitterTweet(){
2  tweet = "Kadar_CO2_"+String(temp)+"_PPM_dan_Kadar_Zat_Berbahaya_"+String(temp1) +"_PPM";
3   kirimPerintah("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n", 16000, DEBUG); //cek koneksi ke thingspeak
4   if(esp8266.find("Error")){
5    Serial.println("AT+CIPSTART error");
6    return;
7   }
```

Gambar 5.39 *Script* Konfigurasi ThingTweet Twitter Part 1

Disini isi tweet dideskripsikan atau diinputkan dalam sebuah variabel

```
8  String getStr1 = "GET /apps/thingtweet/1/statuses/update?api_key=";
9    getStr1 += twitKey;
10   getStr1 +="&status=";
11   getStr1 += tweet;
12   getStr1 +="\r\n\r\n";
13
14    String cmd = "AT+CIPSEND=";
15   cmd += String(getStr1.length());
16   esp8266.println(cmd);
```

Gambar 5.40 *Script* Konfigurasi ThingTweet Twitter Part 2

Pada konfigurasi untuk koneksi ke twitter yang menggunakan aplikasi tersedia dari thingspeak, perintah yang digunakan adalah GET /apps/thingtweet/1/statuses/update?api_key=<twitkey>&status=<tweet>\r\n\r n. Dimana tujuan dari GET adalah *apps* thingtweet dan melakukan status *update* dengan autentikasi user yang memiliki *key* yang sama dengan yang diikutkan pada peritah tersebut, dan isi status yang sudah disiapkan dalam Arduino seperti pada gambar 5.39

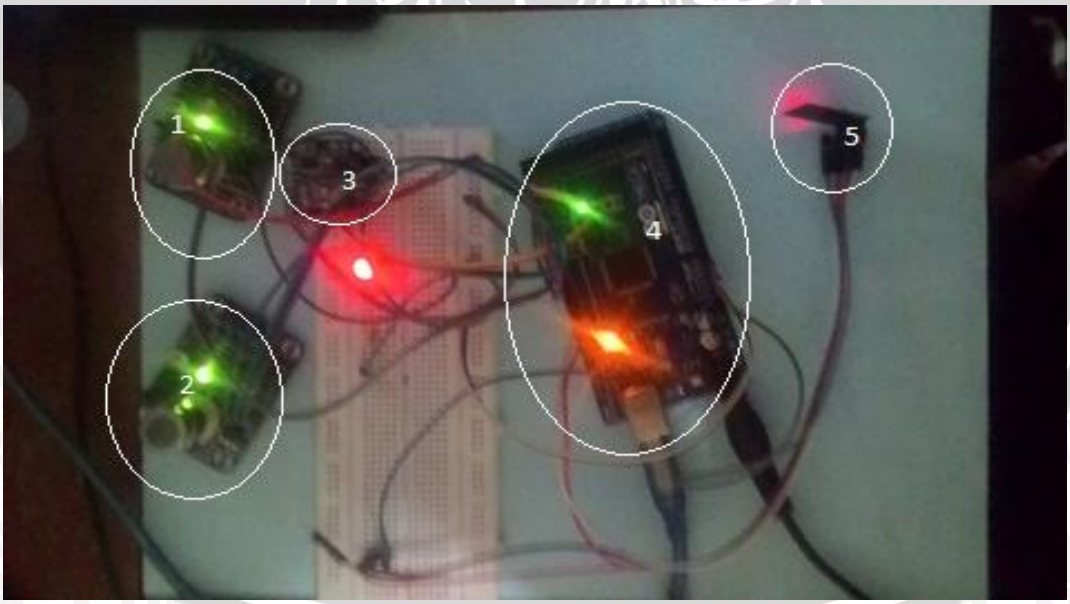
```
18  if(esp8266.find(">")){
19   esp8266.print(getStr1);
20  }
21  else{
22   esp8266.println("AT+CIPCLOSE");
23   // alert user
24   Serial.println("AT+CIPCLOSE");
25  }
26  Serial.print("Twitter Status Updated!");
27  delay (16000);
28  }
```

Gambar 5.41 *Script* Konfigurasi ThingTweet Twitter Part 3

Apabila data telah dikirim, akan muncul pesan "*Twitter Status Updated*!" Pada serial monitor.

5.3.2.6 Hasil Alat

Adapun hasil jadi alat atau hardware dapat dilihat pada Gambar 5.42



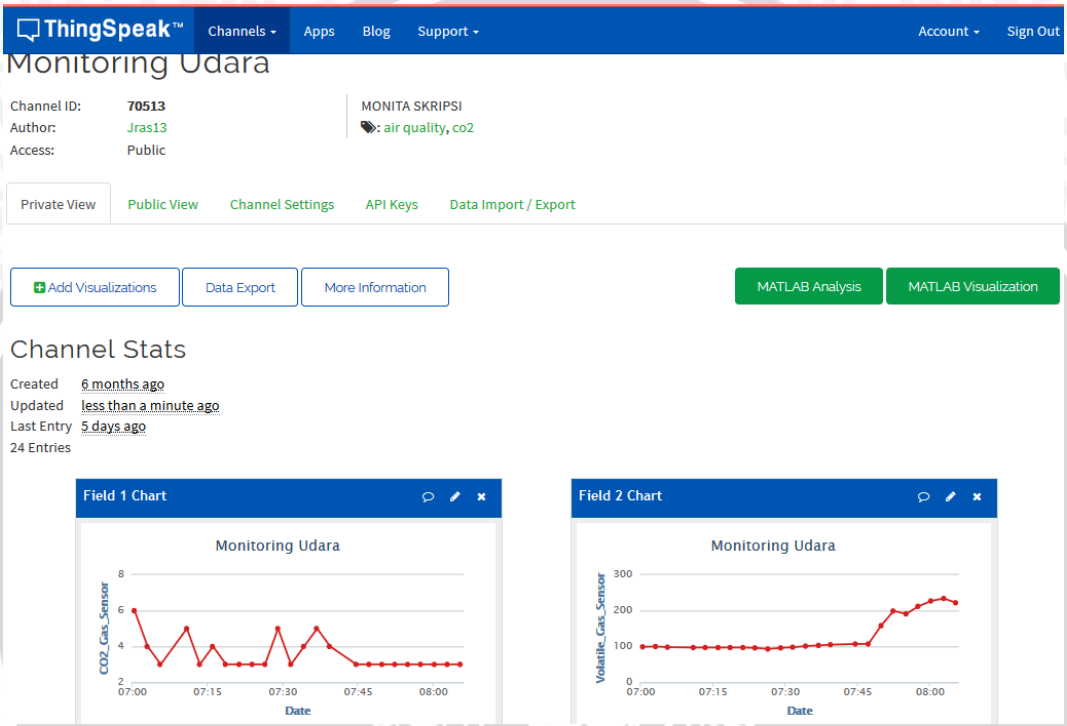
Gambar 5.42 Hasil Jadi Alat sesuai perancangan

Pada Gambar 5.42 terdapat angka dan lingkaran, berikut penjelasan mengenai obyek yang ditandai

1. Sensor MQ-135 yang bekerja dengan ditandai indikator hijau.
2. Sensor MG-811 yang bekerja dengan ditandai indikator hijau.

3. RTC Modul tidak memiliki indikator, sehingga ditambahkan LED berwarna merah.
4. Arduino Mega 2560 sebaga mikrokontroler.
5. ESP8266 yang dalam keadaan standby, ditandai dengan led merah, apabila dalam keadaan bekerja, led biru akan nyala dan mati secara bergantian

Adapun Hasil dari data yang sudah masuk kedalam thingspeak dapat dilihat pada Gambar 5.43



Gambar 5.43 Thingspeak dengan *Graphic Data*

Grafik yang berada dikiri adalah kumpulan data yang dikirim ke dalam Field1, dengan nilai yang berisikan semua data CO_2 dari sensor MG-811, sedangkan Grafik yang berada dikanan adalah kumpulan data yang dikirim ke dalam Field2, dengan nilai yang berisikan semua data Air Quality MQ-135. Setiap satu data masuk maka satu titik di grafik akan muncul dan juga sebuah tweet akan masuk ke dalam twitter, berikut contoh tweet yang terkirim ke dalam twitter dapat dilihat pada Gambar 5.44

Gambar 5.44 Twitter Tweet

Sebuah tweet diposting atas nama akun yang sudah diberi autentikasi, dan menuliskan post sesuai dengan yang tertulis pada Gambar 5.39.

BAB 6 PENGUJIAN

Pada bab ini akan membahas hal yang terkait dengan pengujian seperti skenario atau prosedur pengujian, proses pengujian serta analisis terhadap hasil data dari pengujian yang telah dilakukan.

6.1 Pengujian Konektivitas ESP8266

Adapun pengujian konektivitas ESP8266 dilakukan dengan tahap-tahap berikut.

6.1.1 Tujuan Pengujian

Tujuan dari penelitian ini untuk mengetahui apakah ESP8266 dapat terhubung ke jaringan internet dengan membuatnya terhubung ke sebuah *Access Point* yang menyediakan akses internet dan layanan DHCP. Pengujian ini juga dilakukan karena merupakan kebutuhan fungsional dan non fungsional dari sistem yang dibuat, agar diketahui apakah kebutuhan bisa terpenuhi apa tidak pada hasil analisa.

6.1.2 Prosedur Pengujian

Pengujian ini dilakukan dengan mengkonfigurasi ESP8266 melalui AT *Command* pada Arduino agar dapat terhubung ke *Access Point* tertentu yang telah diproteksi dengan *password* yang juga menyediakan layanan DHCP.

6.1.3 Pelaksanaan Pengujian

Pengujian akan dilakukan dengan menyalakan alat dan menunggu selama kurang lebih 1 menit untuk dapat terhubung ke *Access Point*, mendapatkan IP address dan juga akses internet. Pengujian ini tidak perlu menginputkan perintah apapun pada serial monitor Arduino, karena seluruh konfigurasi untuk ESP8266 sudah diletakkan pada *script* di bagian *Void Setup*.

```
1  SoftwareSerial esp8266(10,11);
2  Void Setup(){
3  esp8266.begin(9600);
4    kirimPerintah("AT+RST\r\n", 2000, DEBUG); //reset modul
5    kirimPerintah("AT+CWMODE=1\r\n", 1000, DEBUG); //Set mode 1
6    kirimPerintah("AT+CWJAP=\"cocacola\",\"SadangWoy12312\"\r\n",
7    3000,DEBUG);  delay(20000);
     kirimPerintah("AT+CIFSR\r\n", 1000, DEBUG); //Mendapatkan IP adress
8    Serial.println("ESP8266 Ready!");}
```

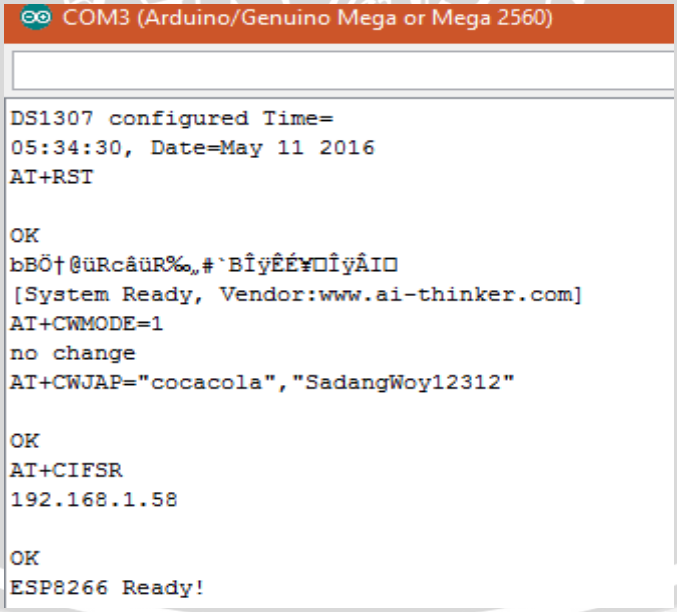
Gambar 6.1 ESP8266 pada Void Setup

Berikut penjelasan lebih lanjut mengenai baris program pada Gambar 6.1

1. Baris 1 pengaturan pin 10 dan 11 agar dapat terhubung secara serial dengan ESP8266 pada pin tx rx milik ESP8266
2. Baris 4 serial *baudrate* untuk ESP8266 diset ke 9600
3. Baris 5 *reset* modul + *delay* 2 detik
4. Baris 6 ubah mode wifi ke mode 1 + *delay* 1 detik
5. Baris 7 buat koneksi ke *Access Point* dengan nama SSID cocacola, *password* SadangWoy12313 + *delay* 3 detik
6. Baris 8 *delay* 20 detik untuk menunggu konfigurasi IP DHCP
7. Baris 9 cek IP address, apabila hasilnya tidak 0.0.0.0 maka sudah terhubung ke *Access Point*
8. Baris 10 menginformasikan melalui serial monitor bahwa ESP8266 sudah selesai dikonfigurasi

6.1.4 Hasil Pengujian dan Analisis

Hasil dari pengujian dapat dilihat pada Gambar 6.2



Gambar 6.2 ESP8266 Terhubung ke *Access Point*

Dari Gambar 6.2 diketahui bahwa ESP8266 berhasil terhubung dengan *Access Point* ditunjukkan dengan munculnya informasi perintah AT *Command* dan *response* dari perintah AT+CWJAP dengan SSID "cocacola" dan *password* "SadangWoy12312" menghasilkan respon OK dan AT+CIFSR dengan respon IP *Address* 192.168.1.58.

Dari hasil percobaan yang dilakukan menunjukkan bahwa ESP8266 dapat terhubung ke *Access Point* dengan melakukan impelmentasi yang sudah dijelaskan pada bab sebelumnya. Adapun hasil pengujian dengan 2 SSID berbeda yang telah dilakukan dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian Konektivitas Berdasarkan SSID

Uji Ke	IP *Address*	SSID
1	192.168.1.58	Cocacola
2	192.168.1.84	S.K.R.I.~P.S.Y~
3	192.168.1.58	Cocacola
4	192.168.1.86	S.K.R.I.~P.S.Y~
5	192.168.1.93	S.K.R.I.~P.S.Y~

Pada SSID Cocacola terdapat kurang dari 10 perangkat yang terhubung, sedangkan pada SSID S.K.R.I.~P.S.Y~ terdapat lebih dari 10 perangkat yang terhubung. Dari Tabel 6.1 dapat dilihat bahwa semakin banyak pengguna pada SSID yang dituju IP Addres akan semakin sering berubah.

6.2 Pengujian Fungsionalitas Sensor

Adapun pengujian fungsionalitas sensor dilakukan dengan tahap-tahap berikut

6.2.1 Tujuan Pengujian

Tujuan dari penelitian ini untuk melihat apakah sensor membaca data dan memberikan nilai sesuai dengan yang diharapkan, Pengujian ini juga dilakukan karena merupakan kebutuhan fungsional dan non fungsional dari sistem yang dibuat, agar diketahui apakah kebutuhan bisa terpenuhi apa tidak pada hasil analisa

6.2.2 Prosedur Pengujian

Pengujian ini dilakukan dengan menyalakan Arduino mega, ESP8266, RTC modul dan sensor MQ-135 dan MG-811 kemudian melakukan pembacaan sensor, apakah sensor dapat bekerja sesuai dengan implementasi dan berjalan sesuai dengan konfigurasi pada bab sebelumnya.

6.2.3 Pelaksanaan Pengujian

Pengujian ini dilakukan dengan cara memerintahkan sensor untuk membaca data selama 10 detik atau lebih dan melihat apakah data terus mendapatkan pembaruan atau tidak.

```
1   Serial1.write(0x41);
2    if (Serial1.available()){
3     if (i == 0){
4       data1=256 * Serial1.read();
5       i = 1;
6     }
7     else{
8       data2=Serial1.read();
9       i = 0;
10      data6 = data1 + data2;
11    }
```

Gambar 6.3 Pembacaan Sensor MG-811 Pada Void Loop

Berikut penjelasan lebih lanjut untuk baris program pada Gambar 6.3 sampai 6.5

1. Baris 1 memerintahkan pembacaan sensor pada serial1
2. Baris 3-11 apabila sensor menyala maka meminta data1, apabila data1 sudah diminta, meminta data2 apabila kedua data sudah ada jumlahkan dan simpan kedalam data6

```
12  Serial2.write(0x41);
13   if (Serial2.available()){
14    if (j == 0){
15      data3=256 * Serial2.read();
16      j = 1;
17    }
18    else{
19      data4=Serial2.read();
20      j = 0;
21      data5 = data3 + data4;
22    }
23   }
```

Gambar 6.4 Pembacaan Sensor MQ-135 Pada Void Loop

3. Baris 12 memerintahkan pembacaan sensor pada serial2

4. Baris 13-23 apabila sensor menyala maka meminta data3, apabila data3 sudah diminta, meminta data4 apabila kedua data sudah ada jumlahkan dan simpan kedalam data5

```
24  if (Serial1.available() && Serial2.available()){
25    int a[2] = {data6, data5};
26    Serial.print("CO2 ");
27    Serial.println(a[0]);
28    Serial.print("N ");
29    Serial.println(a[1]);
30    delay (1000);
```

Gambar 6.5 Menampilkan Hasil Perhitungan Baca Sensor

5. Baris 24 apabila kedua sensor menyala maka masukkan nilai total kedua sensor, yaitu data5 dan data6 kedalam variabel *array* int a,
6. Baris 26-29 melakukan serial print untuk serial monitor dan menginfokan nilai sensor 1 dan sensor 2
7. Baris 30 *delay* 1 detik untuk setiap *loop* pembacaan sensor

6.2.4 Hasil Pengujian dan Analisis

Hasil pengujian sensor dapat dilihat pada Gambar 6.6.

```
Ok, Time = 5:34:57, Date (D/M/Y) = 11/5/2016
Ok, Time = 5:34:58, Date (D/M/Y) = 11/5/2016
Ok, Time = 5:34:59, Date (D/M/Y) = 11/5/2016
CO2 44
N 0
Ok, Time = 5:35:1, Date (D/M/Y) = 11/5/2016
CO2 44
N 225
Ok, Time = 5:35:3, Date (D/M/Y) = 11/5/2016
CO2 43
N 225
Ok, Time = 5:35:5, Date (D/M/Y) = 11/5/2016
CO2 43
N 223
Ok, Time = 5:35:7, Date (D/M/Y) = 11/5/2016
CO2 43
N 223
Ok, Time = 5:35:9, Date (D/M/Y) = 11/5/2016
CO2 43
N 221
Ok, Time = 5:35:11, Date (D/M/Y) = 11/5/2016
CO2 42
N 221
Ok, Time = 5:35:13, Date (D/M/Y) = 11/5/2016
CO2 42
N 218
Ok, Time = 5:35:15, Date (D/M/Y) = 11/5/2016
CO2 41
N 218
Ok, Time = 5:35:17, Date (D/M/Y) = 11/5/2016
CO2 41
N 215
Ok, Time = 5:35:19, Date (D/M/Y) = 11/5/2016
CO2 40
N 215
Ok, Time = 5:35:21, Date (D/M/Y) = 11/5/2016
CO2 40
```

Gambar 6.6 Hasil Pengujian Sensor

Pada Gambar 6.6 dapat dilihat bahwa setiap detik alat memberikan response, pada menit 34:59 memberikan respon waktu, dan pada detik 34:50 memberikan respon nilai CO_2 dan nilai air quality (N) dan nilainya terus berubah ubah seiring berjalannya waktu. Adapun hasil pengambilan data sensor dapat dilihat pada Tabel 6.2.

Tabel 6.2 Pengambilan Data Sensor

Data Ke	CO_2 (PPM)	Air Quality (PPM)
1	35	144
2	26	126
3	20	136
4	12	132
5	9	131
6	10	117
7	10	115
8	14	105
9	14	105
10	14	181
11	13	177
12	13	214
13	13	210
14	10	186
15	10	189
16	9	151
17	9	147
18	9	138
19	9	130
20	9	135
21	8	158
22	8	148

Pada Tabel 6.2 dapat dilihat beberapa data yang diambil oleh sensor Air Quality dan CO_2, CO_2 di udara terlihat lebih kecil dibandingkan Air Pollution kemungkinan dikarenakan pengambilan data berada di wilayah yang dekat dengan kota. Data sensor akan berubah tergantung dari tempat pengambilan data.

6.3 Pengujian total *Delay*

Adapun pengujian total *delay* dilakukan dengan tahap-tahap berikut.

6.3.1 Tujuan Pengujian

Tujuan dari pengujian ini untuk mengetahui banyaknya data yang dapat diambil dalam 1 jam pengujian, apakah terdapat paket loss dan ataupun pembacaan sensor yang tidak *valid*. Pengujian ini juga dilakukan karena merupakan kebutuhan fungsional dan non fungsional dari sistem yang dibuat, agar diketahui apakah kebutuhan bisa terpenuhi apa tidak pada hasil analisis.

6.3.2 Prosedur Pengujian

Pengujian ini dilakukan dengan cara menyalakan alat dan membiarkannya menyala selama 1 jam dan melihat berapa banyak data yang masuk selama 1 jam tersebut ke thingspeak. Pengujian ini dilakukan sebanyak 5 kali.

6.3.3 Pelaksanaan Pengujian

Pengujian akan dilakukan dengan cara menyalakan alat dan membuka situs thingspeak, mengambil data hasil uji selama satu jam dan membandingkan interval waktu pada setiap data yang masuk. Adapun total waktu proses yang dilakukan Arduino yang akan digunakan pada pengujian ini dapat dilihat pada Tabel 6.3.

Tabel 6.3 Total *Delay* pada Program

Event Trigger	*Delay* (ms)
AT+RST	2000
AT+CWMODE	1000
AT+CWJAP	20000
AT+CISFR	1000
RTC	1000
RTC (Loop)	1000
DELAY DISPLAY (Loop)	1000
DELAY SENDING THINGSPEAK	16000
DELAY SENDING TWITTER	16000
DELAY STAND BY	21000
DELAY RESTART	4000
TOTAL DELAY	**84.000**

Dari Tabel 6.3 didapatkan total *delay* 84.000 atau 84 detik. Estimasikan looping yang terjadi paling lama 1 menit atau 60 detik, maka pengiriman data dan akuisisi data pada thingspeak membuat totalnya menjadi 144 detik atau kurang lebih dalam 2,5 menit 1 data didapatkan, maka akan ada 24 data yang didapatkan dalam 1 jam pengujian.

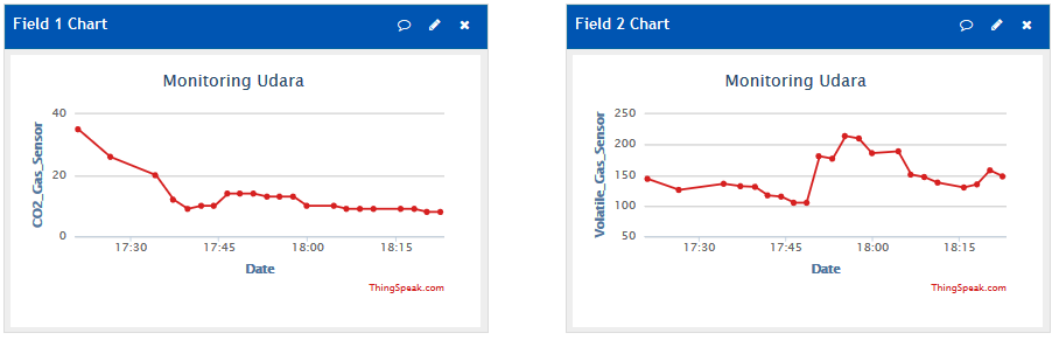
6.3.4 Hasil Pengujian dan Analisis

Adapun hasil pengujian yang sudah dilakukan sebagai berikut

Tabel 6.4 Percobaan 1

created_at	entry_id	field1	field2
2016-04-05 10:21:07 UTC	1	35	144
2016-04-05 10:26:33 UTC	2	26	126
2016-04-05 10:34:17 UTC	3	20	136
2016-04-05 10:37:17 UTC	4	12	132
2016-04-05 10:39:41 UTC	5	9	131
2016-04-05 10:41:56 UTC	6	10	117
2016-04-05 10:44:13 UTC	7	10	115
2016-04-05 10:46:29 UTC	8	14	105
2016-04-05 10:48:35 UTC	9	14	105
2016-04-05 10:50:50 UTC	10	14	181
2016-04-05 10:53:06 UTC	11	13	177
2016-04-05 10:55:21 UTC	12	13	214
2016-04-05 10:57:37 UTC	13	13	210
2016-04-05 10:59:53 UTC	14	10	186
2016-04-05 11:04:26 UTC	15	10	189
2016-04-05 11:06:41 UTC	16	9	151
2016-04-05 11:08:57 UTC	17	9	147
2016-04-05 11:11:13 UTC	18	9	138
2016-04-05 11:15:46 UTC	19	9	130
2016-04-05 11:18:02 UTC	20	9	135
2016-04-05 11:20:17 UTC	21	8	158
2016-04-05 11:22:33 UTC	22	8	148

Pada percobaan 1 didapatkan 22 data dengan rataan waktu 60/22=2.73 menit per data. Hasil penyimpanan data pada thingspeak dapat dilihat pada Gambar 6.7.



Gambar 6.7 Thingspeak Percobaan 1

Pada Gambar 6.7 terdapat 2 buah graph, graph di kiri adalah graph CO_2 dan graph yang di kanan adalah *Air Quality*. Adapun tampilan twitter pada Gambar 6.8.

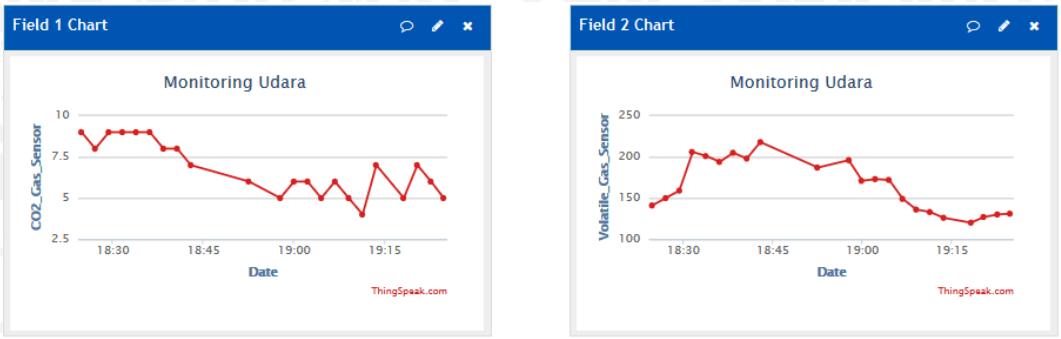
Gambar 6.8 Twitter Percobaan 1

Gambar 6.8 menampilkan tampilan data yang terdapat pada twitter sesuai dengan data pada Tabel 6.4.

Tabel 6.5 Percobaan 2

created_at	entry_id	field1	field2
2016-04-05 11:24:48 UTC	1	9	141
2016-04-05 11:27:04 UTC	2	8	150
2016-04-05 11:29:20 UTC	3	9	159
2016-04-05 11:31:35 UTC	4	9	206
2016-04-05 11:33:51 UTC	5	9	201
2016-04-05 11:36:07 UTC	6	9	194
2016-04-05 11:38:22 UTC	7	8	205
2016-04-05 11:40:38 UTC	8	8	198
2016-04-05 11:42:54 UTC	9	7	218
2016-04-05 11:52:33 UTC	10	6	187
2016-04-05 11:57:42 UTC	11	5	196
2016-04-05 11:59:58 UTC	12	6	171
2016-04-05 12:02:13 UTC	13	6	173
2016-04-05 12:04:29 UTC	14	5	172
2016-04-05 12:06:45 UTC	15	6	149
2016-04-05 12:09:02 UTC	16	5	136
2016-04-05 12:11:20 UTC	17	4	133
2016-04-05 12:13:36 UTC	18	7	126
2016-04-05 12:18:09 UTC	19	5	120
2016-04-05 12:20:24 UTC	20	7	127
2016-04-05 12:22:40 UTC	21	6	130
2016-04-05 12:24:46 UTC	22	5	131

Pada percobaan 2 didapatkan 22 data dengan rataan waktu 60/22=2.73 menit per data. Hasil penyimpanan data pada thingspeak dapat dilihat pada Gambar 6.9.



Gambar 6.9 Thingspeak Percobaan 2

Pada Gambar 6.9 terdapat 2 buah graph, graph di kiri adalah graph CO_2 dan graph yang di kanan adalah *Air Quality*. Adapun tampilan twitter pada Gambar 6.10.

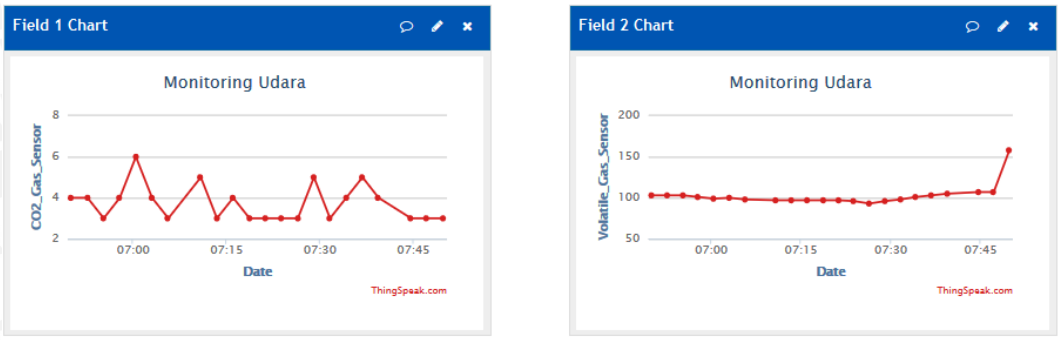
Gambar 6.10 Twitter Percobaan 2

Gambar 6.10 menampilkan tampilan data yang terdapat pada twitter sesuai dengan data pada Tabel 6.5.

Tabel 6.6 Percobaan 3

created_at	entry_id	field1	field2
2016-05-10 23:50:03 UTC	1	4	103
2016-05-10 23:52:40 UTC	2	4	103
2016-05-10 23:55:16 UTC	3	3	103
2016-05-10 23:57:51 UTC	4	4	101
2016-05-11 00:00:26 UTC	5	6	99
2016-05-11 00:03:02 UTC	6	4	100
2016-05-11 00:05:39 UTC	7	3	98
2016-05-11 00:10:52 UTC	8	5	97
2016-05-11 00:13:27 UTC	9	3	97
2016-05-11 00:16:04 UTC	10	4	97
2016-05-11 00:18:41 UTC	11	3	97
2016-05-11 00:21:16 UTC	12	3	97
2016-05-11 00:23:50 UTC	13	3	96
2016-05-11 00:26:27 UTC	14	3	93
2016-05-11 00:29:02 UTC	15	5	96
2016-05-11 00:31:37 UTC	16	3	98
2016-05-11 00:34:13 UTC	17	4	101
2016-05-11 00:36:50 UTC	18	5	103
2016-05-11 00:39:25 UTC	19	4	105
2016-05-11 00:44:38 UTC	20	3	107
2016-05-11 00:47:12 UTC	21	3	107
2016-05-11 00:49:49 UTC	22	3	158

Pada percobaan 3 didapatkan 22 data dengan rataan waktu 60/22=2.73 menit per data. Hasil penyimpanan data pada thingspeak dapat dilihat pada Gambar 6.11.



Gambar 6.11 Thingspeak Percobaan 3

Pada Gambar 6.11 terdapat 2 buah graph, graph di kiri adalah graph CO_2 dan graph yang di kanan adalah *Air Quality*. Adapun tampilan twitter pada Gambar 6.12.

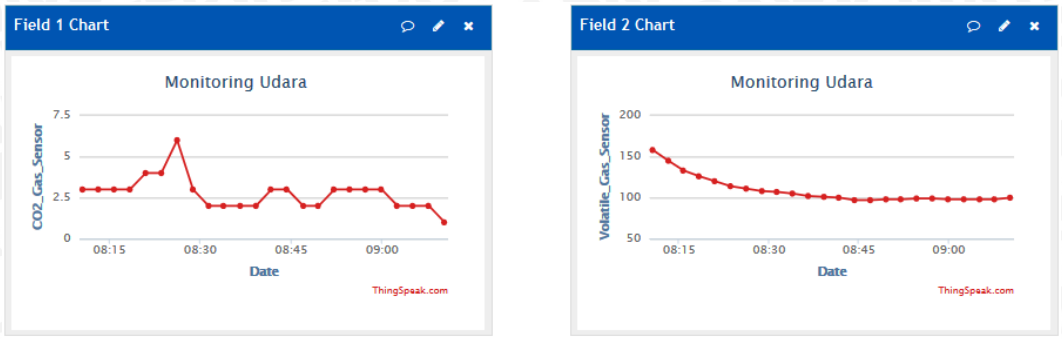
Gambar 6.12 Twitter Percobaan 3

Gambar 6.12 menampilkan tampilan data yang terdapat pada twitter sesuai dengan data pada Tabel 6.6.

Tabel 6.7 Percobaan 4

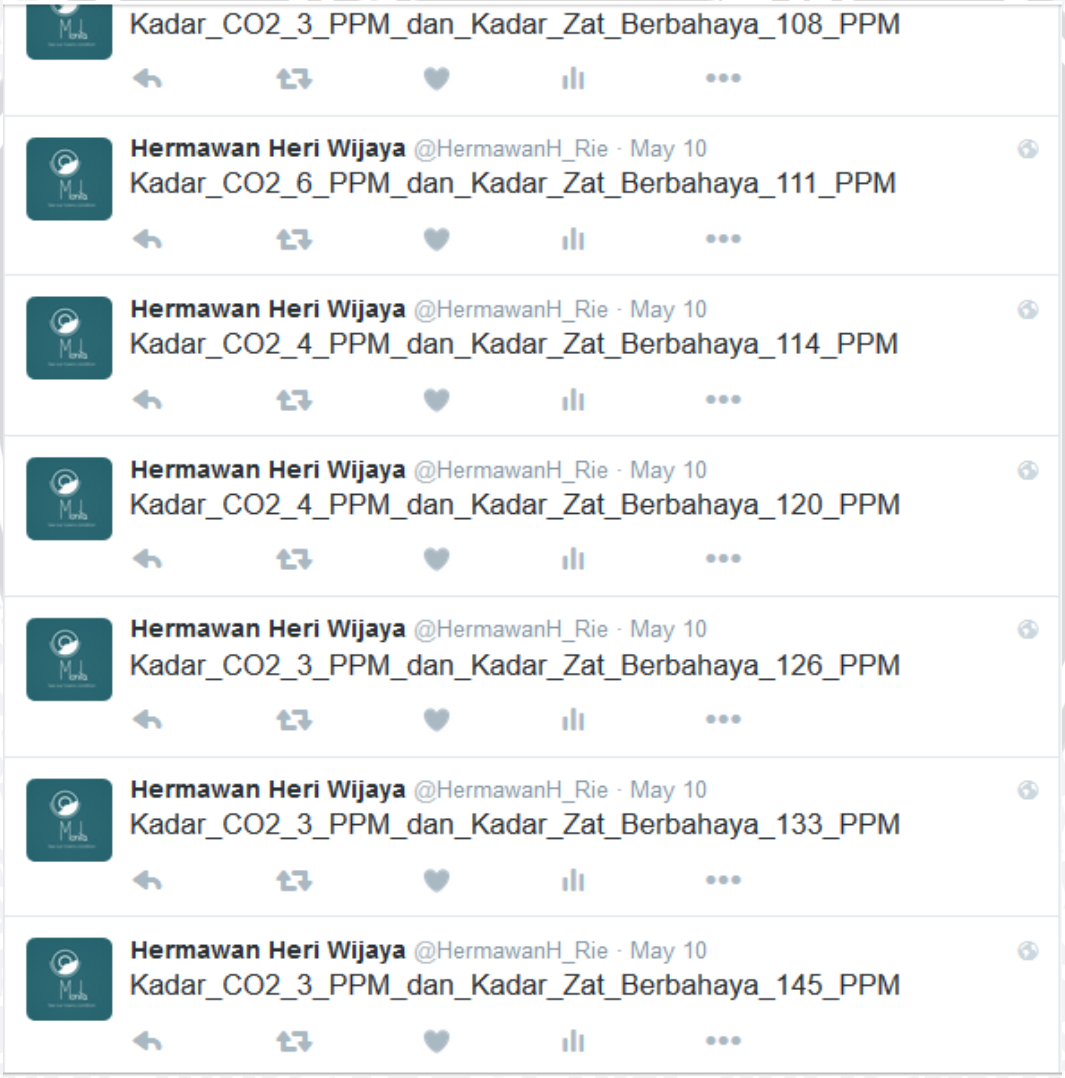
created_at	entry_id	field1	field2
2016-05-11 01:10:37 UTC	1	3	158
2016-05-11 01:13:12 UTC	2	3	145
2016-05-11 01:15:46 UTC	3	3	133
2016-05-11 01:18:23 UTC	4	3	126
2016-05-11 01:21:00 UTC	5	4	120
2016-05-11 01:23:34 UTC	6	4	114
2016-05-11 01:26:11 UTC	7	6	111
2016-05-11 01:28:48 UTC	8	3	108
2016-05-11 01:31:22 UTC	9	2	107
2016-05-11 01:33:57 UTC	10	2	105
2016-05-11 01:36:34 UTC	11	2	102
2016-05-11 01:39:11 UTC	12	2	101
2016-05-11 01:41:45 UTC	13	3	100
2016-05-11 01:44:22 UTC	14	3	97
2016-05-11 01:46:59 UTC	15	2	97
2016-05-11 01:49:33 UTC	16	2	98
2016-05-11 01:52:08 UTC	17	3	98
2016-05-11 01:54:45 UTC	18	3	99
2016-05-11 01:57:22 UTC	19	3	99
2016-05-11 01:59:56 UTC	20	3	98
2016-05-11 02:02:33 UTC	21	2	98
2016-05-11 02:05:10 UTC	22	2	98
2016-05-11 02:07:44 UTC	23	2	98
2016-05-11 02:10:19 UTC	24	1	100

Pada percobaan 4 didapatkan 24 data dengan rataan waktu 60/24=2.5 menit per data. Hasil penyimpanan data pada thingspeak dapat dilihat pada Gambar 6.13.



Gambar 6.13 Thingspeak Percobaan 4

Pada Gambar 6.13 terdapat 2 buah graph, graph di kiri adalah graph CO_2 dan graph yang di kanan adalah *Air Quality*. Adapun tampilan twitter pada Gambar 6.14.



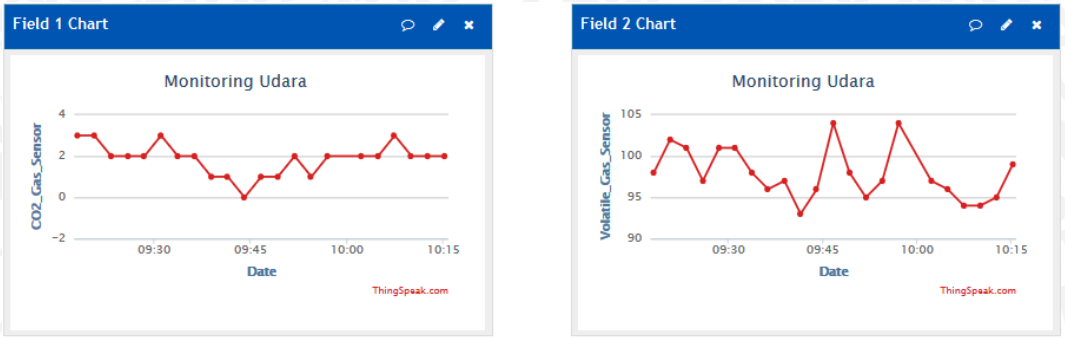
Gambar 6.14 Twitter Percobaan 4

Gambar 6.14 menampilkan tampilan data yang terdapat pada twitter sesuai dengan data pada Tabel 6.7.

Tabel 6.8 Percobaan 5

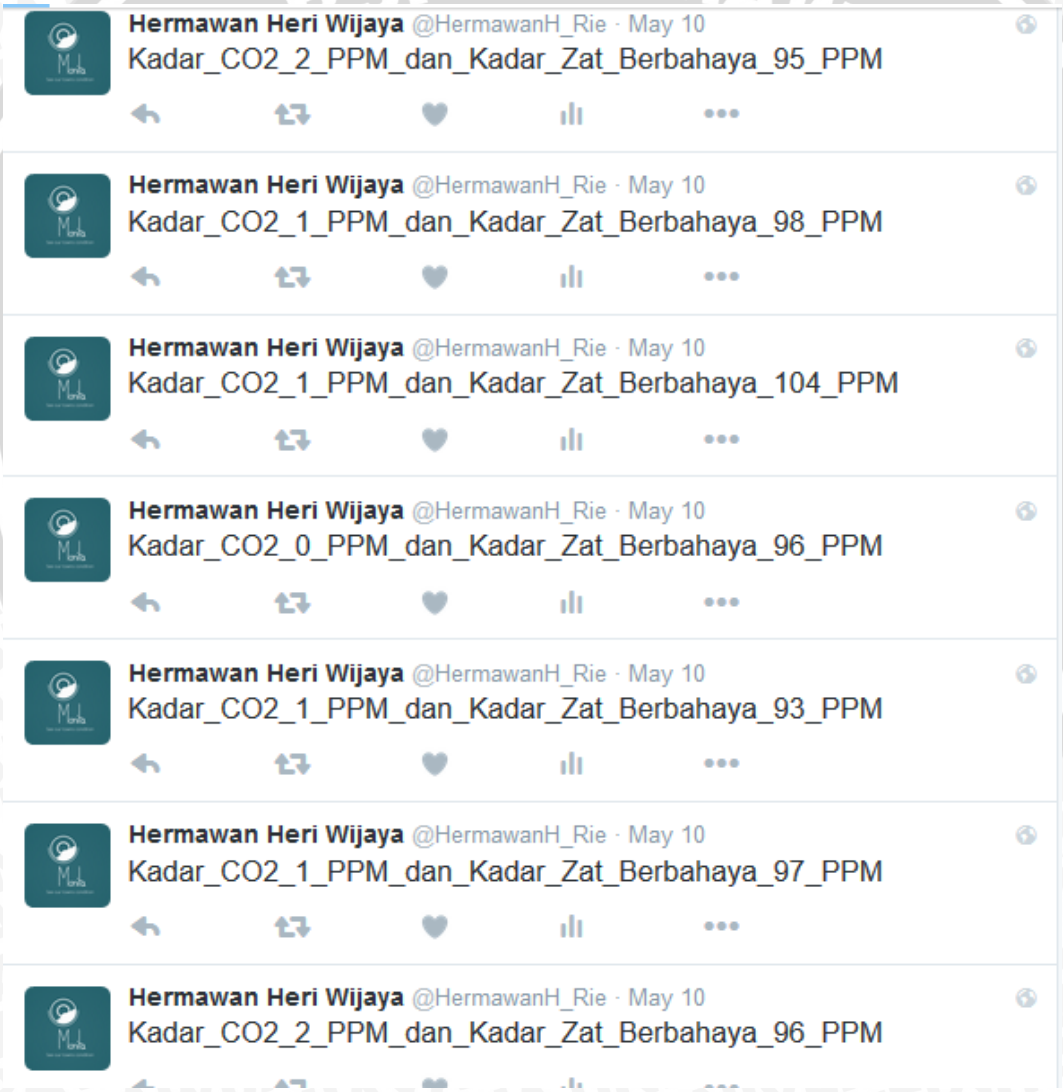
created_at	entry_id	field1	field2
2016-05-11 02:18:07 UTC	1	3	98
2016-05-11 02:20:42 UTC	2	3	102
2016-05-11 02:23:19 UTC	3	2	101
2016-05-11 02:25:55 UTC	4	2	97
2016-05-11 02:28:30 UTC	5	2	101
2016-05-11 02:31:05 UTC	6	3	101
2016-05-11 02:33:42 UTC	7	2	98
2016-05-11 02:36:18 UTC	8	2	96
2016-05-11 02:38:54 UTC	9	1	97
2016-05-11 02:41:28 UTC	10	1	93
2016-05-11 02:44:04 UTC	11	0	96
2016-05-11 02:46:40 UTC	12	1	104
2016-05-11 02:49:15 UTC	13	1	98
2016-05-11 02:51:52 UTC	14	2	95
2016-05-11 02:54:28 UTC	15	1	97
2016-05-11 02:57:03 UTC	16	2	104
2016-05-11 03:02:16 UTC	17	2	97
2016-05-11 03:04:51 UTC	18	2	96
2016-05-11 03:07:26 UTC	19	3	94
2016-05-11 03:10:03 UTC	20	2	94
2016-05-11 03:12:39 UTC	21	2	95
2016-05-11 03:15:14 UTC	22	2	99

Pada percobaan 5 didapatkan 22 data dengan rataan waktu 60/22=2.73 menit per data. Hasil penyimpanan data pada thingspeak dapat dilihat pada Gambar 6.15.



Gambar 6.15 Thingspeak Percobaan 5

Pada Gambar 6.15 terdapat 2 buah graph, graph di kiri adalah graph CO_2 dan graph yang di kanan adalah *Air Quality*. Adapun tampilan twitter pada Gambar 6.14.

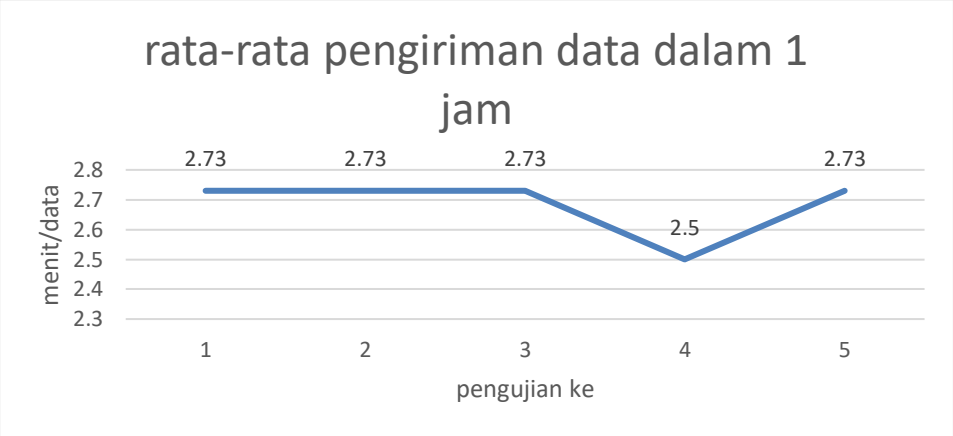


Gambar 6.16 Twitter Percobaan 5

Gambar 6.16 menampilkan tampilan data yang terdapat pada twitter sesuai dengan data pada Tabel 6.8.

6.4 Analisis

Dari hasil 5 kali pengujian didapatkan rata-rata pengiriman data dalam 1 jam pengujian yang dapat ditunjukkan dalam bentuk grafik yang dapat dilihat pada Gambar 6.5



Gambar 6.17 Grafik Rata-Rata Pengiriman Data Dalam 1 Jam Pengujian

Dari Gambar 6.17 dapat dilihat bahwa rata-rata pengiriman adalah 2.73 menit dan terjadi peningkatan pada pengujian ke 4 dengan rata-rata pengiriman 2.5 menit, estimasi pengiriman yang berupa 2,5 menit per data terbukti benar, namun dikarenakan adanya kemungkinan data tidak sampai atau tidak terkirim karena tidak adanya IP address saat terjadi request DHCP atau IP telah habis, atau adanya timelapse yang terjadi pada Arduino dikarenakan sebuah *script*

```
1  if ((ttemp==30)||(ttemp==31)){
2    TestThingspeak();
3    TwitterTweet();
4    Sleepy::loseSomeTime(21000);
5    Serial.println("Autorestart Program!!");
6    delay (4000);
7    asm volatile ("  jmp 0");     }
```

Gambar 6.18 Script untuk mengirim data berdasarkan waktu

Pada Gambar 6.18 terdapat fungsi, jika detik memasuki 30 atau 31 pada waktu yang ditujukkan oleh RTC module, maka lakukan pengiriman ke thingspeak, twitter, stand by 21 detik, serial print, dan *delay* 4 detik, kemudian restart Arduino.

Pada fungsi detik memasuki 30 atau 31 inilah yang berkemungkinan besar menyebabkan perbedaan jumlah banyaknya data yang terekam pada thingspeak, dikarenakan timing saat menyalakan alat dan penyelesaian proses yang berjalan pada Arduino, karena waktu menyalakan alat tidak selalu pasti dan presisi.

BAB 7 PENUTUP

Berdasarkan metodologi penelitian yang sudah disusun sebelumnya, bab ini merupakan tahap akhir dalam melakukan penelitian setelah melakukan pengujian dan analisa hasil pengujian dari sistem yang sudah dirancang. Berikut merupakan kesimpulan dan saran yang dapat ditarik dari penelitian yang telah dilakukan

7.1 Kesimpulan

Berdasarkan hasil pengujian konektivitas ESP8266, pengujian Fungsionlitas Sensor dan Pengujian *Total Delay* didapatkan beberapa kesimpulan sebagai berikut:

1. ESP8266 dapat digunakan dengan AT *Command* pada mikrokontroler Arduino dan dapat digunakan sebagai *station*, *Access Point* ataupun keduanya, tergantung dari penggunaan ESP8266 yang diinginkan.
2. Sensor MG-811 dan MQ-135 memerlukan daya yang lebih untuk melakuan pembacaan sensor, sehingga diperlukan voltase yang lebih besar dari 6 volt namun tidak membuat mikrokontroler overheat karena tegangan yang terlalu besar.
3. Estimasi perhitungan data yang dikirimkan berdasarkan pada Tabel 6.3 yaitu sebanyak 24 data yang masuk dan data yang sampai tidak sepenuhnya sesuai dengan estimasi yang diperhitungkan, berdasarkan hasil yang ditunjukkan oleh Gambar 6.17 dengan rata-rata data yang masuk sebanyak 2,73 menit. Karena banyaknya kemungkinan kesalahan yang dapat terjadi pada saat data mulai dikirim, baik dari IP address, besarnya bandwidth, banyaknya request yang ditangani oleh *Access Point* atau server dan masih banyak lagi.
4. Sistem yang dirancang sudah dapat dikatakan bisa dipergunakan dengan baik karena data sudah dapat sampai ke thingspeak dan terposting ke twitter, namun hanya estimasi banyak data yang harusnya terkirim ternyata tidak sesuai. Namun secara keseluruhan sistem yang sudah dirancang sesuai dengan apa yang diharapkan oleh penulis.

7.2 Saran

Ada beberapa saran untuk para peneliti yang ingin melakukan pengembangan pada penelitian ini, sarannya sebagai berikut

1. Berfokus untuk mengakurasikan data yang didapatkan oleh sensor, mempelajari sensor lebih jauh dan mencari data tentang karbonmonoksida dan zat berbahaya yang terdapat di udara sebesar berapa PPM dan mengaplikasikannya dalam area *outdoor*.
2. Meneliti lebih dalam lagi agar estimasi data terkirim dan data yang diterima agar tidak terpaut terlalu jauh dan lebih baik.

3. ESP8266 tidak dapat bekerja dengan baik apabila traffic upsteam dan downstream pada internet yang digunakan terlalu banyak, selain itu sensor MG-811 dan MQ-135 mudah overheat apabila digunakan di area *outdoor*, akan lebih baik apabila mencari referensi sensor lainnya atau melakukan rancang bangun sensor sendiri.
4. Menambahkan fitur untuk konfigurasi melalui panel LCD dan dengan input berupa keypad, sehingga tidak perlu melakukan pemrograman ulang apabila SSID diganti atau berpindah SSID, hanya cukup menginput SSID dan *Password*.

DAFTAR PUSTAKA

Aditya, Bryan, Gerard & Sally, 2015. *Smart City Architecture and its Applications based on* IoT*,* Coleraine, Co., Londonderry BT52 1SA. UK: A School of Computing and Information Engineering. University of Ulster.

Ali, Z. & Shahzad, W., 2012. *Intelligence, Analysis of Routing Protocols in AD HOC and Sensor Wireless Networks Based on Swarm.* [Online] Available at: http://article.sapub.org/10.5923.j.ijnc.20130301.01.html [Accessed 7 Juni 2016].

Andres, M., Giner, Alejandro & Gandhi, 2012. *Developing Social Networks Mashups: An Overview of RESTBased APIs,* Mexico: Division of Research and Postgraduates Studies.

Ardi, D. & Nur, H., 2015. *prototipe sistem informasi ketinggian air melalui media sosial twitter sebagai sistem peringatan dini bahaya banjir,* Universitas Gajah Mada, Jogja: Diploma III Sekolah Vokasi.

Arduino, 2016. *Arduino Mega 2560.* [Online] Available at: https://www.arduino.cc/en/Main/ArduinoBoardMega2560 [Accessed 7 Juni 2016].

Arun, K. & Selvarathy, T., 2014. *A Review on Building Social Networking APIs,* Tamilnadu, India: Department of Computer Science and Engineering. Einstein College of Engineering, Anna university, Tirunelveli.

Digiware, 2015. *DT-Sense Air Quality Sensor.* [Online] Available at: http://digiwarestore.com/en/gas/dt-sense-air-quality-sensor-991918.html [Accessed 7 Juni 2016].

Digiware, 2015. *DT-Sense Carbon Dioxide Sensor.* [Online] Available at: http://digiwarestore.com/en/gas/dt-sense-carbon-dioxide-sensor-991920.html [Accessed 7 Juni 2016].

Figaro, 2013. *TGS 2602 - for the detection of Air Contaminants.* [Online] Available at: http://www.imagesco.com/catalog/sensors/files/TGS2602_datasheet.pdf [Accessed 2 October 2015].

Heri, H. et al., 2014. *Sistem monitoring udara MONITA Online,* Universitas Brawijaya, Malang: PKM, Program Teknologi Informasi dan Ilmu Komputer.

Iskandar & M.Si, D. M., 2011. *nilai ambang batas faktor fisika dan faktor kimia di tempat kerja.* [Online] Available at: http://betterwork.org/in-labourguide/wp-content/uploads/PERMENA.pdf [Accessed 2 Oktober 2015].

Kemp & Simon, 2015. *Global statshot august version.* [Online] Available at: http://wearesocial.sg/blog/2015/08/global-statshot-august-2015/ [Accessed 2 Oktober 2015].

Kompas, 2013. *Dampak Polusi Udara Bagi Penduduk Jakarta.* [Online] Available at: http://green.kompasiana.com/polusi/2013/12/04/dampak-polusi-udara-bagi-penduduk-jakarta-616543.html [Accessed 18 September 2014].

Modul9, 2013. *pengukuran qos streaming server.* [Online] Available at: http://zenhadi.lecturer.pens.ac.id/kuliah/Jarkom2/Prakt9%20Pengukuran%20QoS%20Streaming%20Server.pdf [Accessed 2 Oktober 2015].

Nico, Sukiswo & Enda, 2015. *Perancangan Aplikasi Pengaturan Wireless Sensor Network Berbasis Web,* Semarang: Jurusan Teknik Elektro, Universitas Diponegoro.

NURDs, 2015. *ESP8266.* [Online] Available at: https://nurdspace.nl/ESP8266 [Accessed 6 Juni 2016].

O'Hare, Marsh, Ruzzelli & Tynan, 2015. *Agents for Wireless Sensor Network Power Management,* Belfield, Dublin 4. Ireland: Department of Computer Science. University College Dublin.

Parallax, 2010. CO_2 *Gas Sensor Module Datasheet.* [Online] Available at: http://datasheet.octopart.com/27929-Parallax-datasheet-13538405.pdf [Accessed 2 Oktober 2015].

Reddy & Jeevananda, D. S., 2008. *Climate Change: Myths and Realities.* 4th ed. Hyderabad: Dr. S. Jeevananda Reddy.

Ridho & Ali, 2010. *rancang bangun sistem pengukuran polutan gas h2s pada lokasi manifestasi geothermal gedung songo menggunakan sensor tgs 2602,* universitas diponegoro, semarang: program studi diii instrumentasi dan elektronika jurusan fisika. fakultas matematika dan ilmu pengetahuan alam.

system & Epressive, 2013. *ESP8266 802.11bgn Smart Device Datasheet.* [Online] Available at: http://wiki.iteadstudio.com/images/e/e0/ESP8266_Specifications_English.pdf [Accessed 2 October 2015].

ThingSpeak, 2016. *Thingspeak.* [Online] Available at: http://www.thingspeak.com [Accessed 6 Juni 2016].

Twitter, 2015. *REST APIs.* [Online] Available at: https://dev.twitter.com/rest/public [Accessed 2 October 2015].

waveshare, 2015. *MQ-135 Gas Sensor.* [Online]-+_= Available at: http://www.waveshare.com/mq-135-gas-sensor.htm [Accessed 7 Juni 2016].

Wikipedia, 2015. *Air pollution.* [Online] Available at: https://en.wikipedia.org/wiki/Air_pollution [Accessed 7 Juni 2016].

Yusad & Yusniwarti., 2003. *Polusi Udara Dikota-kota Besar Dunia,* Medan: Program Studi Kesehatan Masyarakat USU.