TUGAS AKHIR – EE 184801

SISTEM PEMANTAUAN KADAR GAS PADA TAMBANG BATUBARA BERBASIS IOT MENGGUNAKAN TEKNOLOGI KOMUNIKASI LORA

Fauzi Muhammad Ikhsan
NRP 07111540000102

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EE 184801

SISTEM PEMANTAUAN KADAR GAS PADA TAMBANG BATUBARA BERBASIS IOT MENGGUNAKAN TEKNOLOGI KOMUNIKASI LORA

Fauzi Muhammad Ikhsan
NRP 07111540000102

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – EE 184801

GAS MONITORING SYSTEM IN COAL MINE BASED ON IOT USING LORA COMMUNICATION TECHNOLOGY

Fauzi Muhammad Ikhsan
NRP 07111540000102

Supervisor(s)
Dr. Muhammad Rivai, ST., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Sistem Pemantauan Kadar Gas Pada Tambang Batubara Berbasis IoT Menggunakan Teknologi Komunikasi LoRa**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 13 Desember 2019

Fauzi Muhammad Ikhsan
NRP. 0711 15 4000 0102

---Halaman ini sengaja dikosongkan---

SISTEM PEMANTAUAN KADAR GAS PADA TAMBANG BATUBARA BERBASIS IOT MENGGUNAKAN TEKNOLOGI KOMUNIKASI LORA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I

Dr. Muhammad Rivai, ST., MT.
NIP. 196904261994031003

SURABAYA
DESEMBER, 2019

---Halaman ini sengaja dikosongkan---

SISTEM PEMANTAUAN KADAR GAS PADA TAMBANG BARUBARA BERBASIS IOT MENGGUNAKAN TEKNOLOGI KOMUNIKASI LORA

Nama : Fauzi Muhammad Ikhsan
Pembimbing : Dr. Muhammad Rivai, ST., MT.

ABSTRAK

Sistem Pemantauan Kadar Gas Pada Tambang Batubara Berbasis IoT Menggunakan Teknologi Komunikasi LoRa adalah sebuah sistem *portable* yang berfungsi untuk memantau kadar gas yang berbahaya di tambang menggunakan dua sensor serta modul GPS untuk mengetahui posisi dari alat ini, serta menggunakan LoRa sebagai modul komunikasi. Pembuatan alat ini didasarkan oleh adanya gas yang berbahaya untuk manusia di area pertambangan, dari segi kecelakaan kerja seperti kebakaran atau ledakan yang dikarenakan gas CH_4 dan juga sistem pernapasan yang akan terganggu jika terlalu banyak terpapar gas CO_2. Oleh karena itu sistem ini dapat memantau perubahan konsentrasi gas. Sensor yang digunakan adalah sensor gas jenis MQ-02 dan MQ-135. Tugas akhir ini mengimplementasikan sistem IoT untuk pemantauan kadar gas di area tambang batubara. Dalam pembuatan alatnya, *microcontroller* yang digunakan adalah Arduino, dengan dua sensor gas untuk mendeteksi konsentrasi dari gas yang di observasi, GPS untuk menentukan posisi dari alat ini, dan LoRa sebagai alat komunikasi. Alat ini nantinya dapat dibawa oleh pengguna asalkan masih dalam jarak yang stabil untuk komunikasi antar LoRa. Ketika LoRa di *transmitter* mengirimkan data, LoRa di *receiver* akan menerima data tersebut dan mengolahnya, lalu dikirim melalui komunikasi serial ke NodeMCU, dan diteruskan ke aplikasi yang ada di *smartphone*. Pengujian sensor gas di area pertambangan mendapatkan hasil yang berbeda dengan area perkotaan. Jarak komunikasi maksimal yang bisa dilakukan di area perkotaan kurang lebih 600 meter, sementara di area pertambangan dapat mencapai 1,6 Km. Konsentrasi gas yang terukur saat pengujian relatif tidak stabil dikarenakan faktor alam seperti adanya angin.

Kata Kunci: *IoT, LoRa, Konsentrasi Gas, Arduino, Tambang Batubara.*

---Halaman ini sengaja dikosongkan---

GAS MONITORING SYSTEM IN COAL MINE BASED ON IOT USING LORA COMMUNICATION TECHNOLOGY

Name : Fauzi Muhammad Ikhsan
Supervisor : Dr. Muhammad Rivai, ST., MT.

ABSTRACT

Gas Monitoring System in Coal Mine Based on IoT Using LoRa Communication Technology is a portable system to monitor gas concentration that can harm us, using two sensors and a GPS module to detect the position of this system, this system use LoRa as a communication module. This system is invented due to the existence of harmful gas for human in coal mine, from the sector of work accident like wildfire or explosion due to CH_4 gases and respiratory system disorders due to CO_2 gases. Therefore, this system has a capability to monitor the gas concentration change time-by-time, using MQ-02 and MQ-135 gas sensor. In this final project, IoT system is implemented to monitor the gas concentration. This system using Arduino microcontroller, with two gas sensors to observe the gas concentration, GPS to determine the position of transmitter, and LoRa as a communication module. Later, this system could be carried by the user as-long-as they stay in the range between LoRa communication. When LoRa transmitter is sending a data, the other LoRa is receiving the data and process it, then send it to NodeMCU by serial communication, and forward the data to smartphone application. Gas sensor has been calibrated by testing it with various object, such as smoke that was made by burning a paper and cigarette smoke. When the transmitter was tested in coal mine area, results obtained was different when tested in urban area. Results obtained during the test was vary. The maximum range that LoRa could communicate each other in urban area is 600 meters more or less, while in coal mine area LoRa could deliver data up to 1,6 Km. Gas concentration that has been metered during the test relatively goes up and down due to natural factor such as wind blow.

Keywords: IoT, LoRa, Gas Concentration, Coal Mine

---Halaman ini sengaja dikosongkan---

KATA PENGANTAR

Segala puji syukur kepada Allah SWT yang telah memberikan nikmat dan karunia-Nya kepada penulis sehingga penulis dapat menyelesaikan laporan tugas akhir dengan judul "**Sistem Pemantauan Kadar Gas Pada Tambang Batubara Berbasis IoT Menggunakan Teknologi Komunikasi LoRa**", sebagai salah satu persyaratan dalam menyelesaikan pendidikan program studi S1 di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan banyak terima kasih kepada semua pihak yang telah membantu dan memberikan dukungan dalam penulisan dan penyusunan laporan tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih yang tulus dan sebesar-besarnya kepada:

1. Dr. Muhammad Rivai, ST., MT. selaku dosen pembimbing yang telah membimbing dan memberikan saran selama pengerjaan dan penulisan laporan tugas akhir.
2. Dr. Ir. Hendra Kusuma, M.Eng.Sc., Ir. Tasripan, MT., Dr. Ir. Totok Mujiono, M.IKom., Muhammad Attamimi, B.Eng., M.Eng., Ph.D. sebagai dosen penguji.
3. Kepala Departemen Teknik Elektro ITS, Dr. Eng. Ardyono Priyadi, ST., M.Eng. atas izin dan kesempatan yang diberikan kepada penulis untuk melaksanakan tugas akhir ini.
4. Orang tua yang sudah saya repotkan namun selalu memberikan dukungan dan doa kepada penulis.
5. Rekan-rekan Kalpataru, Ayam Jantan, e55, Ngatoibae, Thaygeeks dan Moniyca sudah menjadi keluarga saya yang selalu memberikan dukungan dan bantuan kepada saya.

Penulis berharap agar laporan tugas akhir ini dapat memberikan manfaat sebesar-besarnya kepada siapapun yang membacanya. Penulis juga menyadari masih banyak kekurangan dalam penulisan laporan tugas akhir ini. Oleh karena itu penulis menerima setiap kritik dan saran yang diberikan. Akhir kata, penulis mengucapkan terima kasih yang sebesar-besarnya.

Surabaya, 13 Desember 2019

Penulis

---Halaman ini sengaja dikosongkan---

DAFTAR ISI

---Halaman ini sengaja dikosongkan---

DAFTAR GAMBAR

DAFTAR TABEL

---Halaman ini sengaja dikosongkan---

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pertambangan adalah cara untuk mengambil mineral dan sumber energi yang tersimpan di bawah tanah yang kita pijak ini. Seperti batu bara, minyak, emas, kapur, dan sejenisnya. Saat ini, industri pertambangan mineral dan sumber energi sudah dilakukan hampir di seluruh dunia. Di masa lalu, ada orang yang dikirim untuk mengecek kondisi tambang di beberapa bagian. Mereka memiliki tugas untuk melaporkan kondisi tambang dengan waktu nyata. Metode ini sangat membuang- buang waktu dan tidak efektif [1].

Untuk tujuan memperbaiki sistem komunikasi yang ada di daerah tambang, dan untuk mencapai keinginan dalam mobilitas tinggi serta keefektifan waktu, dibutuhkan sebuah sistem yang mendukung. Sistem pemantau yang berbasis IoT dapat membantu untuk mengirim data kondisi tambang secara waktu nyata [1].

Kondisi tambang batubara yang diobservasi antara lain adalah kadar karbondioksida dan kadar gas metana. Di bagian sensor terdiri atas sensor gas CO_2 (MQ-135), dan sensor gas CH_4 (MQ-2). Sensor- sensor tersebut telah disesuaikan dengan kondisi kadar gas yang ada di tambang [2].

Sekarang, sistem yang diperlukan adalah sistem wireless sensor network, dimana sistem ini mempunyai volume kecil dan ringan. Dengan keuntungan tersebut, lokasi dari wireless node dapat berpindah-pindah dan dapat dilepas mau pun dipasang seperlunya. Wireless node ini mengungguli sistem komunikasi yang masih menggunakan kabel dan bisa dipasang di daerah yang tidak ada manusia. Dalam suatu kasus ketika jalur komunikasi terputus karena suatu masalah, wireless node tersebut tetap dapat melakukan komunikasi dan menyampaikan informasi [3].

Sistem ini sendiri di bagian processing memakai sebuah mikrokontroller. Untuk sistem komunikasi antara node dengan receiver, memakai modul LoRa. Modul LoRa dikondisikan untuk menerima informasi dari node dan mengirim informasi tersebut ke LoRa lainnya untuk dikirim ke receiver yang selanjutnya akan dipublish ke sebuah aplikasi dengan menggunakan bantuan NodeMCU. Hasil yang diharapkan adalah alat ini dapat membantu mengurangi tingkat kecelakaan atau kematian di daerah pertambangan.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan masalah sebagai berikut.

1. Bagaimana cara mengetahui keadaan gas di area pertambangan?
2. Bagaimana cara mengirim data yang terukur dari area pertambangan supaya pengguna dapat melakukan pemantauan dengan jarak yang jauh?

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut.

1. Memanfaatkan sensor gas untuk mengobservasi konsentrasi gas yang sudah ditentukan.
2. Mengirim data konsentrasi gas dari *transmitter* ke *receiver* menggunakan LoRa, lalu diunggah ke aplikasi untuk pemantauan.

1.4 Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah sebagai berikut.

1. Modul LoRa hanya bisa berkomunikasi dalam jarak maksimal 10 km.
2. Parameter yang digunakan adalah konsentrasi gas yang telah ditentukan.
3. Analisis yang dilakukan tidak mempertimbangkan aspek eksternal yang mungkin ditimbulkan.
4. Pembahasan yang dilakukan dengan melibatkan 1 *node* sensor saja.

1.5 Metodologi Penelitian

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. Studi literatur kandungan gas di area pertambangan

   Studi literatur ini akan berisi pengumpulan serta pengkajian teori, data dan penelitian yang dianggap relevan dan terpercaya untuk mendukung keabsahan tugas akhir ini, terutama literatur tentang

kandungan gas yang ada pada area pertambangan. Literatur yang digunakan akan memiliki batasan-batasan tertentu. Yaitu, literatur yang digunakan harus bersumber dari *paper*, jurnal, buku, maupun artikel yang berasal dari badan pemerintahan atau institusi akademik terpercaya.

2. Perancangan keseluruhan sistem monitoring

Perancangan keseluruhan sistem ini bertujuan untuk membuat serangkaian sistem, dimana sistem tersebut digunakan untuk mengirimkan informasi yang sudah dikumpulkan dan juga menerima informasi kemudian mengolahnya. Hasil yang diharapkan yaitu sistem tersebut dapat dikemas dengan baik sehingga memiliki mobilitas yang tinggi.

3. Perancangan dan Pengujian sensor gas dengan tabel

Perancangan sensor bisa dengan menggunakan pcb yang sudah dibuat dan didalamnya tedapat modul Arduino, dimana sensor akan disesuaikan tempatnya agar mendapatkan data yang tepat. Pengujian dilakukan dengan gas buatan yang dibuat sedemikian rupa sehingga mirip dengan gas yang ada di area pertambangan.

4. Perancangan sistem komunikasi RF dan GPS

Perancangan sistem komunikasi RF dilakukan dengan cara menggunakan modul LoRa, modul GPS, pcb yang sama dengan pcb yang digunakan untuk sensor yang sudah dibuat beserta dengan Arduino. Dimana sistem ini nantinya akan ditempatkan dengan tepat agar dapat mengirim data yang sesuai dengan yang diinginkan.

5. Perancangan sistem di *smartphone*

Perancangan sistem yang ada di *smartphone*, yaitu merancang sebuah aplikasi berbasis *smartphone* dari data yang telah diterima dan diolah oleh modul Arduino. Perancangan sistem ini membutuhkan suatu server internet pada suatu *mikrokontoroller* untuk menyimpan dan memperbarui data kadar gas. *Software* antarmuka untuk pemantauan kadar gas pada tambang batubara ini akan dirancang dengan memanfaatkan komponen-komponen yang tersedia pada aplikasi Blynk.

6. Pengujian secara keseluruhan

Pengujian secara keseluruhan dilakukan ketika semua sistem telah dibuat. Pengujian dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan apa yang diinginkan atau belum. Pengujian ini meliputi kinerja sensor, *transmitter*, *receiver*, *mikrokontroller*, hingga interface yang ada pada aplikasi Blynk.

7. Penyusunan Laporan Tugas Akhir

Tahap penyusunan laporan merupakan tahap terakhir dari proses pengerjaan tugas akhir ini. Laporan berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, studi literatur, tinjauan pustaka, perancangan dan pembuatan sistem, pengujian dan analisa, serta penutup.

1.6 Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

- BAB I: Pendahuluan
  Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.
- BAB II: Tinjauan Pustaka
  Bab ini berisi mengenai teori yang mendasari penyusunan laporan tugas akhir secara umum khususnya teori yang berhubungan dengan komponen yang akan digunakan.
- BAB III: Perancangan Sistem
  Bab ini menjelaskan tentang perencanaan sistem yang meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*) untuk pembuatan alat pada tugas akhir ini.
- BAB IV: Pengujian Dan Analisis
  Bab ini berisi tentang pengujian alat pada sistem yang telah dibuat dan analisa hasil dari pengujian yang telah didapat.
- BAB V: Penutup
  Bab ini berisi tentang kesimpulan yang diperoleh dari alat yang telah dibuat serta saran untuk pengembangan lebih lanjut.

1.7 Relevansi

Sistem pemantauan tambang batubara menggunakan komunikasi Radio Frequency (RF) merupakan sistem pemantauan yang memanfaatkan teknologi RF sebagai media komunikasi dan menggabungkannya dengan server dan user interface sehingga dapat memudahkan pemantauan kadar gas. Merupakan pengembangan dari teknologi RF yang digunakan dalam berkomunikasi, dengan pengimplementasian di area tambang batubara.

BAB II
TINJAUAN PUSTAKA

2.1 Kandungan Gas Pada Tambang Batubara

Di area pertambangan, persentase dari oksigen akan berkurang dalam suatu proses yang dikenal sebagai pengenceran oleh mesin pembakaran, api dan pembakaran kayu suhu rendah, yang dalam proses tersebut menambahkan gas-gas lain seperti karbonmonoksida dan karbondioksida.

Salah satu gas yang berbahaya lainnya dalam pertambangan adalah metana. Gas metana biasanya ada karena terlepas dari fraktur material saat penambangan berlangsung. Dengan kondisi ketika konsentrasi gas metana dapat menciptakan ledakan ketika berada diantara 5% sampai 15%. Ketika konsentrasinya sudah melebihi 15% maka yang terjadi hanyalah adanya kebakaran tanpa ledakan. Dan dibawah 5% maka tidak akan terjadi kebakaran atau ledakan.

Tabel 2.1 Komposisi udara bersih di atmosfer [2]

Gases	Konsentrasi	
	Volume (%)	PPM
Nitrogen	78,08	780.840
Oksigen	20,95	209.460
Argon	0,934	9.340
Karbondioksida	0,033	330
Neon	0,00180	18
Helium	0,00050	5
Metana	0,00020	2
Kripton	0,00010	1

Tabel 2.2 Kadar gas pada area tambang batubara [2]

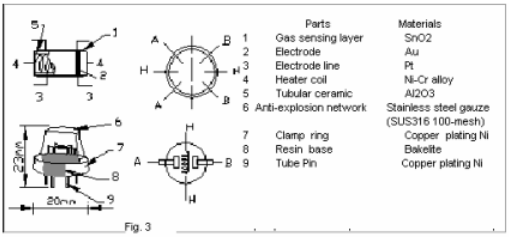
Gas	Ambang batas
O_2	Konsentrasi minimum 19.5%
CH_4	Konsentrasi maksimum 5000 ppm
CO_2	Konsentrasi maksimum 5000 ppm
CO	Konsentrasi maksimum 25 ppm

2.2 Sensor Gas MQ-02

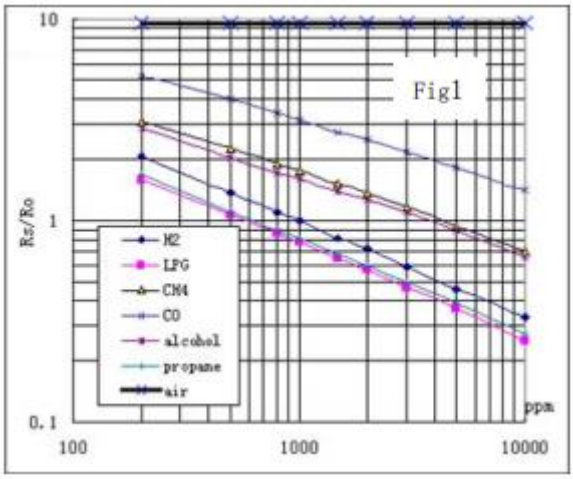
Bahan utama dari sensor ini adalah SnO2 dengan konduktivitas rendah pada udara bersih. Jika terdapat kebocoran gas konduktifitas sensor menjadi lebih tinggi, setiap kenaikan konsentrasi gas maka konduktifitas sensor juga naik. MQ-2 sensitif terhadap gas LPG, Propana, Hidrogen, Karbon Monoksida, Metana dan Alkohol serta gas mudah terbakar diudara lainnya.

Sensor MQ-2 terdapat 2 masukan tegangan yakni VH dan VC. VH digunakan untuk tegangan pada pemanas (Heater) internal dan Vc merupakan tegangan sumber. Catu daya yang dibutuhkan pada sensor MQ-2 adalah Vc < 24VDC dan VH = 5V ±0.2V tegangan AC atau DC. Sensor gas dan asap ini mendeteksi konsentrasi gas yang mudah terbakar di udara serta asap dan output membaca sebagai tegangan analog. Sensor dapat mengukur konsentrasi gas mudah terbakar dari 300 sampai 10.000 sensor ppm. Dapat beroperasi pada suhu dari -20 sampai 50 ° C dan mengkonsumsi kurang dari 150 mA pada 5V. Gambar 2.1 adalah modul dari MQ-02, gambar 2.2 adalah struktur alam MQ-02, dan gambar 2.3 adalah kurva karakteristik sensitivitas dari MQ-02.

Gambar 2.1 Sensor Gas MQ-2 [4].



Gambar 2.2 Struktur MQ-2 [4].



Gambar 2.3 Karakteristik Sensitivitas MQ-2 [4].

Grafik diatas menunjukkan karakteristik sensitifitas dari sensor MQ-2 dalam mendeteksi gas. Sumbu X menandakan konsentrasi gas, sumbu Y menandakan rasio resistensi dari sensor (Rs/Ro). Dimana Rs adalah resistansi dari berbagai macam gas, sementara Ro adalah resistansi sensor dalam 1000ppm *Hydrogen*.

Untuk kalibrasi sensor gas sehingga didapatkan nilai ppm dari gas yang diinginkan yaitu pertama melakukan analisa dari kurva karakteristik sensitivitas dari sensor dan menggunakan rumus *power function* dengan persamaan (2.1), sehingga didapatkan persamaan (2.2) untuk mendapatkan nilai ppm dari gas yang diukur.

$$y = a.x^b \tag{2.1}$$

jadi,

$$ppm = a.(Rs/Ro)^b \tag{2.2}$$

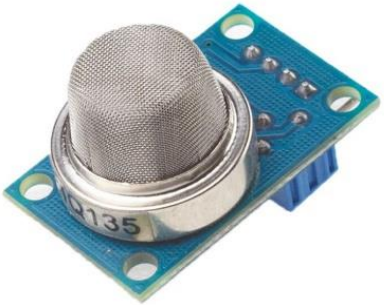
Menggunakan *power regression*, dapat didapatkan *scaling factor* (a), dan *exponent* (b), untuk gas yang akan diukur. Lalu menggunakan persamaan (2.3) untuk mendapatkan nilai Ro.

$$Ro = Rs.sqrt(a/ppm\,, b) = Rs.\exp\left(\ln(a/ppm) \,/\, b\right) \qquad (2.3)$$

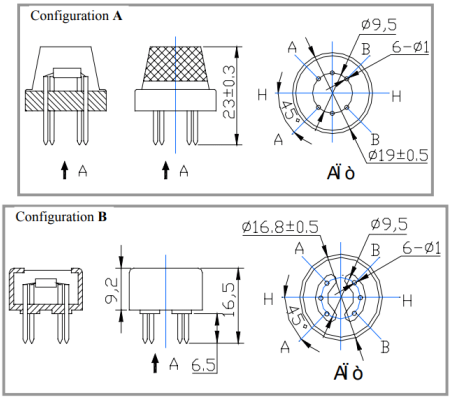
2.3 Sensor Gas MQ-135

Sensor MQ-135 merupakan sebuah modul sensor yang dapat digunakan untuk menentukan kadar konsentrasi gas-gas berbahaya dalam udara. Modul ini berbasiskan sensor MQ-135, yaitu sensor yang dapat mendeteksi gas $CO2$, gas amonia, bensol, alkohol, serta gas berbahaya lainnya. Modul ini cocok digunakan pada proses penentuan kualitas udara (air quality control). Sensor yang biasanyadipakai adalah sensor sensor dengan seri MQ. Sensor MQ-135 membutuhkan waktu pemanasan lebih sedikit, memberikan respon cepat dan sensitivitasnya baik [5].

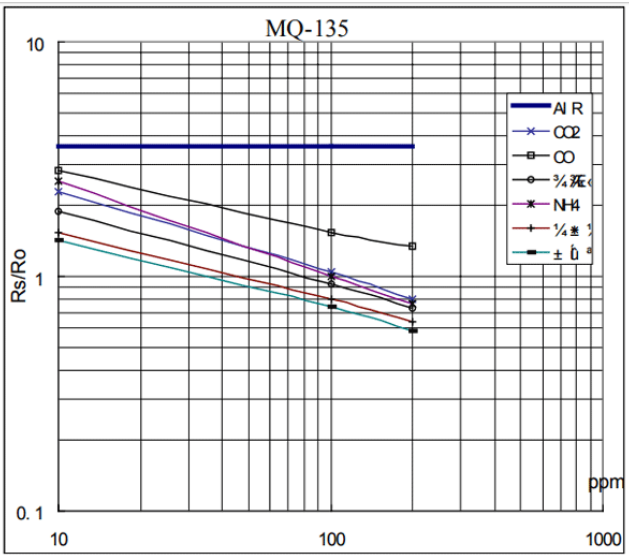
Prinsip kerja dari sensor ini adalah Ketika timah dioksida (partikel semikonduktor) dipanaskan di udara pada suhu tinggi, oksigen diserap di permukaan. Di udara bersih, elektron donor dalam timah dioksida tertarik ke arah oksigen yang diserap pada permukaan bahan sensor. Ini mencegah aliran arus listrik. Dengan adanya gas pereduksi, kerapatan permukaan oksigen yang teradsorpsi berkurang karena bereaksi dengan gas pereduksi. Elektron kemudian dilepaskan ke dalam timah dioksida, yang memungkinkan arus mengalir bebas melalui sensor. Gambar 2.4 adalah modul dari MQ-135, gambar 2.5 adalah struktur dari MQ-135 itu sendiri, dan gambar 2.6 adalah kurva karakteristik sensitivitas dari MQ-135.



Gambar 2.4 Sensor Gas MQ-135 [5].



Gambar 2.5 Struktur MQ-135 [6].



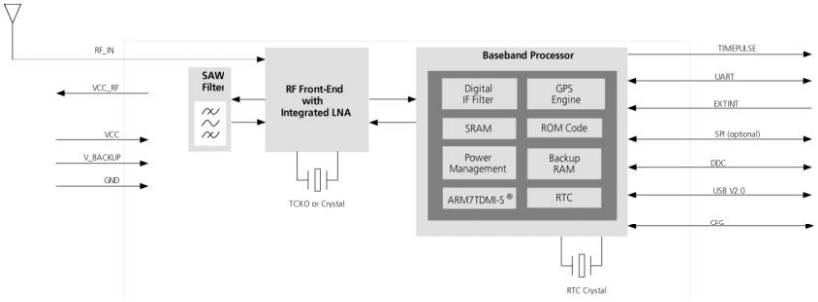
Gambar 2.6 Karakterisitik Sensitivitas MQ-135 [6].

2.4 GPS u-Blox NEO-6M

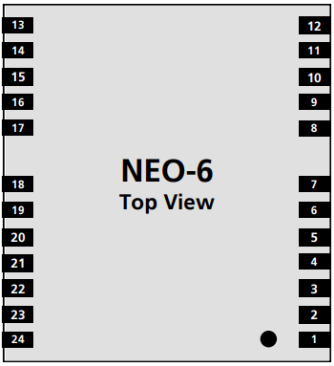
Modul GPS dengan jenis NEO-6M berukuran ringkas ini (25x35mm untuk modul, 25x25mm untuk antena) berfungsi sebagai penerima GPS

yang dapat mendeteksi lokasi dengan menangkap dan memproses sinyal dari satelit navigasi. Jenis GPS ini cukup dapat diandalkan karena memiliki keakuratan yang cukup baik dan juga beberapa fitur yang cukup menguntungkan di antaranya terdapat baterai cadangan data, *built-in* elektronik kompas, dan *built-in* antenna keramik untuk menangkap sinyal dengan kuat. Sistem posisi u-Blox 6 yang memiliki 50 kanal menawarkan *Time-To-First-Fix* (TTFF) dalam waktu dibawah 1 detik. Di mesin akuisisi, dengan 2 juta korelator, mampu melakukan pencarian waktu/frekuensi ruang parallel yang masif, memungkinkannya menemukan satelit secara instan [7]. Gambar 2.7 merupakan bentuk modul dari NEO-6M, gambar 2.8 merupakan blok diagram sistem dari NEO-6M, dan gambar 2.9 adalah pinout dari modul GPS NEO-6M. Tabel 2.3 menjelaskan pin diagram dari NEO-6M.

Gambar 2.7 Modul GPS (NEO-6M) [8].



Gambar 2.8 Blok Diagram Modul NEO-6M [8].



Gambar 2.9 Pinout Modul NEO-6M [8].

Tabel 2.3 Nomor pin, nama pin, tipe pin dan diskripsi pin [8].

Nomor Pin	Nama Pin	Tipe Pin	Diskripsi Pin
1	Reserved	I	Reserved
2	SS_N	I	SPI Slave Select
3	TIMEPULSE	O	Timepulse (1PPS)
4	EXTINT0	I	External Interrupt Pin
5	USB_DM	I/O	USB Data
6	USB_DP	I/O	USB Data
7	VDDUSB	I	USB Supply
8	Reserved	-	See Hardware Integration Manual Pin 8 and 9 must be connected together.
9	VCC_RF	O	Output Voltage RF Section Pin 8 and 9 must be connected together.
10	GND	I	Ground
11	RF_IN	I	GPS Signal Input
12	GND	I	Ground
13	GND	I	Ground
14	MOSI/CFG_COM0	O/I	SPI MOSI / Configuration Pin. Leave open if not used.
15	MISO/CFG_COM1	I	SPI MISO / Configuration Pin. Leave open if not used.
16	CFG_GPS0/SCK	I	Power Mode Configuration Pin / SPI Clock.

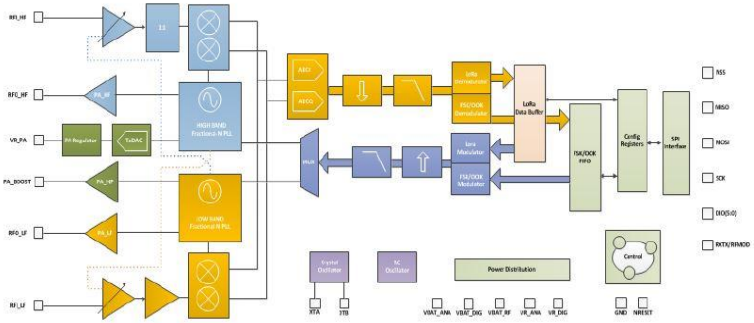
Nomor Pin	Nama Pin	Tipe Pin	Diskripsi Pin
			Leave open if not used.
17	Reserved	I	Reserved
18	SDA2	I/O	DDC Data
19	SCL2	I/O	DDC Clock
20	TxD1	O	Serial Port 1
21	RxD1	I	Serial Port 1
22	V_BCKP	I	Backup Voltage Supply
23	VCC	I	Supply Voltage
24	GND	I	Ground

2.5 LoRa RF96

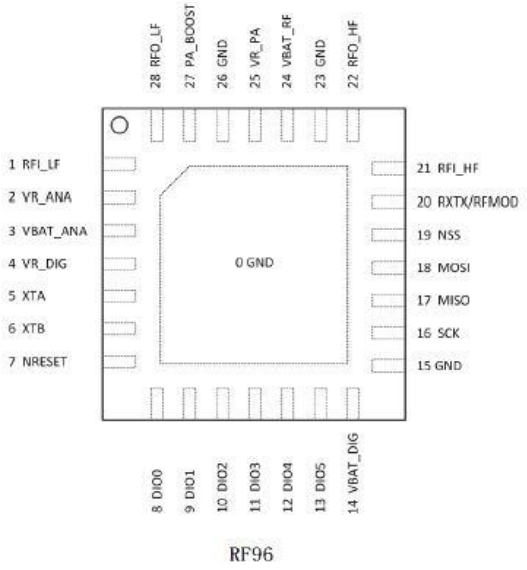
RF96/97/98 menggabungkan modem loRa *spread spectrum* yang mana mampu mencapai jangkauan yang jauh secara signifikan dari sistem yang ada berdasarkan modulasi FSK atau OOK. Dengan skema modulasi baru ini sistem memiliki sensitivitas 8 dB lebih baik daripada FSK yang dapat dicapai dengan referensi kristal berbiaya rendah dan toleransi yang rendah. Peningkatan sensitivitas ini memberikan banyak hal seperti, jangkauan yang lebih panjang dan ketahanan tehadap gangguan tanpa perlu penguatan eksternal. LoRa juga memberikan keuntungan yang signifikan pada selektivitas dan kinerja pemblokiran gangguan, lebih lanjut dapat meningkatkan keandalan dalm komunikasi tanpa kabel. Untuk fleksibilitas maksimum, pengguna dapat menentukan penyebaran spectrum pada modulasi *bandwidth* (BW), faktor penyebaran (SF) dan tingkat koreksi kesalahan (CR). Keuntungan lainnya dari *spread modulation* adalah bahwa setiap faktor penyebaran bersifat ortogonal - sehingga beberapa sinyal yang ditransmisikan dapat menempati saluran yang sama tanpa terinteferensi. Hal ini juga memungkinkan koeksistensi sederhana dengan sistem berbasis FSK yang ada. Standar modulasi GFSK, FSK, OOK, dan GMSK juga disediakan untuk memungkinkan terjadinya kompatibilitas dengan sistem yang telah ada atau standar yang sudah ada seperti MBUS nirkabel dan IEEE 802.15.4g [9].

Kementerian Komunikasi dan Informatika (Kominfo) menyisakan setidaknya lima kebijakan untuk dirampungkan pada 2018. Dua di antaranya yakni standardisasi Internet of Things (IoT) dan konsolidasi operator rencananya bakal dirilis Kuartal I-2019. Salah satu yang akan diatur adalah optimalisasi penggunaan frekuensi. Kebijakan ini nantinya bakal mengatur terkait batasan, optimalisasi frekuensi, metode

pelaksanaan, dan kewajiban setiap pengguna yang memegang izin pita frekuensi radio [10]. Gambar 2.10 merupakan gambar diagram blok dari RF96, gambar 2.11 merupakan gambar diagram pin RF96, dan tabel 2.4 merupakan penjelasan dari diagram pin RF96.



Gambar 2.10 *Skematik blok diagram* [9].



Gambar 2.11 Pin diagram RF96 [9].

Tabel 2.4 Nomor pin, nama pin, tipe pin dan diskripsi pin [9].

Nomor Pin	Nama Pin	Tipe Pin	Diskripsi Pin
1	RFI_LF	I	RF input for lower bands
2	VR_ANA	-	Regulated supply voltage for analogue circuitry
3	VBAT_ANA	-	Supply voltage for analogue circuitry
4	VR_DIG	-	Regulated supply voltage for digital blocks
5	XTA	I/O	XTAL connection or TCXO input
6	XTB	I/O	XTAL connection
7	NRESET	I/O	Reset trigger input
8	DIO0	I/O	Digital I/O, software configured.
9	DIO1/DCLK	I/O	Digital I/O, software configured.
10	DIO2/DATA	I/O	Digital I/O, software configured.
11	DIO3	I/O	Digital I/O, software configured.
12	DIO4	I/O	Digital I/O, software configured.
13	DIO5	I/O	Digital I/O, software configured.
14	VBAT_DIG	-	Supply voltage for digital blocks
15	GND	-	Ground
16	SCK	I	SPI Clock input
17	MISO	O	SPI Data output
18	MOSI	I	SPI Data input
19	NSS	I	SPI Chip select input
20	RXTX/RF_MOD	O	Rx/Tx switch control: high in Tx
21	RFI_HF	I	RF input for upper bands
22	RFO_HF	O	RF output for upper bands
23	GND	-	Ground
24	VBAT_RF	-	Supply voltage for RF blocks
25	VR_PA	-	Regulated supply for the PA
26	GND	-	Ground
27	PA_BOOST	O	Optional high-power PA output, lower or upper bands
28	RFO_LF	O	RF output for lower bands

2.6 *Microcontroller* Arduino Uno

Arduino UNO merupakan sebuah board mikrokontroler yang didasarkan pada ATmega328. Arduino UNO sendiri memiliki 14 pin digital yang bias digunakan sebagai input atau output (6 pin di antaranya dapat digunakan sebagai output PWM), selain itu juga memiliki 6 input analog, sebuah osilator Kristal 16 MHz, sebuah koneksi USB, sebuah

power jack, sebuah ICSP *header*, dan sebuat tombol reset. Arduino Uno berbeda dari semua board Arduino sebelumnya, Arduino UNO tidak menggunakan chip driver FTDI USB-to-serial.

Arduino UNO dapat disuplai melalui koneksi USB atau dengan sebuah power suplai eksternal. Suplai eksternal (non-USB) untuk Arduino dapat diperoleh dari sebuah adaptor AC ke DC atau dari sebuah battery. Pemasangan adaptor dapat dilakukan dengan mencolokkan sebuah center-positive plug yang panjangnya 2,1 mm ke power jack dari board. Sedangkan untuk pemasangan baterai, kabel negatif dari sebuah baterai dapat dimasukkan ke dalam header / kepala pin Ground (Gnd) dan kabel positif dari baterai dapat dimasukkan ke dalam header / kepala pin Vin pada Arduino.

Board Arduino UNO beroperasi pada sebuah suplai tegangan eksternal 6 sampai 20 Volt. Jika disuplai dengan tegangan yang lebih kecil dari 7 V, contohnya ketika diberi tegangan 5 Volt mungkin akan mensuplai lebih kecil dari 5 Volt dan board Arduino UNO bisa jadi tidak stabil. Jika suplai yang digunakan lebih besar dari 12 Volt, voltage regulator yang ada di dalam Arduino bisa kelebihan panas dan dapat membahayakan board Arduino UNO. Range tegangan suplai eksternal yang direkomendasikan adalah 7 sampai 12 Volt [11].

Berikut adalah *power pin* yang dimiliki Arduino Uno:

1. **VIN:** Pin ini dapat digunakan sebagai sumber *external* (*unregulated*) untuk daya pada Arduino yang dapat diberi *input* sebesar 7-12V.
2. **5V:** Pin ini merupakan pin 5V hasil dari tegangan yang telah melewati *regulator.*
3. **3V3:** Pin ini merupakan pin 3,3V hasil dari tegangan yang telah melewati *regulator* dengan arus maksimal 50mA.
4. **GND:** *Ground pin.*

Arduino Uno memiliki 6 pin analog *input* atau biasa disebut ADC (*Analog to Digital Converter*) dengan resolusi 8 - 10 bit, dengan label A0 sampai A5. Selain itu Arduino unojuga memiliki 14 pin digital yang 6 diantaranya dapat digunakan sebagai PWM, dengan label angka 0 sampai 13. pada Arduino Mega juga terdapat juga pin yang memiliki beberapa fungsi khusus yaitu *serial ports, external interrupts, PWM (Pulse Width Modulation), SPI (Serial Peripheral Interface),* LED *pin* 13, *TWI (Two Wire Interface),* AREF (*Analog Reference*), dan *Reset*. Berikut ini untuk lebih jelasnya:

1. **Serial: 0 (RX) dan 1 (TX)**. Pin ini digunakan untuk berkomunikasi dengan protokol *serial* yang menggunakan dua pin TX (*transmit*)

dan RX (*receive*). Setiap TX dan RX berpasangan sebagai berikut; Serial: 0 (RX) and 1 (TX).

2. **External Interrupts:** Pin ini digunakan untuk memicu masuknya pogram ke program *interrupt.* Pin ini terdiri dari: 2 (interrupt 0), 3 (interrupt 1).
3. **PWM:** pin ini Memberikan 8-bit PWM output dengan fungsi *analogWrite()*. Pin ini terdiri dari pin 3 - 11
4. **SPI:** Pin ini dapat mensupport komunikasi SPI menggunakan SPI *library*. Pin SPI tediri dari pin 10 (SS), 11 (MOSI), 12 (MISO), dan 13 (SCK).
5. **LED:** Ada sebuah LED yang terpasang, terhubung ke pin digital 13. Ketika pin bernilai HIGH LED menyala, ketika pin bernilai LOW LED mati.
6. **TWI:** Pin ini dapat mensupport komunikasi TWI dengan menggunakan *Wirelibrary*. Pin itu adalah pin A4 atau SDA dan pin A5 atau SCL.
7. **AREF**: Pin ini digunakan oleh Arduino sebagai tegangan referensi *analog input*.
8. **Reset**. Membawa saluran ini LOW untuk mereset mikrokontroler. Secara khusus, digunakan untuk menambahkan sebuah tombol reset untuk melindungi yang memblock sesuatu pada board.

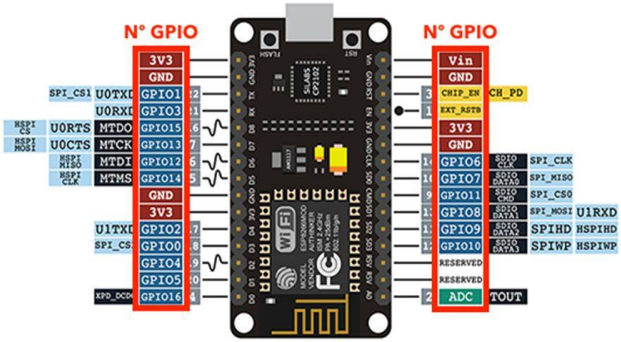
Memori yang ada pada Arduino Uno berbasis pada ATmega328 yang mempunyai memori sebesar 32 KB (dengan 0,5 KB digunakan untuk bootloader). ATmega 328 juga mempunyai 2 KB SRAM dan 1 KB EEPROM (yang dapat dibaca dan ditulis (RW/read and written) dengan EEPROM library). Untuk melihat bentuk fisik dari Arduino uno dapat dilihat pada gambar 2.12.

Gambar 2.12 Board Arduino Uno [11].

2.7 NodeMCU ESP8266

NodeMCU merupakan sebuah Firmware Open-Source dan *Development Kit* yang dapat membantu membangun produk berbasis IoT. NodeMCU dikembangkan untuk mempermudah Penggunaan API (Application Programming Interface) yang canggih untuk perangkat keras IO. API dapat mengurangi pekerjaan berlebih untuk mengkonfigurasi dan memanipulasi perangkat keras. NodeMCU dirancang memiliki *Input* dan *Output* seperti perangkat keras Arduino (IO). NodeMCU menggunakan MCU Wi-Fi dengan biaya terendah yaitu ESP 8266. ESP8266 Merupakan Chip Wi-Fi paling terintegrasi. Ukuran Chip adalah 5mm x 5mm. ESP8266EX minimal membutuhkan rangkaian eksternal dan pengintegrasian 32-bit Tensilica MCU, standar antarmuka digital periferal, sakelar antena, balun R, *Power Amplifier*, penguat penerima dengan *noise* rendah, filter dan modul manajemen daya. Semua diintegrasikan dalam satu paket kecil.

ESP8266EX mengintegrasikan Tensilica L106 *Micro* 32-bit *Controller* (MCU) dengan fitur ekstra konsumsi daya yang rendah dan 16-bit RSIC, dapat mencapai kecepatan clock maksimum 160 MHz. Dengan Real Time Operation System (RTOS) yang diaktifkan dan WiFi *stack* yang berfungsi, sekitar 80% pemrosesan daya masih tersedia untuk pemrograman dan pengembangan aplikasi pengguna. NodeMCU memiliki Pin tegangan *input* dan *output* sebesar 3.3 V pada setiap GPIO. Menyediakan pin tegangan 3.3 Volt sebanyak 3 buah. Mempunyai 16 buah GPIO [12]. Bentuk fisik dari NodeMCU dan *pinout* dapat dilihat pada gambar 2.13.

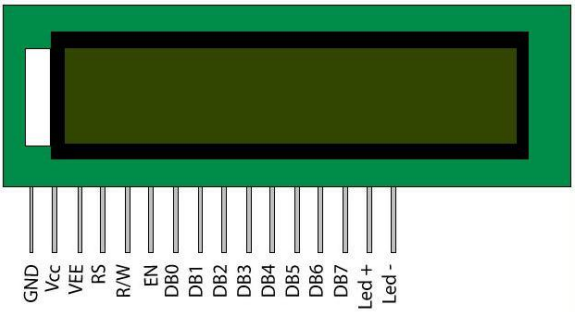


Gambar 2.13 NodeMCU dan pinout [12].

2.8 LCD I2C 16X2

Layar LCD (Liquid Crystal Display) adalah modul penampil elektronik dan dapat digunakan dalam berbagai aplikasi. Layar LCD 16x2 adalah modul yang sangat dasar dan sangat umum digunakan di berbagai perangkat dan sirkuit. LCD 16x2 berarti dapat menampilkan 16 karakter per baris dan ada 2 baris pada layar. LCD ini memiliki dua register, yaitu, *Command* dan *Data*. Register *Command* menyimpan instruksi perintah yang diberikan kepada LCD. Perintah adalah instruksi yang diberikan kepada LCD untuk melakukan tugas yang telah ditentukan seperti menginisialisasi, membersihkan layarnya, mengatur posisi kursor, mengontrol tampilan dll. Register *Data* menyimpan data yang akan ditampilkan pada LCD. Data merupakan kode ASCII dari karakter yang akan ditampilkan pada LCD. Gambar pin yang ada pada modul LCD 16X2 dapat dilihat pada gambar 2.14.

Modul LCD I2C merupakan modul untuk antarmuka LCD dengan mikrokontroler. Biasanya, untuk mengkoneksikan layar LCD ke mikokontroler akan menghabiskan pin yang ada pada mikrokontroler dengan mudah, terutama ketika mengkoneksikannya dengan Arduino Uno. Selain itu akan sangat rumit untuk membuat jalur. Setelah LCD dikoneksikan modul LCD I2C 16x2 dan selanjutnya dikoneksikan dengan Arduino, Itu berarti hanya akan membutuhkan 4 pin untuk layar LCD yaitu VCC, GND, SDA, SCL. Ini akan menghemat setidaknya 4 pin digital / analog pada Arduino. Untuk menghindari pertentangan alamat I2C dengan perangkat I2C lainnya, seperti sensor ultrasonik, IMU, akselerometer dan giroskop, alamat I2C modul dapat dikonfigurasi dari 0x20-0x27. Dan kontrasnya dapat disesuaikan secara manual. Gambar pin yan ada pada modul LCD I2C 16X2 dapat dilihat pada gambar 2.15.



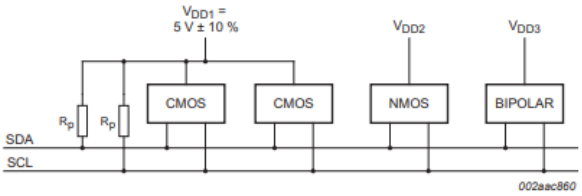
Gambar 2.14 Pin diagram LCD 16X2

Gambar 2.15 Modul I2C untuk LCD [13].

2.9 Komunikasi I2C

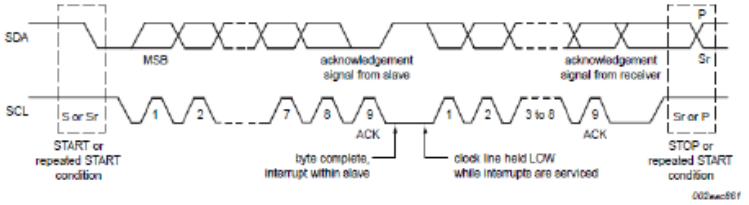
Komunikasi I2C merupakan komunikasi dengan tipe serial *synchronous* yaitu komunikasi dengan menyamakan ketukan transfer data, dengan model *two wire interface* (dua kabel antamuka) yang dapat mengirimkan data digital melalui jalur bus. Komunikasi I2C pertama kali diperkenalkan oleh Philips Semiconductor. Alamat perangkat yang menggunakan komunikasi I2C terdiri dari 7-bit data alamat dan 1-bit konfigruasi R/W (*read* atau *write*). I2C memiliki protokol komunikasi yang terdiri dari *start condition*, MSB data, LSB data, ACK, dan *stop condition*. I2C memiliki 2 jalur data, yaitu SDA dan SCL. Jalur data ini membawa informasi antara perangkat yang terhubung ke bus. Setiap perangkat dikenali oleh alamat unik (perangkat tersebut seperti mikrokontroler, *driver* LCD, memori atau antarmuka *keyboard*) dan setiap perangkat dapat beroperasi sebagai pemancar atau penerima, tergantung pada fungsi perangkat. Driver LCD mungkin hanya berfungsi sebagai penerima data, sedangkan memori dapat menerima dan mengirim data. Baik SDA dan SCL merupakan saluran dua arah, kedua jalur tersebut terhubung ke tegangan suplai positif melalui sumber arus atau resistor pull-up seperti pada gambar. Ketika bus tidak tersambung apapun, kedua jalur itu belogika tinggi. Pada bagian output perangkat yang terhubung ke bus harus memiliki jalur *open-drain* atau *open-collector* untuk untuk mendukung kerja fungsi *wired-AND* [13]. Pada kedua jalur tersebut perlu diberikan resistor *pull-up* karena bersifat *open drain*. Pemilihan resistor *pull-up* dapat menggunakan persamaan (2.4) dan gambar 2.16 menunjukkan bahwa komunikasi I2C dapat digunakan untuk lebih dari satu komunikasi, sedangkan protokol komunikasi I2C dapat dilihat pada gambar 2.17.

$$R_{min} = \frac{V_{CC} - V_{OL}}{I_{OL}} \text{ dan } R_{max} = \frac{T_{SMBus}}{C_{Bus}} \quad (2.4)$$



VDD2, VDD3 are device-dependent (for example, 12 V).

Gambar 2.16 Perangkat berbagai tegangan suplai dan bus sama [13].



Gambar 2.17 Transfer data pada bus I2C [13].

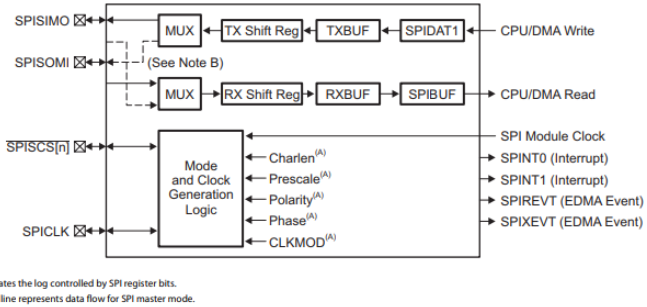
2.10 Komunikasi SPI

SPI merupakan port input / output serial sinkron berkecepatan tinggi yang memungkinkan aliran bit serial dari bit panjang terprogram (2 hingga 16 bit) untuk digeser masuk dan keluar dari perangkat pada laju transfer bit terprogram. SPI biasanya digunakan untuk komunikasi antar perangkat dan periferal eksternal. Aplikasi yang umum termasuk antarmuka ke eksternal I / O atau ekspansi periferal melalui perangkat seperti register geser, driver layar, SPI EPROMS, dan konverter analog-ke-digital. Komunikasi SPI memiliki beberapa fitur seperti:

1. 16-bit shift register
2. 16-bit Receive buffer register (SPIBUF) and 16-bit Receive buffer emulation alias register (SPIEMU)
3. 16-bit Transmit data register (SPIDAT0) and 16-bit Transmit data and format selection register (SPIDAT1)
4. 8-bit baud clock generator
5. Serial clock (SPICLK) I/O pin
6. Slave in, master out (SPISIMO) I/O pin

7. Slave out, master in (SPISOMI) I/O pin
8. Multiple slave chip select (SPISCS[n]) I/O pins (4 pin mode only)
9. Programmable SPI clock frequency range
10. Programmable character length (2 to 16 bits)
11. Programmable clock phase (delay or no delay)
12. Programmable clock polarity (high or low)
13. Interrupt capability
14. DMA support (read/write synchronization events)
15. Up to 66 MHz operation

Komunikasi SPI memungkinkan perangkat lunak untuk memprogram dengan beberapa pilihan seperti, Frekuensi SPICLK (modul SPI Clock / 2 hingga modul SPI Clock / 256). Sedangkan untuk penggunaan pin dapat digunakan 3-pin (MISO, MOSI, dan SCK) atau 4-pin (MISO, MOSI, SCK, dan SS atau Chip Select). Panjang karakter yang dikirim mulai dari 2 bit hingga 16 bit dan arah gesernya MSB/LSB terlebih dahulu. Fase clock bisa diberi Penundaan atau tanpa penundaan dan polaritas dapat belogika tinggi atau rendah. Dalam mode master memungkinkan keterlambatan antar transmisi. Dalam mode master terdapat pengaturan *chip select*. Fungsi *Chip Select* akan ditahan dalam mode master. Gambar 2.18 merupakan gambar yang menunjukkan blok diagram dari komunikasi SPI [14].



Gambar 2.18 Blok diagram komunikasi SPI [14].

2.11 Komunikasi Serial

Universal Asynchronous Receiver / Transmitter (UART) merupakan protokol komunikasi serial standar untuk bertukar data antara dua perangkat. Dalam protokol komunikasi ini, data ditransfer secara berurutan, satu bit dalam satu waktu. Implementasi ini menggunakan

bingkai, dalam satu bingkai terdiri dari 8 bit data, satu bit mulai, satu opsional bit paritas, dan satu atau beberapa bit *stop*. Bit paritas merupakan 1 bit dengan nilai 0 atau 1, yang ditambahkan ke blok data untuk tujuan pendeteksian kesalahan. Bit ini bersifat opsional, itu munkin atau tidak mungkin ditambahkan ke muatan data. Juga dapat diatur menjadi ganjil atau genap. Bit-bit ini sering digunakan dalam transmisi data untuk memastikan data tersebut tidak rusak selama proses transfer. Jika protokol transmisi data diatur ke paritas ganjil, masing-masing paket data harus memiliki paritas ganjil. Jika disetel genap, setiap paket harus memiliki paritas genap. Jika paket diterima dengan paritas yang salah, akan muncul kesalahan dan data harus dikirim ulang. Bit paritas untuk setiap paket data dihitung sebelum data dikirimkan. Karena protokol UART bersifat tidak sinkron, protokol ini tidak memerlukan sinyal *clock*. Dalam komunikasi UART, kecepatan ditentukan oleh *baud rate*. *Baud rate* didefinisikan sama dengan jumlah bit yang ditransmisikan per detik termasuk bit mulai dan berhenti. Untuk pengiriman dan penerimaan data yang andal dan tanpa adanya bit yang hilang, baik pemancar dan penerima harus memiliki *baud rate* yang sama. Ketidakcocokan *baud rate* antara pemancar dan penerima biasanya menghasilkan kesalahan *framing*. *Baud rate* yang umum digunakan adalah 4800, 9600, 19200, 38400, 57600, dan 115200. Tetapi *rate* lainnya mungkin juga bisa digunakan.

Modul Pengguna UART mengimplementasikan pemancar dan penerima serial. Peta UART menjadi dua PSoC Blok Komunikasi Digital yang didesain menjadi TX dan RX. TX Blok PSoC memberikan fungsionalitas pemancar dan RX blok PSoC memberikan fungsionalitas penerima. RX dan TX beroperasi secara independen. Masing-masing memiliki daftar Kontrol dan Status mereka sendiri, dapat diberikan program interupsi, *Input* / *Output*, register *Buffer*, dan register *Shift*. Baik TX dan RX berbagi format *enable*, *clock*, dan *data* yang sama. Mengatur bit *enable* pada register Kontrol RX dan Kontrol TX memungkinkan UART untuk beroperasi. Mengaktifkan dan menonaktifkan dilakukan menggunakan fungsi yang telah disediakan API (*Application Programming Interface*). Modul *clock* Pengguna UART dibagikan oleh komponen RX dan TX. Frekuensi *clock* yang dipilih harus delapan kali frekuensi dari laju bit data yang diperlukan. Setiap bit data yang diterima atau dikirim membutuhkan delapan siklus *clock input*. *Clock* ini dikonfigurasi menggunakan Editor Perangkat PSoC Designer. Data yang diterima dan dikirim adalah bit *stream* yang terdiri dari bit awal, delapan bit data, bit *opsional parity*, dan bit *stop*. Paritas dapat diatur menjadi

tidak ada, genap, atau ganjil, dan diatur menggunakan PSoC Designer Editor atau menggunakan API UART. Baik RX dan TX diatur ke konfigurasi paritas yang sama [15]. Blok diagram dari komunikasi serial dapat dilihat pada gambar 2.19. Untuk diagram dari RX dan TX timing dapat dilihat pada gambar 2.20 dan 2.21.

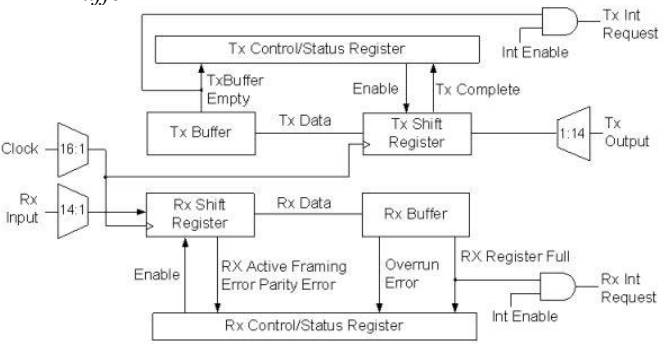
2.11.1 TX – UART Transmitter

Transmitter menggunakan register TX *Buffer*, TX *Shift*, dan TX *Control* dari tipe *Digital Communications Blok PSoC*. Register TX *control* diinisialisasi dan dikonfigurasi menggunakan UART *User Module firmware API routines*. Ketika bit *enable* di register Kontrol TX diatur, bit *clock* internal dibagi delapan akan dihasilkan. Sebuah *byte* data untuk pengiriman ditulis oleh API *routine* ke dalam register *Buffer* TX, pembersihan TX *Buffer Empty* bit status di register *control* TX. Bit status ini dapat digunakan untuk mendeteksi dan mencegah *overrun* pengiriman *error*. Kenaikan dari bit *clock* berikutnya akan mentransfer data ke register *Shift* dan mengatur bit TX *Buffer Empty* di register *control* TX. Jika *interrupt enable mask* diaktifkan, perintah interupsi akan dipicu. Interupsi ini memungkinkan antrian *byte* berikutnya untuk dikirim. Jadi ketika *byte* data saat ini sepenuhnya ditransmisikan, *byte* baru ditransmisikan pada transmisi *clock* berikutnya yang tersedia. Bit mulai ditransmisikan pada saat yang sama dengan *byte* data ditransfer dari register TX *Buffer* ke register TX *Shift*. Bit *Clock* berturut-turut menggeser bit *Stream* serial ke *output*. Bit *stream* tersusun dari masing-masing bit *byte* data, bit LSB (*Least Significant Bit*), bit *paritas opsional*, dan bit *stop*. Ketika bit berhenti sepenuhnya ditransmisikan, bit TX *Complete Status* pada register TX *Control* diatur. Bit ini tetap ada dan valid hingga dibaca. Jika byte data baru telah ditulis ke register TX *Buffer*, *byte* data ditransfer ke register TX *Shift* dan transmisi data dimulai pada kenaikan berikutnya dari bit *Clock*.

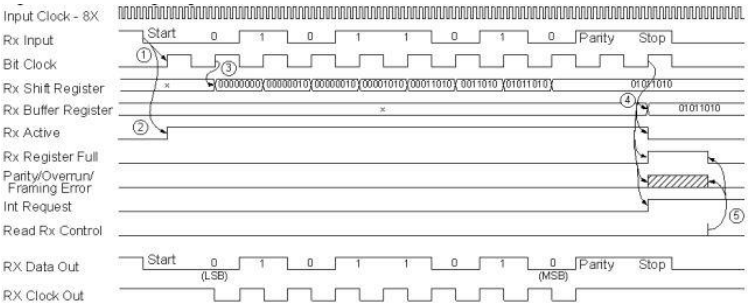
2.11.2 RX – UART Receiver

Receiver menggunakan register RX *Buffer*, RX *Shift*, dan RX *Control* dari tipe *Digital Communications Blok PSoC*. Register Kontrol RX diinisialisasi dan dikonfigurasi menggunakan UART *User Module firmware API routines*. Inisialisasi RX terdiri dari pengaturan paritas UART, secara opsional memungkinkan terjadinya interupsi pada register RX *Full Condition*, kemudian aktifkan UART. Ketika bit awal terdeteksi pada *input* RX, bit clock yang terbagi delapan dimulai dan disinkronkan

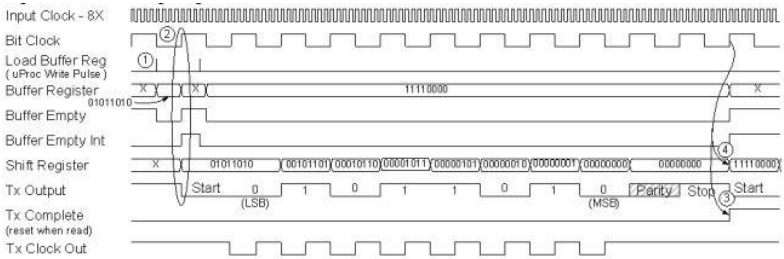
ke sampel data di tengah bit yang diterima. Pada kenaikan bit clock yang terbagi delapan berikutnya, *input* data diambil sampelnya dan dialihkan ke register RX *Shift*. Jika paritas diaktifkan, bit *clock* berikutnya akan mengambil sampel bit paritas. Pengambilan sampel dari bit *stop*, pada bit *clock* berikutnya, menghasilkan transfer *byte* data yang diterima ke register RX *Buffer*.



Gambar 2.19 Blok diagram UART [15].



Gambar 2.20 Diagram RX timing [15].



Gambar 2.21 Diagram TX timing [15].

2.12 Aplikasi Blynk

Blynk merupakan platform sebuah layanan *server* yang digunakan untuk mendukung project *Internet of Things* dengan aplikasi berbasis iOS dan Android yang dapat dikontrol dengan Arduino atau Raspberry Pi dengan menggunakan akses Internet. Blynk merupakan dasbor digital di mana aplikasi ini dapat membangun antarmuka grafis bagi pengguna dengan tombol hanya drag dan drop [16]. Blynk diciptakan dengan tujuan untuk *control* dan *monitoring hardware* secara jarak jauh menggunakan komunikasi data internet ataupun intranet (jaringan LAN). Kemampuan menyimpan data dan menampilkan data secara visual baik menggunakan angka, warna ataupun grafis semakin memudahkan dalam pembuatan *project* di bidang *Internet of Things.* Gambar 2.22 merupakan ilustrasi hubungan antara Blynk dengan perangkat IoT. Terdapat 3 komponen utama Blynk, yaitu:

2.12.1 Blynk Apps

Blynk Apps memungkinkan untuk membuat proyek antarmuka dengan berbagai macam komponen input dan output. Blynk mendukung untuk pengiriman maupun penerimaan data serta merepresentasikan data tersebut sesuai dengan komponen yang dipilih. Representasi data dapat berbentuk visual angka maupun grafik. Terdapat 4 kategori komponen pada aplikasi Blynk, yaitu:

1. Controller: berfungsi untuk mengirim data atau perintah ke hardware.
2. Display: berfungsi untuk menampilkan data yang berasal dari hardware ke smarthphone.
3. Notification: befungsi untuk mengirim pesan notifikasi.

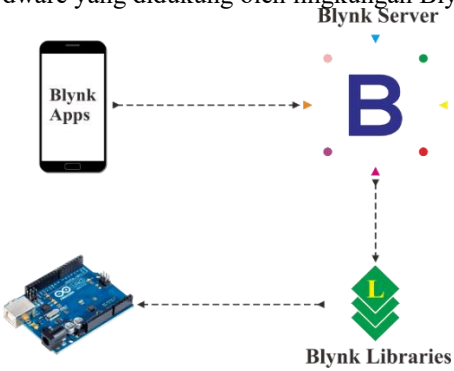
4. Interface: berfungsi untuk mengatur tampilan pada aplikasi Blynk, dapat berupa menu ataupun tab.
5. Others: komponen tambahan yang tidak termasuk pada 3 kategori sebelunya seperti Bridge, RTC, dan Bluetooth.

2.12.2 Blynk Server

Blynk *server* merupakan fasilitas *back-end service* berbasis *cloud* yang bertanggung jawab untuk mengatur komunikasi antara aplikasi *smartphone* dengan lingkungan hardware. Kemampun untuk menangani hardware yang banyak pada saat yang bersamaan semakin memudahkan untuk membuat projek dengan sistem IoT. Blynk *server* juga tersedia dalam bentuk server lokal apabila digunakan pada lingkungan tanpa internet. Blynk server lokal bersifat terbuka dan dapat diimplementasikan pada sederhana sekalipun, contohnya *Raspberry Pi* atau Arduino.

2.12.3 Blynk Library

Blynk *Library* berfungsi untuk membantu pengembangan code. Blynk library tersedia pada banyak platform perangkat keras sehingga semakin mempermudah dalam pembuatan proyek berbasis IoT dengan fleksibilitas hardware yang didukung oleh lingkungan Blynk.

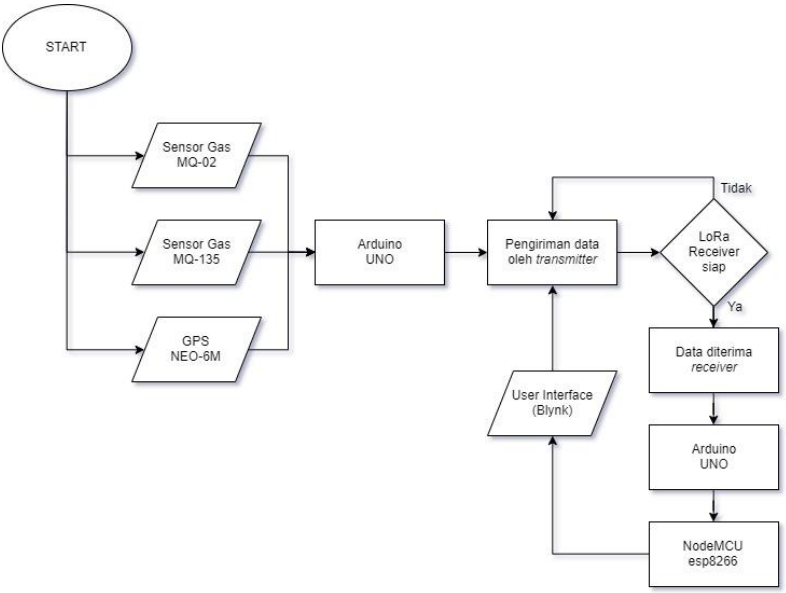


Gambar 2.22 Ilustrasi hubungan antara BLYNK dan perangkat IoT [16].

BAB III
PERANCANGAN SISTEM

3.1 Gambaran Umum Sistem

Pada bab ini dijelaskan perancangan sistem secara keseluruhan. Alat yang dirancang bertujuan untuk memonitoring kadar gas di area tambang batubara terutama kadar gas metana (CH_4), dan gas karbon dioksida (CO_2). Untuk flowchart dari sistem dapat dilihat pada gambar 3.1.



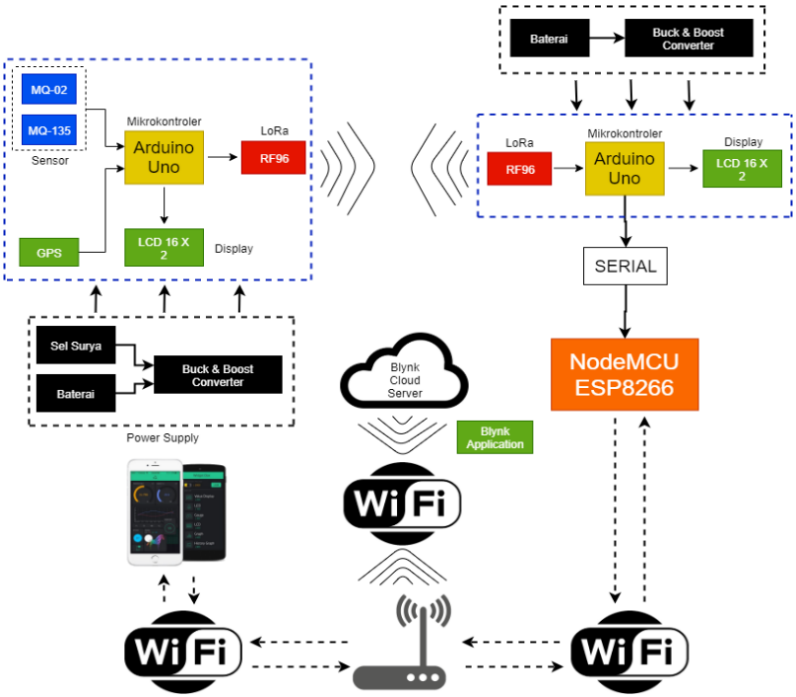
Gambar 3.1 Flowchart sistem pemantauan

Perancangan alat berupa sebuah *board* yang berisi mikrokontroler Arduino Uno dilengkapi dengan dua sensor gas, modul GPS, LoRa (RF96), LCD display, NodeMCU, dan terminal catu daya. Sensor gas digunakan untuk mengetahui kadar gas di area tambang batubara. Modul GPS akan mengambil data dari satelit untuk mengetahui lokasi dari alat ini dengan menunjukkan *latitude* dan *longitude*. LCD digunakan untuk menampilkan data yang didapatkan dari sensor. Mikrokontroler akan membaca data dari sensor-sensor yang terhubung, kemudian data tersebut

akan dikirim dengan modul LoRa. Alat dilengkapi dengan sel surya sebagai sumber daya pada bagian kotak *transmitter*. Data yang dikirim akan diterima oleh mikrokontroler pada kotak *receiver* melalui modul LoRa. Selanjutnya data diolah dan diteruskan ke NodeMCU dan LCD display. Setelah data diterima oleh NodeMCU, selanjutnya data akan dikirim ke aplikasi Blynk di *smartphone* untuk memonitoring kadar gas dan lokasi dari *transmitter*.

3.2 Diagram Blok Sistem

Pada tugas akhir ini, mikrokontroler yang digunakan sebagai unit pengolah data yaitu Arduino Uno. Untuk blok diagram dapat dilihat pada gambar 3.2.



Gambar 3.2 Diagram Blok Sistem

Sensor yang digunakan adalah sensor gas MQ-02 dan sensor gas MQ-135. Kedua sensor tersebut diletakkan pada kotak *transmitter*

bersama dengan modul GPS, LCD, LoRa, serta terminal catu daya. Sensor MQ-02 digunakan untuk mengetahui kadar gas CH_4, dan sensor MQ-135 digunakan untuk mengetahui kadar gas CO_2. Sedangkan LoRa digunakan untuk mengirim data sensor yang telah didapat. Sumber tegangan dari mikrokontroler dan sensor didapat dari sel surya maupun baterai Li-Po yang telah diturunkan tegangannya terlebih dahulu menggunakan *buck converter*.

Sensor gas MQ-02, MQ-135 dan modul GPS dibaca oleh mikrokontroler Arduino Uno setiap 8 detik sekali. Hasil pembacaan dari sensor ini akan dikirimkan dengan menggunakan modul LoRa. Selain itu akan ditampilkan pada LCD display.

Data yang dikirim akan diterima dengan LoRa pada kotak *receiver*. Pada kotak *receiver* juga terdapat mikrokontroler Arduino Uno untuk mengolah data sensor yang diterima odari LoRa. Kemudian juga ada mikrokontroler NodeMCU ESP8266 untuk mengirimkan data yang diterima ke aplikasi Blynk di *smartphone*.

3.3 Perancangan Perangkat Keras

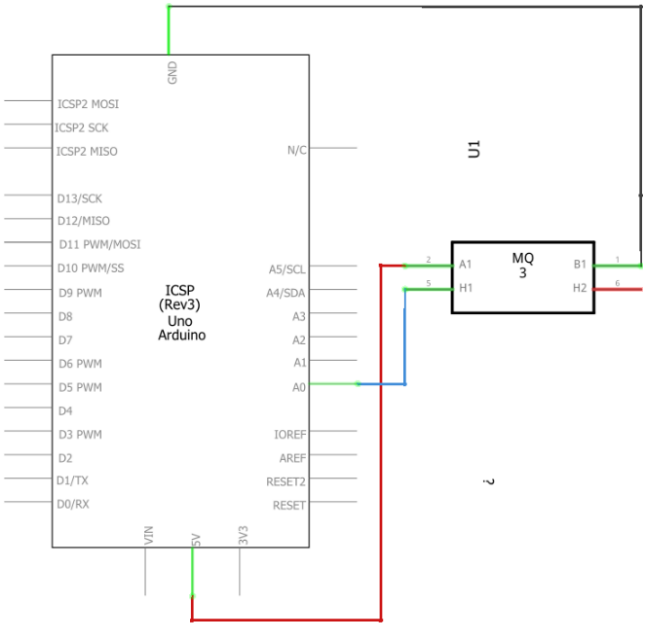
Salah satu perancangan perangkat keras pada tugas akhir ini yaitu desain elektronik. Desain elektronik meliputi pembuatan *board* untuk *transmitter* yang berisi Arduino Uno, LoRa Shield + RF96, sensor gas MQ-02 dan MQ-135, modul GPS, LCD dan terminal catu daya. Selain itu juga pembuatan *board* untuk *receiver* yang berisi Arduino Uno, LoRa Shield + RF96, LCD, NodeMCU dan terminal catu daya.

3.3.1 Sensor Gas MQ-02

Pada perancangan sensor untuk mendeteksi gas CH_4 dilakukan dengan menggunakan sensor berjenis MQ-02. Sensor MQ-02 ini dihubungkan dengan Arduino Uno. Ada 4 pin yang ada pada sensor MQ-02, yaitu pin VCC, Ground, Digital Output, dan Analog Output. Pin yang digunakan hanya tiga, yaitu pin suplai (VCC dan Ground) dan pin untuk membaca nilai tegangan sensor yang bisa dipilih antara pin analog output atau digital output. Pada perancangan ini digunakan pin analog output. Tegangan yang keluar dari analog output akan dibaca dengan pin analog input Arduino Uno, sedangkan untuk suplai sensor bisa didapat dari tegangan suplai 5 Volt. Untuk hubungan antara sensor dan Arduino dapat dilihat pada tabel 3.1. Untuk ilustrasi gambar hubungan antara sensor MQ-02 dengan Arduino Uno dapat dilihat pada gambar 3.3.

Tabel 3.1 *Wiring* pin antara pin Arduino dan sensor MQ-02

Arduino Uno	Sensor MQ-02
Pin 5 Volt	Pin VCC
Pin Ground	Pin Ground
Pin Analog A0	Pin Analog Output (AOUT)



Gambar 3.3 *Wiring* diagram sensor gas CH_4 MQ-02 dengan Arduino

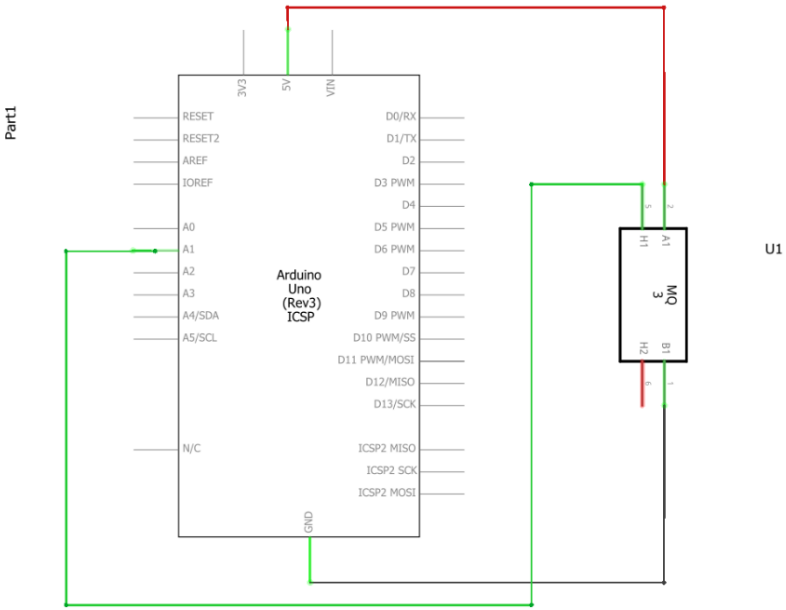
3.3.2 Sensor Gas MQ-135

Pada perancangan sensor gas CO_2 dilakukan dengan menggunakan sensor berjenis MQ-135. Sensor MQ-135 ini dihubungkan dengan Arduino Uno. Ada 4 pin yang ada pada sensor MQ-135, yaitu pin VCC, Ground, Digital Output, dan Analog Output. Pin yang digunakan hanya tiga, yaitu pin suplai (VCC dan Ground) dan pin untuk membaca nilai tegangan sensor yang bisa dipilih antara pin analog output atau digital output. Pada perancangan ini digunakan pin analog output. Tegangan yang keluar dari analog output akan dibaca dengan pin analog input Arduino Uno, sedangkan untuk suplai sensor bisa didapat dari tegangan

suplai 5 Volt. Untuk hubungan anatara sensor dan Arduino Uno dapat dilihat pada Tabel 3.2. Untuk Ilustrasi gambar hubungan antara sensor MQ-135 dengan Arduino Uno dapat dilihat pada gambar 3.4.

Tabel 3.2 *Wiring* pin antara pin Arduino dan sensor MQ-135

Arduino Uno	Sensor MQ-02
Pin 5 Volt	Pin VCC
Pin Ground	Pin Ground
Pin Analog A1	Pin Analog Output (AOUT)

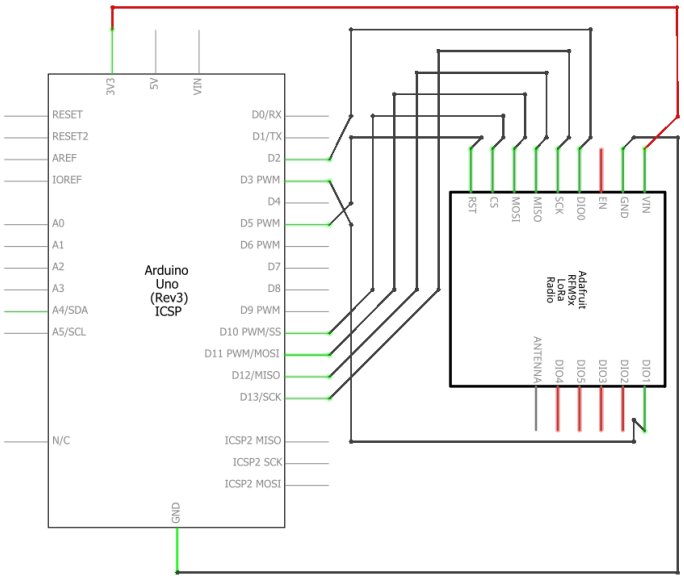


Gambar 3.4 *Wiring* diagram sensor gas CO_2 MQ-135 dengan Arduino

3.3.3 LoRa RF96

Pada perancangan perangkat keras RF96 dilakukan dengan menghubungkan beberapa pin yang ada pada RF96. Pin yang dipakai adalah pin 3.3 Volt, pin Ground, pin MISO, pin MOSI, Pin SCK, Pin SS, pin RESET, pin DIO0, dan pin DIO1. Pin 3.3 Volt di hubungkan dengan pin 3.3 Volt yang ada pada Arduino, pin Ground pada RF96 dihubungkan

dengan pin Ground Arduino. Pin MISO, MOSI, SCK dan SS dihubungkan dengan Pin 11, 12, 13 dan 10 pada Arduino. Pin RESET dihubungkan dengan pin 5, sedangkan pin DIO0 dan DIO1 dihubungkan dengan pin 2 dan pin 3 pada Arduino. Untuk hubungan antara LoRa dan Arduino dapat dilihat pada tabel 3.3. Untuk ilustrasi gambar hubungan standar antara RF96 dengan Arduino Uno dapat dilihat pada gambar 3.5.



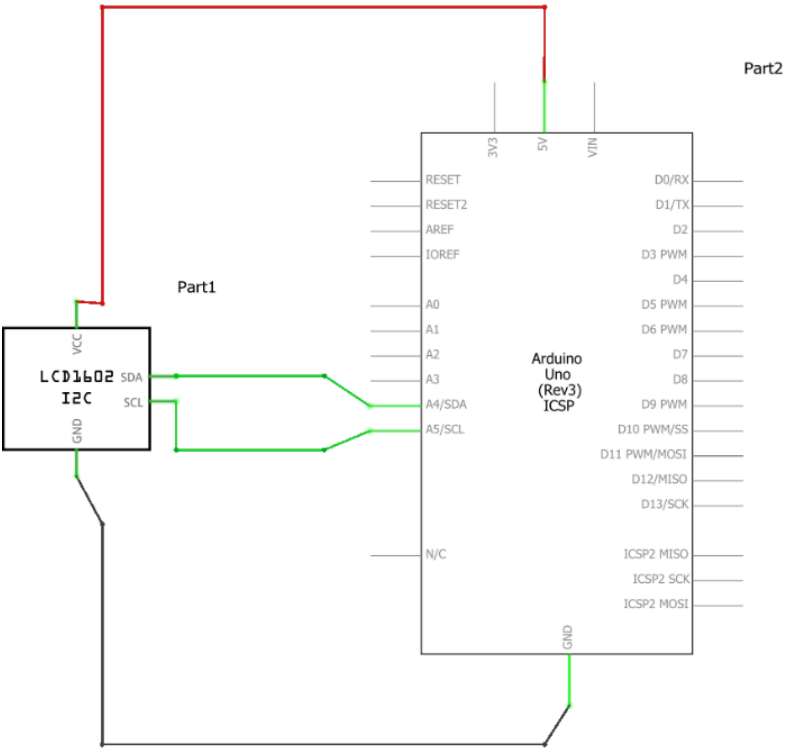
Gambar 3.5 *Wiring diagram* RF96 ke Arduino Uno

Tabel 3.3 *Wiring* pin antara pin Arduino dan RF96

Arduino Uno	RF96
Pin 3.3 Volt	Pin 3.3 Volt
Pin Ground	Pin Ground
Pin Digital 5 (5)	Pin Reset
Pin NSS (10)	Pin NSS
Pin SCK (13)	Pin SCK
Pin MISO (12)	Pin MISO
Pin MOSI (11)	Pin MOSI
Pin Digital 2 (2)	Pin DIO0
Pin Digital 3 (3)	Pin DIO1

3.3.4 LCD 16X2 I2C

Perancangan LCD pada Arduino dilakukan dengan menggabungkan LCD 16X2 dengan modul LCD I2C. Modul LCD I2C sendiri memiliki 4 pin yaitu VCC, Ground, SDA, dan SCL. Tujuan digunakannya modul LCD I2C adalah agar tidak menghabiskan banyak pin, cukup memakai 4 pin. Jika tidak memakai modul LCD I2C pengkoneksian antara LCD dan Arduino membutuhkan 16 pin dengan konfigurasi jalur yang tidak efisien. 4 pin yang digunakan duhubungkan pada pin Arduin Uno dengan konfigurasi dapat dilihat pada tabel dan gambar 3.3 untuk visualisai kabel. Untuk hubungan antara LCD I2C dan Arduino dapat dilihat pada tabel 3.4. Untuk ilustrasi hubungan antara LCD I2C dengan Arduino uno dapat dilihat pada gambar 3.6.



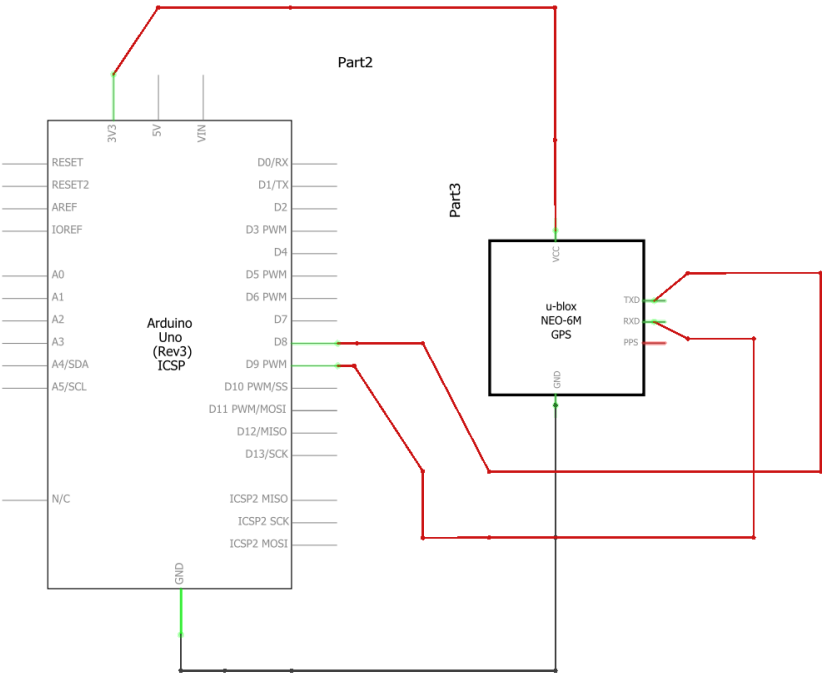
Gambar 3.6 *Wiring diagram* LCD 16X2 I2C dengan Arduino Uno

Tabel 3.4 *Wiring* pin antara pin Arduino dengan LCD I2C

Arduino Uno	LCD I2C
5 Volt	VCC
Ground	Ground
SDA	SDA
SCL	SCL

3.3.5 GPS u-Blox NEO-6M

Perancangan komunikasi antara NEO-6M dan Arduino dilakukan dengan menggunakan model komunikasi serial. Pin yang digunakan untuk berkomunikasi serial, tidak langsung memakai pin serial yang ada pada Arduino Uno. Tetapi menggunakan pin lain, lebih tepatnya menggunakan pin digital yang ada pada *microcontroller*.



Gambar 3.7 *Wiring diagram* u-Blox NEO-6M dengan Arduino Uno

Karena ada pin (pin digital) yang pada masing-masing *microcontroller* yang bisa difungsikan sebagai pin serial atau lebih tepatnya sebagai pin RX dan TX. Pengunaan pin ini harus diatur menjadi pin RX dan TX pada program Arduino dengan menggunakan *library Software Serial*. Pada perancangan ini digunakan pin Digital 8 sebagai pin RX dan pin Digital 9 sebagai pin TX. Hubungan kabel RX dan TX antara Arduino Uno dengan NEO-6M harus disilang, pin RX pada NEO-6M dihubungkan dengan pin TX pada Arduino Uno, begitu juga sebaliknya. Untuk hubungan antara GPS dan Arduino dapat dilihat pada tabel 3.5. Untuk ilustrasi hubungan antara GPS dengan Arduino Uno dapat dilihat pada gambar 3.7.

Tabel 3.5 *Wiring* pin antara pin Arduino dengan u-Blox NEO-6M

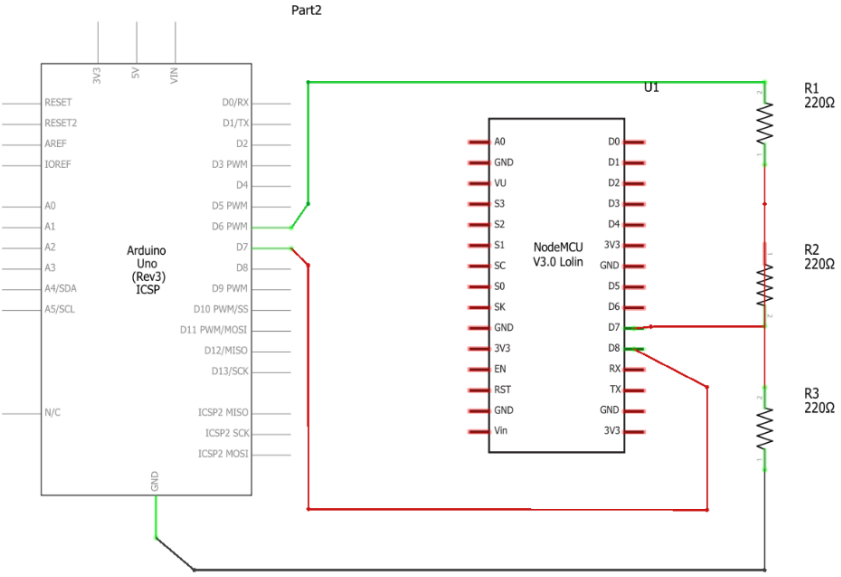
Arduino Uno	u-Blox NEO-6M
3.3 Volt	VCC
Ground	GND
Pin Digital 9 (TX)	RXD
Pin Digital 8 (RX)	TXD

3.3.6 NodeMCU

Perancangan komunikasi antara NodeMCU dan Arduino dilakukan dengan menggunakan model komunikasi Serial. Untuk pin yang digunakan untuk bekomunikasi serial, tidak langsung memamkai pin serial yang ada pada Arduino Uno maupun yang ada pada NodeMCU. Tetapi menggunakan pin lain, lebih tepatnya menggunakan pin digital yang ada pada masing-masing microcontroller. Karena ada pin (pin digital) yang pada masingmasing microcontroller yang bisa difungsikan sebagai pin serial atau lebih tepatnya sebagai pin RX dan TX. Penggunaan pin ini harus diatur menjadi pin RX dan TX pada program Arduino dengan menggunakan library Software Serial. Pada perancangan ini digunakan pin Digital 6 sebagai pin TX dan pin Digital & sebagai pin RX. Sedangkan pada NodeMCU digunakan pin D7 sebagai pin RX dan pin D8 sebagai pin TX. Hubungan kabel TX dan RX antara Arduino Uno dengan NodeMCU harus disilang, pin TX pada Arduino Uno dihubunkan dengan pin RX pada NodeMCU, begitu juga sebalikanya. Karena input dan output dari NodeMCU adalah tegangan 3.3 Volt dan input dan output Arduino Uno adalah 5 Volt, ketika hubungan antara pin TX NodeMCU dengan pin RX Arduino Un bisa langsung dihubungkan. Sedangkan Jika

pin TX yang ada pada arduino dihubungkan dengan pin RX NodeMCU harus Menggunakan Pembagian rangkaian pembagi tegangan agar yang masuk pada NodeMCU adalah 3.3 Volt, karena keluaran dari pin TX Arduino Uno adalah 5 Volt. Rangkaina pembagi tegangan dapat digunakan rangkaian sederhana yaitu dengang mengunakan 3 buah resistor bernilai 1 KOhm yang dihubungkan seri, untuk rumus dapat dilihat pada rumus 3.1. Untuk visualisasi gambar dapat dilihat pada gambar 3.8. Untuk hubungan antara NodeMCU dan Arduino dapat dilihat pada tabel 3.6.

$$V_{out} = \frac{2}{3} V_{Tx} \tag{3.1}$$



Gambar 3.8 *Wiring* diagram NodeMCU to Arduino Uno

Tabel 3.6 *Wiring* pin antara Arduino Uno dengan NodeMCU

Arduino Uno	NodeMCU
Pin 6 (TX)	Pin D7 (RX)
Pin 7 (RX)	Pin D8 (TX)

3.4 Perancangan Perangkat Lunak

3.4.1 Pembacaan Sensor MQ-02

Perancangan perangkat lunak pada sensor MQ-02 dimulai dengan pembacaan tegangan ADC yang masuk pada Arduino Uno dengan menggunakan program sebagai berikut:

```
float MQResistanceCalculation (int raw_adc, float rl_value)
{
 return
(long)((long)  (1024  *  (long)rl_value)  /  raw_adc  -
(long)rl_value);
}
```

rl_value adalah nilai rl yang terukur pada sensor, sedangkan raw_adc adalah tegangan yang keluar dari analog *output* sensor dan dibaca oleh Arduino Uno. Selanjutnya akan dicari nilai rs dengan menggunakan program sebagai berikut:

```
float MQRead(int mq_pin, float rl_value)
{
 int i;
 float rs = 0;

 for (i = 0; i < Read_Sample_Times; i++) {
   rs    +=    MQResistanceCalculation(analogRead(mq_pin),
   rl_value);
   delay(Read_Sample_Interval);
}

 rs = rs / Read_Sample_Times;

 return rs;
}
```

Read_Sample_Times merupakan banyaknya sample yang akan diambil. Fungsi analogRead() berfungsi untuk mengambil data analog

yang terbaca. Sedangkan Read_Sample_Interval merupakan delay waktu pengambilan satu sample data. Setelah didapat rs, selanjutnya mencari ro. Dalam mencari ro dapat digunakan program sebagai berikut:

```
float MQCalibration(int mq_pin, double ppm, double rl_value,
float *pcurve )
{
 int i;
 float val = 0;

 for (i = 0; i < Calibration_Sample_Times; i++) {     //take
 multiple samples

 val     +=     MQResistanceCalculation(analogRead(mq_pin),
 rl_value);
 delay(Calibration_Sample_Interval);
}
val  =  val  /  Calibration_Sample_Times;  //calculate  the
average value
//Ro = Rs * sqrt(a/ppm, b) = Rs * exp( ln(a/ppm) / b )

return  //val;
(float)val * exp((log(pcurve[0] / ppm) / pcurve[1]));

}
```

Didalam program MQCalibration juga memakai program MQResistanceCalculation yang telah dijelaskan diatas. Nilai PPM merupakan nilai PPM gas yang ada pada udara bersih. Calibration_Sample_Times merupakan pengambilan multiple sample. Sedangkan Calibration_Sample_Interval adalah lamanya jeda pengambilan nilai val. Val sendiri merupakan nilai yang dihasilkan dari program MQCalibration. Val disini juga bisa di definisikan sebagai ro. Setelah didapat ro selanjutnya akan dicari nilai persentasi konsentrasi gas. Dalam mencari persentase gas dapat digunakan program sebagai berikut:

```
float MQGetGasPercentage(float rs_ro_ratio, float ro, int
gas_id, int sensor_id)
{
 if (sensor_id == MQ02) {
 if ( gas_id == GAS_CH4 ) {
 return  MQGetPercentage(rs_ro_ratio,     ro,  CH4_Curve);
 //MQ02
} else if ( gas_id == GAS_LPG ) {
 return  MQGetPercentage(rs_ro_ratio,     ro,  LPG_Curve);
```

```
 //MQ02
} else if ( gas_id == GAS_C3H8 ) {
 return MQGetPercentage(rs_ro_ratio,   ro, C3H8_terCurve);
 //MQ02
} else if ( gas_id == GAS_H2 ) {
 return MQGetPercentage(rs_ro_ratio, ro, H2_Curve); //MQ02
} else if ( gas_id == GAS_CO ) {
  return  MQGetPercentage(rs_ro_ratio,  ro,  CO_terCurve);
  //MQ02
} else if ( gas_id == GAS_C2H5OH ) {
  return  MQGetPercentage(rs_ro_ratio,  ro,  C2H5OH_Curve);
  //MQ02
 }
}
  return 0;
}
```

Karena dalam satu sensor dapat membaca beberapa gas maka digunakan fungsi if untuk memilih gas mana yang akan dideteksi. Fungsi diatas merupakan fungsi MQGetPercentage dengan isi rasio rs terhadap ro, nilai dan kurva gas yang dibaca, nilainya didapat dari kurva karakteristik yang ada pada datasheet. Cara mencarinya dapat menggunakan plot nilai PPM dan nilai perbandingan rs banding ro. Fungsi diatas hanya menginisialisasi dan memisahkan gas apa yang dibaca. Setelah tahu gas apa yang dibaca, maka akan masuk ke program berikut untuk mendapatkan nilai PPM:

```
float  MQGetPercentage(float rs_ro_ratio, float ro, float
*pcurve)
{
 return (float)(pcurve[0] * pow(((float)rs_ro_ratio / ro),
 pcurve[1]));
}
```

Didalam program diatas terdapat fungsi pow yang digunakan untuk meringkas perkalian suatu bilangan sebanyak pangkat bilangan tesebut. Dalam fungsi diatas berarti perbandingan rs banding ro dibagi dengan ro pangkat nilai curva pada array kedua. Setelah didapatkan hasilnya akan dikalikan dengan nilai kurva yang pertama.

3.4.2 Pembacaan Sensor MQ-135

Perancangan perangkat lunak pada sensor MQ-135 hampir sama dengan perancangan sensor MQ-02, dimulai dengan pembacaan tegangan

ADC yang masuk pada Arduino Uno dengan menggunakan program sebagai berikut:

```
float MQResistanceCalculation (int raw_adc, float rl_value)
{
 return
(long)((long)  (1024  *  (long)rl_value)  /  raw_adc  -
(long)rl_value);
}
```

rl_value adalah nilai rl yang terukur pada sensor, sedangkan raw_adc merupakan tegangan yang keluar dari analog *output* sensor dan dibaca oleh Arduino Uno. Selanjutnya dicari nilai rs dengan menggunakan program sebagai berikut:

```
float MQRead(int mq_pin, float rl_value)
{
 int i;
 float rs = 0;

 for (i = 0; i < Read_Sample_Times; i++) {
   rs    +=    MQResistanceCalculation(analogRead(mq_pin),
   rl_value);
   delay(Read_Sample_Interval);
}

 rs = rs / Read_Sample_Times;

 return rs;
}
```

Read_Sample_Times merupakan banyaknya sample yang akan diambil. Fungsi analogRead() berfungsi mengambil data analog yang terbaca. Sedangkan Read_Sample_Interval merupakan delay waktu pengambilan satu sample data. Setelah didapat rs, selanjutnya mencari ro, dalam mencari ro dapat menggunakan program sebagai berikut:

```
float MQCalibration(int mq_pin, double ppm, double rl_value,
float *pcurve )
{
 int i;
 float val = 0;

 for (i = 0; i < Calibration_Sample_Times; i++) {
```

```
 val     +=     MQResistanceCalculation(analogRead(mq_pin),
 rl_value);
 delay(Calibration_Sample_Interval);
}
val = val / Calibration_Sample_Times;

return  //val;
(float)val * exp((log(pcurve[0] / ppm) / pcurve[1]));

}
```

Didalam program MQCalibration juga memakai program MQResistanceCalculation yang telah dijelaskan diatas. Nilai PPM merupakan nilai PPM gas yang ada pada udara bersih. Calibration_Sample_Times merupakan pengambilan multiple sample. Sedangkan Calibration_Sample_Interval adalah lamanya jeda pengambilan nilai val. Val sendiri merupakan nilai yang dihasilkan dari program MQCalibration. Val disini juga bisa di definisikan sebagai ro. Setelah didapat ro selanjutnya akan dicari nilai persentasi konsentrasi gas. Dalam mencari persentase gas dapat digunakan program sebagai berikut:

```
float MQGetGasPercentage(float rs_ro_ratio, float ro, int
gas_id, int sensor_id)
{
 if (sensor_id == MQ135) {
  if ( gas_id == GAS_CO2 ) {
      return MQGetPercentage(rs_ro_ratio, ro, CO2_Curve);
    } else if ( gas_id == GAS_C2H5OH ) {
return MQGetPercentage(rs_ro_ratio,ro, C2H5OH_Curve);
  } else if ( gas_id == GAS_CH3 ) {
    return MQGetPercentage(rs_ro_ratio, ro,CH3_Curve);
  } else if ( gas_id == GAS_CH3_2CO ) {
    returnMQGetPercentage(rs_ro_ratio, ro,CH3_2CO_Curve);
  } else if ( gas_id == GAS_CO ) {
    return MQGetPercentage(rs_ro_ratio,  ro,CO_Curve);
  } else if ( gas_id == GAS_NH4 ) {
    return MQGetPercentage(rs_ro_ratio,  ro,NH4_Curve);
  }
 }
  return 0;
}
```

Karena dalam satu sensor dapat membaca beberapa gas maka digunakan fungsi if untuk memilih gas mana yang akan dideteksi. Fungsi diatas merupakan fungsi MQGetPercentage dengan isi rasio rs terhadap

ro, nilai dan kurva gas yang dibaca, nilainya didapat dari kurva karakteristik yang ada pada datasheet. Cara mencarinya dapat menggunakan plot nilai PPM dan nilai perbandingan rs banding ro. Fungsi diatas hanya menginisialisasi dan memisahkan gas apa yang dibaca. Setelah tahu gas apa yang dibaca, maka akan masuk ke program berikut untuk mendapatkan nilai PPM:

```
Float MQGetPercentage(float rs_ro_ratio, float ro, float
*pcurve)
{
 return (float)(pcurve[0] * pow(((float)rs_ro_ratio / ro),
 pcurve[1]));
}
```

Didalam program diatas terdapat fungsi pow yang digunakan untuk meringkas perkalian suatu bilangan sebanyak pangkat bilangan tesebut. Dalam fungsi diatas berarti perbandingan rs banding ro dibagi dengan ro pangkat nilai curva pada array kedua. Setelah didapatkan hasilnya akan dikalikan dengan nilai kurva yang pertama.

3.4.3 Pembacaan GPS u-Blox NEO-6M

Perancangan perangkat lunak pada GPS u-Blox NEO-6M dimulai dengan program untuk mendeteksi GPS dengan Arduino Uno.

```
while (ss.available() > 0)
 if (gps.encode(ss.read()))
   data_sender();

if (millis() > 5000 && gps.charsProcessed() < 10)
{
 Serial.println(F("No GPS detected: check wiring."));
 while(true);
}
```

Didalam program diatas terdapat perintah untuk memeriksa kembali *wiring* GPS dengan *microcontroller* jika GPS tidak terdeteksi. Selanjutnya ketika GPS sudah terdeteksi, GPS akan mengambil data posisi.

```
char gps_lon[20]={"\0"};
char gps_lat[20]={"\0"};
if(gps.location.isValid())
{
```

```
 dtostrf(gps.location.lat(), 0, 6, gps_lat);
 dtostrf(gps.location.lng(), 0, 6, gps_lon);
}
```

Data posisi yang didapatkan oleh GPS adalah berupa *latitude* dan *longitude* yang kemudian diubah ke dalam bentuk string agar siap untuk dikirim melalui LoRa.

3.4.4 Pengiriman Data dengan LoRa

Pada perancangan perangkat lunak untuk pengiriman data dengan LoRa diawali dengan melakukan inisialisasi dengan mengatur *baudrate* dan inisialisasi *library* LoRa menggunakan fungsi rf95.setFrequency (frekuensi yang digunakan). Pengaturan LoRa dijelaskan dalam program berikut:

```
Serial.begin(9600);
ss.begin(9600);

while (!Serial);
if (!rf95.init())
  Serial.println("init failed");
rf95.setFrequency(Frequency);
rf95.setTxPower(13);
```

Sebelum data dikirim dengan LoRa, data yang bertipe *Float* diubah menjadi tipe *String*. Setelah tipe data yang akan dikirim dirubah, kemudian data digabung menjadi satu, karena data yang akan dikirim memiliki banyak variabel, sehingga akan lebih mudah jika digabung dan dikirim satu kali saja. Untuk mengubah tipe data *Float* menjadi tipe *String* digunakan fungsi dtostrf(). Kemudian setelah tipe data dirubah, digabungkan menggunakan fungsi strcat() dan strcpy(). Untuk lebih jelasnya dapat dilihat pada program berikut:

```
int data_sender() {
CH4 = MQGetGasPercentage(MQRead(MQ02, RL0), Ro0, GAS_CH4,
MQ02);

CO2 = MQGetGasPercentage(MQRead(MQ135, RL1), Ro1, GAS_CO2,
MQ135);

char gps_lon[20]={"\0"};
char gps_lat[20]={"\0"};
if(gps.location.isValid())
  {
```

```
   dtostrf(gps.location.lat(), 0, 6, gps_lat);
   dtostrf(gps.location.lng(), 0, 6, gps_lon);
  }

char data_1[10] = {"\0"};
char data_2[10] = {"\0"};
char data_sensor[100] = {"\0"};
dtostrf(CH4, 5, 2, data_1);
dtostrf(CO2, 5, 2, data_2);

strcat(data_sensor, data_1);
strcat(data_sensor, "\n");
strcat(data_sensor, data_2);
strcat(data_sensor, "\n");
strcat(data_sensor, gps_lat);
strcat(data_sensor, "\n");
strcat(data_sensor, gps_lon);
strcat(data_sensor, "\n");

strcpy((char*)data, data_sensor);
Serial.println((char*)data);
}
```

Setelah data diolah, data akan dikirim dengan *library* LoRa. Dimana sudah terdapat fungsi untuk memaketkan data dan sekaligus mengirimnya. Berikut program yang diberikan:

```
rf95.send(data, sizeof(data));
rf95.waitPacketSent();
```

Program rf95.send(data, sizeof(data)) merupakan fungsi yang digunakan untuk mengirim data. Didalam fungsi tersebut, diisi dengan data yang akan dikirim serta ukuran data yang akan dikirim. Sedangkan program rf95.waitPacketSent() merupakan fungsi yang digunakan untuk menunggu data terkirim ke penerima.

3.4.5 Penerimaan Data dengan LoRa

Pada perancangan perangkat lunak untuk penerimaan data menggunakan LoRa, diawali dengan melakukan inisialisasi dengan mengatur *baudrate* dan inisialisasi *library* LoRa dengan fungsi rf95.setFrequency(frekuensi yang digunakan). Untuk lebih jelasnya dapat dilihat pada program berikut:

```
Serial.begin(9600);
```

```
while (!Serial) ; // Wait for serial port to be available
if (!rf95.init())
  Serial.println("init failed");
  rf95.setFrequency(frequency);
  rf95.setTxPower(13);
```

Setelah inisialisasi, banyaknya paket yang akan diterima harus ditentukan. Ukuran paket yang dikirim harus sama dengan ukuran paket yang akan diterima. Untuk menentukan ukuran paket tersebut dapat menggunakan program sebagai berikut:

```
uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
```

Setelah ukuran paket ditentukan, maka dilanjutkan dengan penerimaan data yang akan disimpan ke dalam variabel data menggunakan program sebagai berikut:

```
(rf95.recv(buf, &len);
```

3.4.6 Pengiriman Data ke NodeMCU

Perancangan perangkat lunak untuk pengiriman data ke NodeMCU dari Arduino Uno dimulai dengan melakukan inisialisasi. Menggunakan *library* SoftwareSerial karena pengiriman tidak menggunakan pin RX dan TX dari Arduino Uno. Setelah itu harus menentukan data awal, data akhir, dan daya yang akan dikirim, juga menentukan *baudrate* pada SoftwareSerial seperti pada program berikut:

```
SoftwareSerial kirimSerial(7,6);
char nilai[5][10];
char line = '\n';
const byte numChars = 50;
char start = '<';
char stopped = '>';

kirimSerial.begin(9600);
```

Setelah itu daya yang akan dikirim dengan suatu fungsi yang ada pada *library* SoftwareSerial. Fungsi yang dipakai adalah fungsi name_Serial.write(). Data yang dikirim adalah data start, data yang diterima dari LoRa yang disimpan pada variabel data, dan yang terakhir adalah data stopped. Untuk lebih jelasnya bisa dilihat pada program

berikut:

```
kirimSerial.write(start);
kirimSerial.write(dataTerima,sizeof(dataTerima));
kirimSerial.write(stopped);
```

3.4.7 Pengolahan Data di NodeMCU

Setelah data di terima di NodeMCU, dua data sensor beserta data dari GPS masih menjadi satu, yang kemudian dipisah menjadi 3. Program yang digunakan untuk memisah data seperti berikut:

```
for(i = 0; i <= numChars ; i++){

if(receivedChars[i] != line){
dataSensor[data_ke][index] = receivedChars[i];
index++;
}
else
{
  data_ke++;
  index = 0;
 }
}
```

3.4.8 Pengiriman Data ke Aplikasi Blynk

Setelah data dipisah kemudian untuk mengirim data sensor dan label data ke peta yang ada di Blynk. Data dan label yang berupa string digabung menjadi satu dan diberi pemisah berupa karakter spasi. Fungsi yang digunakan seperti fungsi strcat dan strcpy. Untuk lebih jelasnya dapat dilihat pada program berikut:

```
char dataSenMap[50] = {"\0"};
strcat(dataSenMap, "CH4 =");
strcat(dataSenMap, dataSensor[0]);
strcat(dataSenMap, "  ");
strcat(dataSenMap, "CO2 =");
strcat(dataSenMap, dataSensor[1]);
strcat(dataSenMap, "  ");
strcat(dataSenMap, "lat =");
strcat(dataSenMap, dataSensor[2]);
strcat(dataSenMap, "  ");
strcat(dataSenMap, "lon =");
strcat(dataSenMap, dataSensor[3]);
strcat(dataSenMap, "  ");
```

```
strcpy(dataMap, dataSenMap);
```

Setelah data dipisah kemudian dilakukan pengiriman data ke aplikasi Blynk. Fungsi yang dipakai adalah Blynk.virtualWrite(). Pada fungsi ini berisi pin yang akan digunakan pada aplikasi Blynk yang kita tentukan sendiri dan data yang akan dikirim. Sedangkan untuk menampilkan data ke peta isi dari fungsi tersebut terdapat latitude dan longitude berupa koordinat yang kita berikan.

```
Blynk.virtualWrite(V1, dataSensor[0]);
Blynk.virtualWrite(V2, dataSensor[1]);
Blynk.virtualWrite(V4,       index_1,       dataSensor[2],
dataSensor[3], dataMap);
```

Untuk menghubungkan NodeMCU ke aplikasi Blynk pertama harus diinisiasi menggunakan fungsi Blynk.begin(). Fungsi ini berisi auth, ssid, pass. Auth merupakan nomer autentifikasi yang didapat dari aplikasi Blynk, yang berfungsi sebagai pengenal dan pembeda dari cloud yang dimiliki Blynk. Ssid merupakan nama WiFi atau jaringan yang digunakan untuk menghubungkan NodeMCU dan aplikasi Blynk. Pass merupakan password dari WiFi atau jaringan yang digunakan. Dan untuk menjalankan aplikasi, digunakan fungsi Blynk.run().

```
Blynk.begin(auth, ssid, pass);
Blynk.run();
```

---Halaman ini sengaja dikosongkan---

BAB IV
PENGUJIAN DAN ANALISIS

4.1 Pengujian Sensor Gas

Pengujian sensor gas, yaitu sensor gas MQ-02 maupun MQ-135 yang sudah terpasang pada *board* mikrokontroler dilakukan secara bersama-sama, yang bertempat di Gedung B lantai 4 Teknik Elektro ITS. Pengujian ini dilakukan untuk mengetahui apakah sensor gas MQ-02 dan MQ-135 yang sudah terpasang di *board* dapat bekerja dan terkalibrasi dengan baik. Pengujian dilakukan dengan empat kondisi, yaitu kondisi pertama merupakan kondisi ketika udara bersih atau bisa dibilang udara bebas, kondisi kedua ketika ada asap rokok, kondisi ketiga ketika ada asap kertas, dan kondisi keempat ketika ada gas metana.

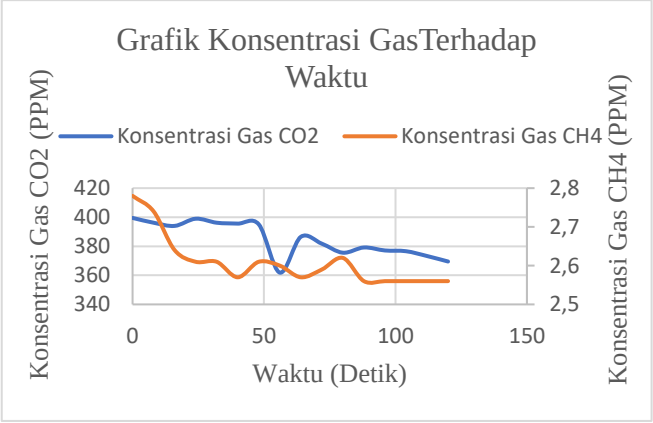
4.1.1 Pengujian Sensor Terhadap Udara Bersih

Pada pengujian ini dilakukan dengan udara yang ada pada sekitar lokasi uji yaitu Gedung B lantai 4. Pengujian dilakukan dengan tidak memperhitungkan adanya polusi yang ada pada lokasi uji. Sehingga data yang didapat cukup bervariasi. Untuk hasil pengujian di udara bersih dapat dilihat pada tabel 4.1. Grafik dari hasil pengujian dapat dilihat pada gambar 4.1.

Tabel 4.1 Pengujian Sensor Terhadap Udara Bersih

Waktu (s)	PPM Terbaca	
	MQ-02	MQ-135
0	2,78	399,51
8	2,74	396,20
16	2,64	394,02
24	2,61	398,96
32	2,62	396,20
40	2,57	395,66
48	2,61	395,11
56	2,60	361,85
64	2,57	386,50
72	2,59	381,77
80	2,62	375,57
88	2,56	379,17
96	2,56	377,11

Waktu (s)	PPM Terbaca	
	MQ-02	MQ-135
104	2,56	376,60
112	2,56	373,03
120	2,56	369,51



Gambar 4.1 Grafik konsentrasi gas terhadap waktu di udara bersih

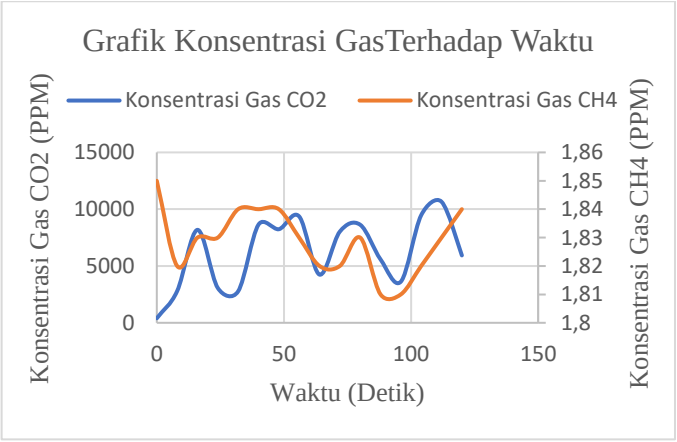
4.1.2 Pengujian Sensor Terhadap Asap Rokok

Pada pengujian ini dilakukan dengan membakar rokok hingga menghasilkan asap. Pengujian ini tidak memperhitungkan kondisi yang dapat mengganggu pengujian, seperti angin, jarak antara asap dengan sensor, dan asap yang dikeluarkan oleh rokok juga bisa tidak stabil. Karena faktor tersebut data yang terbaca berubah-ubah tidak mengikuti perubahan volume asap. Untuk hasil pengujian terhadap asap rokok dapat dilihat pada tabel 4.2. Grafik dari hasil pengujian dapat dilihat pada gambar 4.2.

Tabel 4.2 Pengujian Sensor Terhadap Asap Rokok

Waktu (s)	PPM Terbaca	
	MQ-02	MQ-135
0	1,85	397,90
8	1,82	278,26
16	1,83	818,23
24	1,83	3061,75

Waktu (s)	PPM Terbaca	
	MQ-02	MQ-135
32	1,84	2773,59
40	1,84	8641,89
48	1,84	8273,45
56	1,83	9361,80
64	1,82	4243,80
72	1,82	8027,94
80	1,83	8631,76
88	1,81	5583,27
96	1,81	3622,66
104	1,82	9473,39
112	1,83	10658,01
120	1,84	5930,87



Gambar 4.2 Grafik konsentrasi gas terhadap waktu di asap rokok

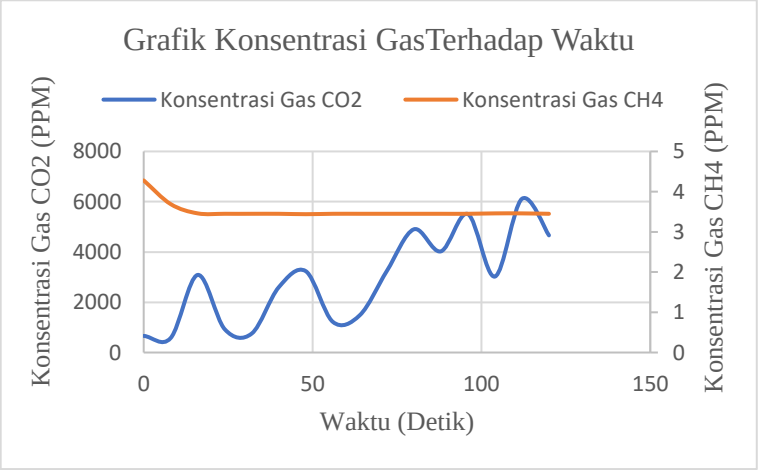
4.1.3 Pengujian Sensor Terhadap Asap Kertas

Pada pengujian ini dilakukan dengan membakar kertas hingga menghasilkan asap. Pengujian ini tidak memperhitungkan kondisi yang dapat mengganggu pengujian, seperti angin, jarak antara asap dengan sensor, dan asap yang dikeluarkan oleh kertas juga bisa tidak stabil. Karena faktor tersebut data yang terbaca berubah-ubah tidak mengikuti perubahan volume asap. Untuk hasil pengujian terhadap asap rokok dapat

dilihat pada tabel 4.3. Grafik dari hasil pengujian dapat dilihat pada gambar 4.3.

Tabel 4.3 Pengujian Sensor Terhadap Asap Kertas

Waktu (s)	PPM Terbaca	
	MQ-02	MQ-135
0	4,28	662,79
8	3,69	578,34
16	3,46	3087,70
24	3,45	926,01
32	3,45	755,36
40	3,45	2598,61
48	3,44	3235,56
56	3,45	1220,46
64	3,45	1486,26
72	3,45	3244,55
80	3,45	4902,44
88	3,45	4025,54
96	3,45	5509,73
104	3,46	3021,23
112	3,46	6115,51
120	3,45	4661,08



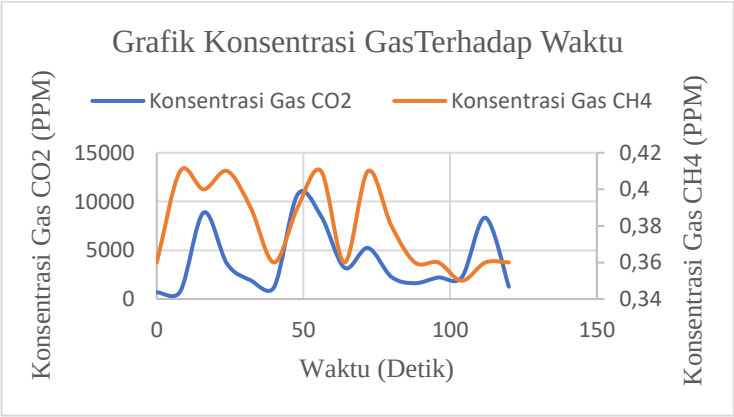
Gambar 4.3 Grafik konsentrasi gas terhadap waktu di asap kertas

4.1.4 Pengujian Sensor Terhadap Asap Batubara

Pada pengujian ini dilakukan dengan membakar batubara hingga menghasilkan asap.

Tabel 4.4 Pengujian Sensor Terhadap Asap Batubara

Waktu (s)	PPM Terbaca	
	MQ-02	MQ-135
0	0,36	691,50
8	0,41	769,93
16	0,40	8874,24
24	0,41	3593,90
32	0,39	1927,98
40	0,36	1224,45
48	0,39	10721,11
56	0,41	8514,63
64	0,36	3212,67
72	0,41	5243,50
80	0,38	2268,33
88	0,36	1625,07
96	0,36	2209,24
104	0,35	2238,53
112	0,36	8342,08
120	0,36	1267,79



Gambar 4.4 Grafik konsentrasi gas terhadap waktu di asap batubara

Pengujian ini tidak memperhitungkan kondisi yang dapat mengganggu pengujian, seperti angin, jarak antara asap dengan sensor, dan asap yang dikeluarkan oleh kertas juga bisa tidak stabil. Karena faktor tersebut data yang terbaca berubah-ubah tidak mengikuti perubahan volume asap. Untuk hasil pengujian terhadap asap rokok dapat dilihat pada tabel 4.4. Grafik dari hasil pengujian dapat dilihat pada gambar 4.4.

4.2 Pengujian Sensor per Volume

Pengujian ini dilakukan untuk mengetahui respon sensor terhadap gas yang diberikan dengan memberi sensor gas 1-10 ml gas yang akan diobservasi.

4.2.1 Pengujian Sensor MQ-135

Pengujian ini dilakukan untuk mengetahui respon sensor terhadap volume asap yang diberikan. Asap yang digunakan adalah asap yang dihasilkan dari pembakaran kertas. Metode pengujian ini dilakukan dengan membakar kertas dan memasukkan asapnya ke dalam botol. Untuk mengambil asap digunakan tabung suntik yang ada ukuran volumenya, dan memiliki satuan ukuran mililiter.

Pengujian dilakukan dengan memberikan sensor asap yang telah diambil dari botol melalui tabung suntik. Hasil yang didapatkan setiap kenaikan satu mililiter volume, nilai ppm yang terbaca juga naik.

Tabel 4.5 Pengujian Sensor Terhadap Asap Kertas

Volume (mL)	MQ-135 (PPM)
1	700,23
2	962,34
3	2786,35
4	3528,13
5	5374,05
6	5709,64
7	8479,75
8	8620,44
9	9866,23
10	11060,8

Gambar 4.5 Grafik hubungan antara kenaikan volume dan konsentrasi

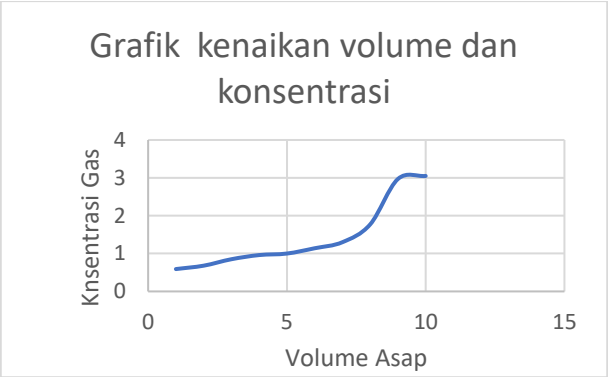
4.2.2 Pengujian Sensor MQ-02

Pengujian ini dilakukan untuk mengetahui respon sensor terhadap volume asap yang diberikan. Asap yang digunakan adalah asap yang dihasilkan dari pembakaran batubara. Metode pengujian ini dilakukan dengan membakar batubara. Untuk mengambil asap digunakan tabung suntik yang ada ukuran volumenya, dan memiliki satuan ukuran mililiter.

Pengujian dilakukan dengan memberikan sensor asap yang telah diambil dari botol melalui tabung suntik. Hasil yang didapatkan setiap kenaikan satu mililiter volume, nilai ppm yang terbaca juga naik.

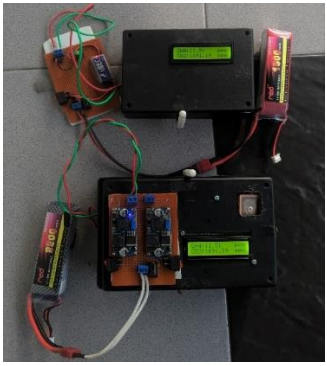
Tabel 4.6 Pengujian Sensor Terhadap Batubara

Volume (mL)	MQ-02 (PPM)
1	0,59
2	0,68
3	0,85
4	0,96
5	1,00
6	1,14
7	1,30
8	1,77
9	2,97
10	3,05



Gambar 4.6 Grafik hubungan antara kenaikan volume dan konsentrasi

4.3 Pengujian LoRa



Gambar 4.7 Sistem secara keseluruhan untuk pengujian

4.3.1 Pengujian LoRa dengan Posisi Receiver di K-Mart

Pengujian pengiriman data menggunakan LoRa dilakukan dengan meletakkan LoRa *Receiver* pada K-Mart, sedangkan *Transmitter* akan dibawa menggunakan motor yang tiap 100 meter akan berhenti dan dilihat apakah data yang dikirimkan dapat diterima oleh *Receiver* atau tidak, dan apakah data yang diterima nilainya sama dengan data yang dikirim. Jarak terjauh untuk *Receiver* menerima data dari *Transmitter* adalah 600 meter ketika pengujian. Ada beberapa faktor yang mempengaruhi, seperti interferensi objek yang menjadi penghalang, atau gangguan frekuensi dari

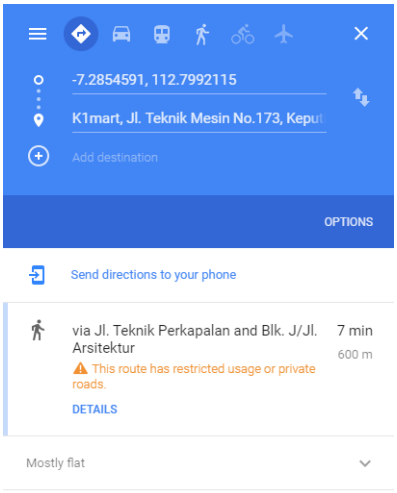
sinyal yang ada pada daerah perkotaan.

Tabel 4.7 Pengujian LoRa

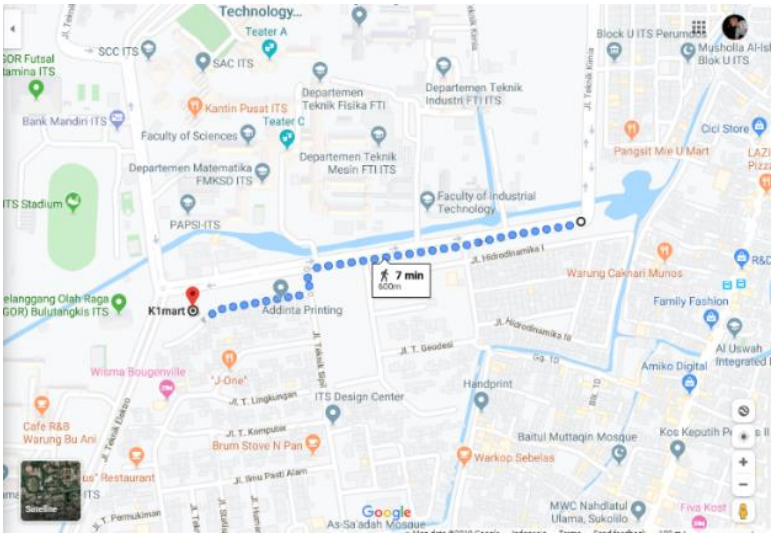
Jarak (m)	Data Terkirim		Data Diterima	
	MQ-02	MQ-135	MQ-02	MQ-135
100	0,91	361,64	0,91	361,64
200	0,91	360,18	0,91	360,18
300	0,91	309,91	0,91	309,91
400	0,91	297,6	0,91	297,6
500	0,91	348,38	0,91	348,38
600	0,91	321,66	0,91	321,66

4.3.2 Metode Pengukuran Jarak

Penentuan jarak dilakukan dengan menentukan posisi awal menggunakan aplikasi google map. Setelah menentukan posisi awal, pengguna membawa *Transmitter* sambil berjalan sampai *Receiver* tidak dapat menerima data lagi dari *Transmitter*. Ketika sampai di titik terakhir dimana *Receiver* sudah tidak dapat menerima data lagi, maka posisi terakhir itu dibuka di dalam google map, dan dicatat koordinatnya. Jarak akan didapatkan ketika 2 titik lokasi sudah didapatkan.



Gambar 4.8 Titik Lokasi yang Digunakan Untuk Mengukur Jarak



Gambar 4.9 Panjang Jarak Maksimal Komunikasi LoRa

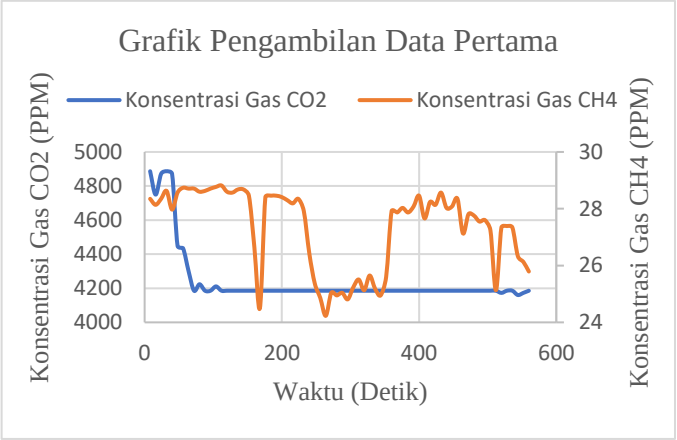
4.4 Pengujian Alat Secara Keseluruhan

Pengujian alat secara keseluruhan dilakukan di Kabupaten Samboja, Kalimantan Timur, dengan mengambil tempat pengujian di area pertambangan.

4.4.1 Pengambilan Data Pertama

Pengujian dilakukan dengan meletakkan *transmitter* di daerah yang terdapat lumayan banyak batubara. Sedangkan *receiver* diletakkan pada jarak sekitar 600 meter. Pengamatan dilakukan selama 560 detik. Hasil yang didapatkan adalah konsentrasi gas CH_4 berubah-ubah dan konsentrasi gas CO_2 relatif stabil. Konsentrasi gas CO_2 termasuk tinggi, dengan nilai tertinggi 4886,88 ppm dan nilai terendah 4160,48 ppm. Sementara konsentrasi gas CH_4 berubah-ubah dengan nilai tertinggi 28,82 ppm dan nilai terendah 24,23 ppm, namun konsentrasi gas CH_4 masih termasuk rendah. Nilai konsentrasi CO_2 yang relatif stabil disebabkan oleh dekatnya *transmitter* dengan lokasi batubara, sedangkan nilai konsentrasi gas CH_4 yang rendah dikarenakan kualitas batubara yang ada di lokasi tersebut termasuk buruk, atau bisa dikatakan tidak layak untuk diambil sebagai hasil tambang. Dengan jarak sejauh 600 meter *receiver* masih bisa menerima data yang dikirim oleh *transmitter* dengan baik. Respon yang didapatkan dari sensor cukup lambat dikarenakan prinsip

kerja sensor tersebut.

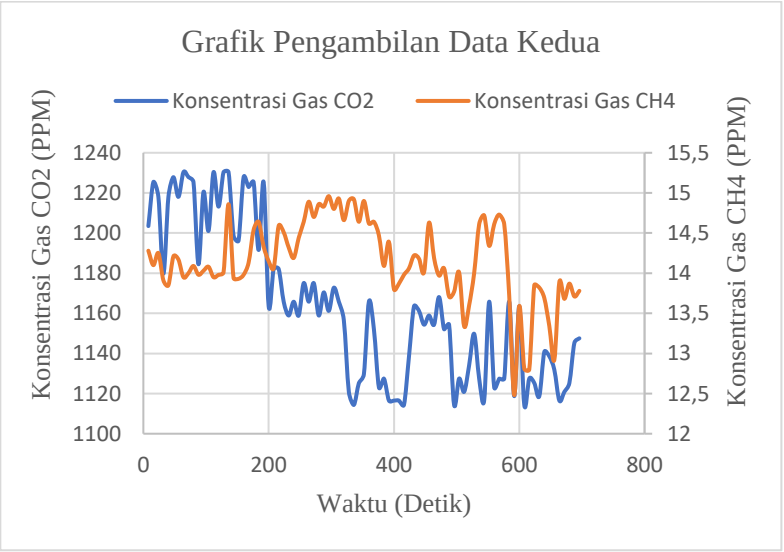


Gambar 4.10 Grafik pengambilan data pertama

Gambar 4.11 Pengambilan data pertama

4.4.2 Pengambilan Data Kedua

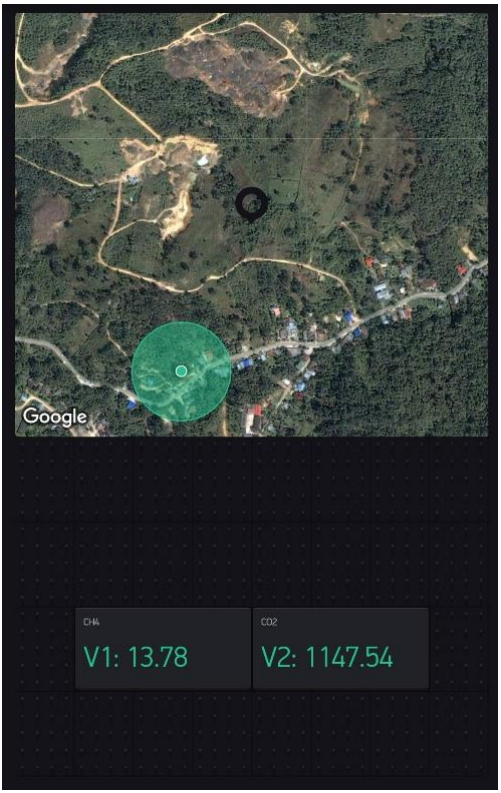
Pengambilan data kedua dilakukan di area tambang dengan lokasi *transmitter* berbeda dengan lokasi pengambilan data pertama. Pengujian dilakukan dengan menempatkan *transmitter* di area yang memiliki rata-rata batubara relatif sedikit. Sedangkan *receiver* diletakkan pada jarak sekitar 1 Km. Pengamatan dilakukan selama 696 detik. Hasil yang didapatkan adalah konsentrasi gas CH_4 dan CO_2 berubah-ubah namun perubahannya relatif rendah. Konsentrasi gas CH_4 termasuk rendah dengan nilai tertinggi sebesar 14,96 ppm dan nilai terendah sebesar 12,48 ppm. Konsentrasi gas CO_2 juga masih termasuk rendah dengan nilai tertinggi sebesar 1230,32 ppm dan nilai terendah sebesar 1134,34 ppm. Nilai konsentrasi CO_2 yang relatif rendah disebabkan oleh penempatan *transmitter* dengan lokasi yang minim batubara, sedangkan nilai konsentrasi gas CH_4 yang rendah dikarenakan kualitas batubara yang ada di lokasi tersebut termasuk buruk, atau bisa dikatakan tidak layak untuk diambil sebagai hasil tambang. Dengan jarak sejauh 1 Km *receiver* masih bisa menerima data yang dikirim oleh *transmitter* dengan baik. Respon yang didapatkan dari sensor cukup lambat dikarenakan prinsip kerja sensor tersebut.



Gambar 4.12 Grafik pengambilan data kedua

Gambar 4.13 Pengambilan data kedua

Tampilan dari aplikasi Blynk berisi nilai konsentrasi gas yang diobservasi oleh sensor gas CH_4 dan CO_2 beserta tampilan dari google maps yang menunjukkan lokasi dari *transmitter* juga lokasi dari *receiver* dapat kita lihat dari tampilan aplikasi. Untuk contoh visualisasi tampilan data yang ada pada aplikasi Blynk yang ada di *smartphone* kurang lebihnya dapat dilihat pada gambar 4.14.

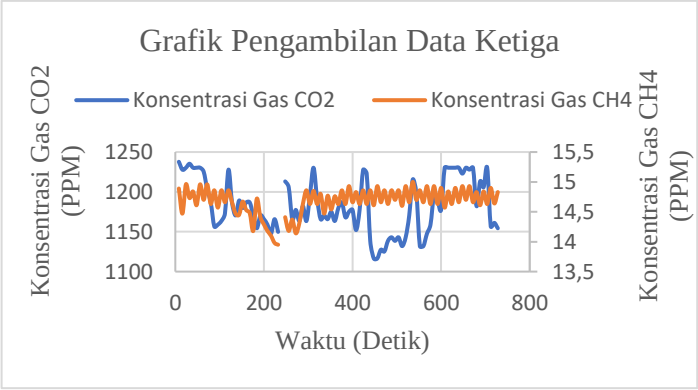


Gambar 4.14 Tampilan pengambilan data kedua dalam aplikasi Blynk

4.4.3 Pengambilan Data Ketiga

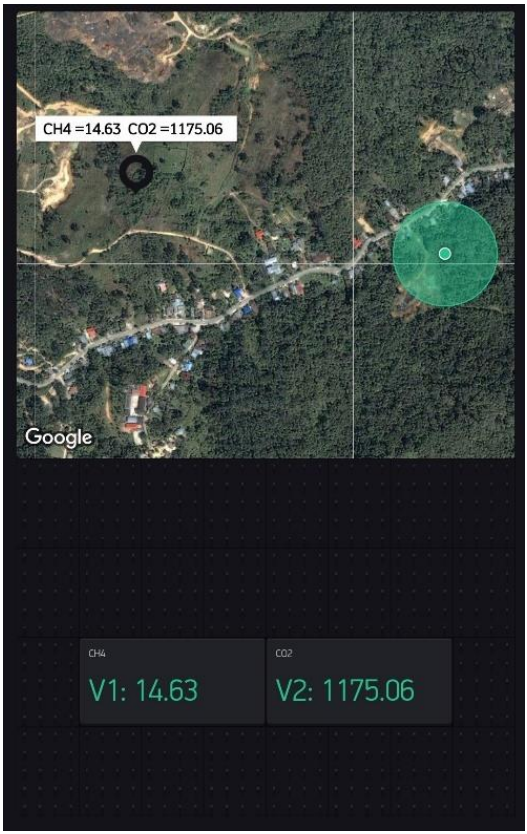
Pengambilan data ketiga dilakukan di area tambang dengan lokasi *transmitter* sama dengan lokasi pengambilan data kedua. Pengujian dilakukan dengan menempatkan *transmitter* di area yang memiliki rata-rata batubara relatif sedikit. Sedangkan *receiver* diletakkan pada jarak sekitar 1,6 Km. Pengamatan dilakukan selama 728 detik. Hasil yang didapatkan adalah konsentrasi gas CH_4 dan CO_2 berubah-ubah namun perubahannya relatif rendah. Konsentrasi gas CH_4 termasuk rendah dengan nilai tertinggi sebesar 14,96 ppm dan nilai terendah sebesar 13,95 ppm. Konsentrasi gas CO_2 juga masih termasuk rendah dengan nilai

tertinggi sebesar 1237,78 ppm dan nilai terendah sebesar 1116,51 ppm. Nilai konsentrasi CO_2 yang relatif rendah disebabkan oleh penempatan *transmitter* dengan lokasi yang minim batubara, sedangkan nilai konsentrasi gas CH_4 yang rendah dikarenakan kualitas batubara yang ada di lokasi tersebut termasuk buruk, atau bisa dikatakan tidak layak untuk diambil sebagai hasil tambang. Dengan jarak sejauh 1,6 Km *receiver* masih bisa menerima data yang dikirim oleh *transmitter* dengan baik. Respon yang didapatkan dari sensor cukup lambat dikarenakan prinsip kerja sensor tersebut.



Gambar 4.15 Grafik pengambilan data ketiga

Gambar 4.16 Pengambilan data ketiga



Gambar 4.17 Tampilan pengambilan data ketiga pada aplikasi Blynk

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan percobaan yang telah dilakukan pada pelaksanaan tugas akhir ini didapat beberapa kesimpulan yaitu, pembacaan sensor dirasa masih belum sempurna karena adanya beberapa faktor, diantaranya faktor angin yang menentukan sumber gas dapat mendekati sensor dengan baik atau tidak, dan juga posisi penempatan sensor terhadap sumber. Kalibrasi sensor gas dirasa cukup sulit dikarenakan ada faktor yang cukup mempengaruhi, yaitu tegangan input yang digunakan. Modul GPS dapat membaca lokasi dengan baik ketika berada di luar ruangan, ketika di dalam ruangan sulit untuk mendapatkan data. LoRa dapat mengirimkan data dengan baik dalam jarak kurang lebih 600 meter jika dicoba ditempat yang memiliki banyak halangan atau interferensi seperti pohon-pohon yang tinggi, gedung tinggi, sehingga frekuensi yang dikirimkan *Transmitter* tidak sampai *Receiver*. Namun di area pertambangan LoRa dapat mengirimkan data hingga 1,6 Km. Nilai rata-rata CH_4 dan CO_2 di area pertambangan dari tiga kali pengujian adalah 18,72 ppm dan 2196,20 ppm. Dalam aplikasi Blynk, data yang diterima dari NodeMCU belum bisa secara cepat sampai ke aplikasi dikarenakan ada faktor yang mempengaruhi, seperti jaringan seluler ketika menghubungan Blynk di *smartphone*, ketika jaringan melemah maka pengiriman data bisa menjadi telat dari NodeMCU ke Blynk dikarenakan koneksinya menggunakan menggunakan WiFi sehingga jaringan mempengaruhi.

5.2 Saran

Sebagai sarana pengembangan Alat Pemantauan Kadar Gas pada Tambang Batubara ini, maka terdapat beberapa saran dari penulis berdasarkan hasil yang diperoleh pada saat percobaan. Pengiriman data secara nirkabel memerlukan divais yang lebih mumpuni, terutama di bagian antena nya, memerlukan antena yang bisa mengirim data lebih jauh di berbagai jenis halangan dengan baik. Penelitian selanjutnya dapat menambahkan aktuator dalam serangkaian sistemnya, dikarenakan ketika direalisasikan di area pertambangan, akan lebih baik jika alat tersebut bisa melakukan sesuatu yang dapat membantu dalam proses pemantauan. Penelitian selanjutnya disarankan untuk menjalankan penelitian secara tim tidak secara individu, karena cukup sulit jika diamati seorang diri.

---Halaman ini sengaja dikosongkan---

DAFTAR PUSTAKA

[1] G.L. Borhade, M. B. Kadu, Dr. R. P. Labade, "GSM Operated Wireless Sensor Based Mine Security and Safety Approach". [Accessed: 8-Mar-2019]

[2] S. Molina, I. Soto, R. Carrasco, "Detection of Gases and Collapses in Underground Mines using WSN". [Accessed: 4-May-2019].

[3] Jian Wang, Peng Wang, "Based on Wireless Sensor Network Coal Mine Gas Monitoring System", *2012 International Conference on Industrial Control and Electronics Engineering*, 2012.

[4] Hanwei Electronics Co., Ltd., "Technical Data MQ-2 Gas Sensor", [Accessed: 5-May-2019].

[5] V. Ravindra and S. M. Rajbhoj, "Identification of Toxic Gases using Electronic Nose," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, PUNE, India, 2017, pp. 1–5.

[6] Hanwei Electronics Co., Ltd., "Technical Data MQ-135 Gas Sensor" [Accessed: 5-Oct-2019].

[7] Zhiqiang Wei, Yaqing Song, Hao Liu, Yanxiu Sheng, Xi Wang, "The Research and Implementation of GPS Intelligent Transmission Strategy Based on on-board Android Smartphones", in *2013 3rd International Conference on Computer Science and Network Technology*, 2013.

[8] u-blox Co., "u-blox 6 GPS Modules Data Sheet" [Accessed: 1-Nov-2019].

[9] Hoperf Electronic., "RFM95/96/97/98(W) – Low Power Long Range Transceiver Module V1.0" [Accessed: 1-Nov-2019]

[10] "Kominfo Rilis Aturan IoT dan Konsolidasi Operator pada Kuartal I-2019" [Online]. Available: https://www.katadata.co.id/berita/2019/01/09/kominfo-rilis-aturan-iot-dan-konsolidasi-operator-pada-kuartal-i-2019. [Accessed: 06-Jan-2020].

[11] "Arduino Uno Rev3" [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed: 1-Nov-2019].

[12] L. K. P. Saputra and Y. Lukito, "Implementation of air

conditioning control system using REST protocol based on NodeMCU ESP8266," in *2017 International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICONSONICS)*, Yogyakarta, Indonesia, 2017, pp. 126–130.

[13] "UM10204 I2C-bus specification and user manual," vol. 2014, p. 64, 2014.

[14] "Dasar-Dasar Serial Peripheral Interface (SPI) Mikrokontroler" [Online]. Available: https://insinyoer.com/dasar-dasar-serial-peripheral-interace-spi-mikrokontroler. [Accessed: 1-Nov-2019].

[15] "Back to Basics: The Universal Asynchronous Receiver/Transmitter (UART)" [Online}. Available: https://allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart. [Accessed: 1-Nov-2019].

[16] N. A. Z. M. Noar and M. M. Kamal, "The development of smart flood monitoring system using ultrasonic sensor with blynk applications," in *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, 2017, pp. 1–6.

[17] Qin Xianli, Fu Mingchao, Shen Bin, "Coal Mine Gas Wireless Monitoring System Based on WSNs", *Second International Conference on Digital Manufacturing & Automation*, 2011.

[18] Zhang Changsen, Mao Yan, "Study on Mine Communication Network Based on Ethernet and WSN", *International Conference on Computational Intelligence and Communication Networks*, 2015.

[19] Liu Xiaoyang, Qiao Zhi, Lv Hongjie, "The Event-Driven Wireless Sensor Networks of Coal Mine", *9th IEEE International Conference on Communication Software and Networks*, 2017.

[20] Kumar Keshamoni, Sabbani Hemanth, "Smart Gas Level Monitoring, Booking & Gas Leaked Detector over IoT", *IEEE 7th International Advance Computing Conference*, 2017.

[21] Dedy Rahman Wijaya, Riyanarto Sarno, Enny Zulaika, "Gas Concentration Analysis of Resistive Gas Sensor Array", *International Symposium on Electronics and Smart Devices,* 29-30 November 2016.

[22] Metta Santiputri, Muhammad Tio. "IoT-based Gas Leak

Detection Device". [Accessed: 7-May-2019]
[23] Huang-Chen Lee, Kai-Hsiang Ke, "Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation", *IEEE Transactions on Instrumentation and Measurement, Vol. 67, No. 9,* September 2018.
[24] Shilpa Devalal, A. Karthikeyan, "LoRa technology-an overview", *2nd International conference on Electronics, Communication and Aerospace Technology*, 2018.
[25] Alexandru Lavric, Valentin Popa. "Internet of Things and LoRa Low-Power Wide-Area Networks: A Survey", [Accessed: 7-May-2019].
[26] Luthfan Aufar Akbar, Muhammad Rivai, Fajar Budiman. "Rancang Bangun Sensor Node pada Wireless Sensor Network Menggunakan Deret Sensor Gas dan Jaringan Syaraf Tiruan untuk Mendeteksi Kebakaran Hutan", Jurnal Teknik ITS, Vol. 5, No. 2, 2016.
[27] Hendrik Hermawan, Muhammad Rivai. "Sistem Pemantauan Gunung Berapi Berbasis IoT Menggunakan NodeMCU dan LoRa", [Accessed: 5-May-2019].

---Halaman ini sengaja dikosongkan---

LAMPIRAN A
Program Arduino

1. Kode Program Pembacaan dan Pengiriman Data

```
#include <Wire.h>
#include <SPI.h>
#include <RH_RF95.h>
#include <LiquidCrystal_I2C.h>
#include <max6675.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>


LiquidCrystal_I2C lcd(0x27, 16, 2);

/*****************Define Software Serial*****************/

TinyGPSPlus gps;
SoftwareSerial ss(8, 9);

/*********************Define Sensor*********************/

#define     MQ02                                (A0)
#define     MQ135                               (A1)

/*****************Sample Times And
Interval***************/

#define Calibration_Sample_Times                (500)
#define Calibration_Sample_Interval             (250)

#define Read_Sample_Interval                    (50)
#define Read_Sample_Times                       (50)

/*********************Gas Sensor*********************/

#define         GAS_CL2                         (0)
```

```
#define     GAS_O3                  (1)
#define     GAS_CO2                 (2)
#define     GAS_CO                  (3)
#define     GAS_NH4                 (4)
#define     GAS_CH3                 (6)
#define     GAS_CH3_2CO             (7)
#define     GAS_H2                  (8)
#define     GAS_C2H5OH              (9)
#define     GAS_C4H10               (10)
#define     GAS_LPG                 (11)
#define     GAS_Smoke               (12)
#define     GAS_CO_sec              (13)
#define     GAS_LPG_sec             (14)
#define     GAS_CH4                 (15)
#define     GAS_NO2                 (16)
#define     GAS_SO2                 (17)
#define     GAS_C7H8                (18)
#define     GAS_H2S                 (19)
#define     GAS_NH3                 (20)
#define     GAS_C6H6                (21)
#define     GAS_C3H8                (22)
#define     GAS_NHEX                (23)
#define     GAS_HCHO                (24)

/*********************Curva Gas MQ02*****************/

float       LPG_Curve[2]            =  {538.5973, -2.1079};
float       C3H8_terCurve[2]        =  {606.8924, -2.1399};
float       H2_Curve[2]             =  {924.4283, -2.0903};
float       CH4_Curve[2]            =  {3783.4377, -2.6212};
float       CO_terCurve[2]          =  {26561.9922, -2.9655};
float       C2H5OH_secCurve[2]      =  {3217.3999, -2.6350};
float       Smoke_Curve[2]          =  {3158.7669, -2.2552};

/*********************Curva Gas MQ09*****************/

float       CO2_Curve[2]       = {113.7105289, -3.019713765};
float       CO_Curve[2]        = {726.7809737, -4.040111669};
float       NH4_Curve[2]       = {84.07117895, -4.41107687};
```

```
float         C2H5OH_Curve[2]     = {74.77989144, 3.010328075};
float         CH3_Curve[2]        = {47.01770503, -3.281901967};
float         CH3_2CO_Curve[2]    = {7.010800878, -
2.122018939};

/******************Inisialisasi Calibration***************/

unsigned long SLEEP_TIME = 600;
//float          Ro0;
//float          Ro1;
float         Ro0 = 8.41;
float         RL0 = 2.36;
float         Ro1 = 66.01;
float         RL1 = 2.497;

/*********************Gas dan Suhu*******************/

float         CH4;
float         CO2;

/****************************************************/

uint8_t data[100];

// Singleton instance of the radio driver
float frequency = 915.0;
RH_RF95 rf95;
// RH_RF95 rf95(5, 2);
// RH_RF95 rf95(8, 3);
//#define Serial SerialUSB

void setup()
{
  Serial.begin(9600);
  ss.begin(9600);

  lcd.init();
  lcd.backlight();
  //Serial.print("Calibrating.....\n");
```

```
  //Ro0 = MQCalibration(MQ02, 1.8602, RL0, CH4_Curve);
  //Ro1 = MQCalibration(MQ135, 400, RL1, CO2_Curve);
  //Serial.print("Ro0 = "); Serial.print(Ro0); Serial.print(" Kohm");
  //Serial.print(" Ro1 = "); Serial.print(Ro1); Serial.println("
Kohm");
  //Serial.println("\n");
  // lcd.setCursor(1,0);
  // lcd.print("Hello Fauzi");
  while (!Serial) ; // Wait for serial port to be available
  if (!rf95.init())
    Serial.println("init failed");
  rf95.setFrequency(frequency);
  rf95.setTxPower(13);
}

void loop()
{
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
  {
    Serial.println("Sending to rf95_server");
    data_sender();
    lcdDisplay();

  rf95.send(data, sizeof(data));

  rf95.waitPacketSent();
  // Now wait for a reply
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);

  if (rf95.waitAvailableTimeout(3000))
  {
    // Should be a reply message for us now
    if (rf95.recv(buf, &len))
    {
      Serial.print("got reply: ");
      Serial.println((char*)buf);
      //      Serial.print("RSSI: ");
```

```
    //      Serial.println(rf95.lastRssi(), DEC);
   }
   else
   {
    Serial.println("recv failed");
   }
  }
  else
  {
   Serial.println("No reply, is rf95_server running?");
  }
  delay(400);
  }

// if (millis() > 1000 && gps.charsProcessed() < 10)
//  {
//   Serial.println(F("No GPS detected: check wiring."));
//   while(true);
//  }
}

void lcdDisplay() {
  lcd.setCursor(0, 0);
  lcd.print("CH4:");
  lcd.setCursor(4, 0);
  lcd.print(CH4);
  lcd.setCursor(13, 0);
  lcd.print("ppm");
  lcd.setCursor(0, 1);
  lcd.print("CO2:");
  lcd.setCursor(4, 1);
  lcd.print(CO2);
  lcd.setCursor(13, 1);
  lcd.print("ppm");
}

void data_sender() {
CH4 = MQGetGasPercentage(MQRead(MQ02, RL0), Ro0,
GAS_CH4, MQ02);
```

```
CO2   = MQGetGasPercentage(MQRead(MQ135, RL1), Ro1,
GAS_CO2, MQ135);

  char gps_lon[20]={"\0"};
  char gps_lat[20]={"\0"};
  if(gps.location.isValid())
  {
   dtostrf(gps.location.lat(), 0, 6, gps_lat);
   dtostrf(gps.location.lng(), 0, 6, gps_lon);
  }

  char data_1[10] = {"\0"};
  char data_2[10] = {"\0"};

  char data_sensor[100] = {"\0"};
  dtostrf(CH4, 5, 2, data_1);
  dtostrf(CO2, 5, 2, data_2);

  strcat(data_sensor, data_1);
  strcat(data_sensor, "\n");
  strcat(data_sensor, data_2);
  strcat(data_sensor, "\n");
  strcat(data_sensor, gps_lat);
  strcat(data_sensor, "\n");
  strcat(data_sensor, gps_lon);
  strcat(data_sensor, "\n");

  strcpy((char*)data, data_sensor);
  Serial.println((char*)data);

}

/***************MQResistanceCalculation *****************/

float MQResistanceCalculation(int raw_adc, float rl_value)
{
 return
   (long)((long)(1024 * (long)rl_value) / raw_adc - (long)rl_value);
}
```

```
/*********************MQCalibration******************/

float MQCalibration(int mq_pin, double ppm, double rl_value, float
*pcurve )
{
  int i;
  float val = 0;

  for (i = 0; i < Calibration_Sample_Times; i++) {      //take multiple
samples
    val += MQResistanceCalculation(analogRead(mq_pin),
rl_value);
    delay(Calibration_Sample_Interval);
  }
val = val / Calibration_Sample_Times;                 //calculate the
average value
  //Ro = Rs * sqrt(a/ppm, b) = Rs * exp( ln(a/ppm) / b )

  return  //val;
    (float)val * exp((log(pcurve[0] / ppm) / pcurve[1]));

}

/************************MQRead*********************/

float MQRead(int mq_pin, float rl_value)
{
  int i;
  float rs = 0;

  for (i = 0; i < Read_Sample_Times; i++) {
    rs += MQResistanceCalculation(analogRead(mq_pin), rl_value);
    delay(Read_Sample_Interval);
  }

  rs = rs / Read_Sample_Times;

  return rs;
```

```
}

/***************MQGetGasPercentage ******************/

float MQGetGasPercentage(float rs_ro_ratio, float ro, int gas_id, int
sensor_id)
{
 if (sensor_id == MQ02) {
   if ( gas_id == GAS_CH4 ) {
     return MQGetPercentage(rs_ro_ratio,  ro, CH4_Curve);
//MQ02
   } else if ( gas_id == GAS_LPG ) {
     return MQGetPercentage(rs_ro_ratio,  ro, LPG_Curve); //MQ02
   } else if ( gas_id == GAS_C3H8 ) {
     return MQGetPercentage(rs_ro_ratio,  ro, C3H8_terCurve);
//MQ02
   } else if ( gas_id == GAS_H2 ) {
     return MQGetPercentage(rs_ro_ratio, ro, H2_Curve); //MQ02
   } else if ( gas_id == GAS_CO ) {
     return MQGetPercentage(rs_ro_ratio, ro, CO_terCurve);
//MQ02
   } else if ( gas_id == GAS_C2H5OH ) {
     return MQGetPercentage(rs_ro_ratio, ro, C2H5OH_Curve);
//MQ02
   }
 }
 else if (sensor_id == MQ135) {
   if ( gas_id == GAS_CO2 ) {
     return MQGetPercentage(rs_ro_ratio, ro, CO2_Curve);
//MQ135
   } else if ( gas_id == GAS_C2H5OH ) {
     return MQGetPercentage(rs_ro_ratio,  ro, C2H5OH_Curve);
//MQ135
   } else if ( gas_id == GAS_CH3 ) {
     return MQGetPercentage(rs_ro_ratio, ro,CH3_Curve);
//MQ135
   } else if ( gas_id == GAS_CH3_2CO ) {
     return MQGetPercentage(rs_ro_ratio,  ro,CH3_2CO_Curve);
```

```
//MQ135
   } else if ( gas_id == GAS_CO ) {
     return MQGetPercentage(rs_ro_ratio,  ro,CO_Curve); //MQ135
   } else if ( gas_id == GAS_NH4 ) {
     return MQGetPercentage(rs_ro_ratio,  ro,NH4_Curve);
//MQ135
   }
  }
  return 0;
}

/*******************MQGetPercentage*****************/

float  MQGetPercentage(float rs_ro_ratio, float ro, float *pcurve)
{
  return (float)(pcurve[0] * pow(((float)rs_ro_ratio / ro),
pcurve[1]));
}
```

2. Kode Program Penerimaan dan Pengolahan Data

```
#include <SPI.h>
#include <RH_RF95.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

SoftwareSerial kirimSerial(7,6);

/***************************Inisial Pengiriman
Serial***************************/
char start = '<';
char stopped = '>';
/*****************************Inisial Pemisah
Data*****************************/
char nilai[5][10];
char line = '\n';
```

```
const byte numChars = 50;
char dataTerima[RH_RF95_MAX_MESSAGE_LEN];
/*****************************Inisial
RH_RF95*********************************/
// Singleton instance of the radio driver
float frequency = 915.0;
RH_RF95 rf95;
//RH_RF95 rf95(5, 2); // Rocket Scream Mini Ultra Pro with the
RFM95W
//RH_RF95 rf95(8, 3); // Adafruit Feather M0 with RFM95

// Need this on Arduino Zero with SerialUSB port (eg
RocketScream Mini Ultra Pro)
//#define Serial SerialUSB
/*****************************************************
************************/

void setup()
{
  // Rocket Scream Mini Ultra Pro with the RFM95W only:
  // Ensure serial flash is not interfering with radio communication
on SPI bus
  Serial.begin(9600);
  kirimSerial.begin(9600);
  lcd.init();
  lcd.backlight();
  while (!Serial) ; // Wait for serial port to be available
  if (!rf95.init())
    Serial.println("init failed");
    rf95.setFrequency(frequency);
    rf95.setTxPower(13);
  // Defaults after init are 434.0MHz, 13dBm, Bw = 125 kHz, Cr =
4/5, Sf = 128chips/symbol, CRC on

  // The default transmitter power is 13dBm, using PA_BOOST.
  // If you are using RFM95/96/97/98 modules which uses the
PA_BOOST transmitter pin, then
  // you can set transmitter powers from 5 to 23 dBm:
//  driver.setTxPower(23, false);
```

```
  // If you are using Modtronix inAir4 or inAir9,or any other module
which uses the
  // transmitter RFO pins and not the PA_BOOST pins
  // then you can configure the power transmitter power for -1 to 14
dBm and with useRFO true.
  // Failure to do that will result in extremely low transmit powers.
//  driver.setTxPower(14, true);
}

void loop()
{
//  if (rf95.available())
//  {
    // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);

//    kirimSerial.write(start);
//    kirimSerial.write(dataTerima,sizeof(dataTerima));
//    kirimSerial.write(stopped);

  if (rf95.waitAvailableTimeout(10000))
  {
     if (rf95.recv(buf, &len))
     {
       String dataSensor =  (char*)buf;
//       char dataTerima[RH_RF95_MAX_MESSAGE_LEN];
       dataSensor.toCharArray(dataTerima,sizeof(dataTerima));
       Serial.println(dataTerima);
       kirimSerial.write(start);
       kirimSerial.write(dataTerima,sizeof(dataTerima));
       kirimSerial.write(stopped);
       dataPerSensor();
//       kirimDataSerial();
//       RH_RF95::printBuffer("request: ", buf, len);
       Serial.println("got request: ");
//       Serial.println((char *)buf);
//       Serial.println(dataTerima);
       lcdDisplay();
```

```
//        Serial.print("RSSI: ");
//        Serial.println(rf95.lastRssi(), DEC);

        // Send a reply
        uint8_t data[] = "I'm Back";
        rf95.send(data, sizeof(data));
        rf95.waitPacketSent();
        Serial.println("Sent a reply");
      }
      else
      {
        Serial.println("recv failed");
      }
    }
    else
    {
      Serial.println("No reply, is rf95_server running?");
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("failed");
    }
//  }
  delay(400);
}

void dataPerSensor(){
  int data_ke = 0;
  int index = 0;
  int i;
  for(i = 0; i <= numChars ; i++){
    //i++;
    if(dataTerima[i] != line){
    nilai[data_ke][index] = dataTerima[i];
    index++;
    }
    else
    {
    data_ke++;
    index = 0;
```

```
    }
  }
}

void lcdDisplay(){
  lcd.setCursor(0, 0);
  lcd.print("CH4:");
  lcd.setCursor(4, 0);
  lcd.print(nilai[0]);
  lcd.setCursor(13, 0);
  lcd.print("ppm");
  lcd.setCursor(0, 1);
  lcd.print("CO2:");
  lcd.setCursor(4, 1);
  lcd.print(nilai[1]);
  lcd.setCursor(13, 1);
  lcd.print("ppm");
}

//void kirimDataSerial(){
//  kirimSerial.write(start);
//  kirimSerial.write(dataTerima,sizeof(dataTerima));
//  kirimSerial.write(stopped);
//}
```

3. Kode Program Pengolahan dan Pengiriman Data Sensor ke Aplikasi

```
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define BLYNK_PRINT Serial

SoftwareSerial terimaSerial(D7,D8);

/**************************Inisial Autentifiasi
Blynk**************************/
char auth[] = "vmWnwcKVv7yB84l0HR5PP7CyxbcP0o-G";
```

```
char ssid[] = "Kemproh";
char pass[] = "yekyekyek";
/*******************************Inisial Map
Blynk*******************************/
int index_1 = 1;
//float lat = -7.2849094;
//float lon = 112.7960237;

//String dataSensor_1;
////char data_1[10]={ "\0" };
////char data_2[10]={ "\0" };
//////int dat_1 = 100;
//////int dat_2 = 200;
//////char shift = '\n';
char dataMap[20];

/*******************************Terima dan Olah
Data*****************************/
char dataSensor[10][10];
char line = '\n';
const byte numChars = 40;
char receivedChars[numChars];

boolean newData = false;
/*****************************************************
***************************/

void setup() {
   Serial.begin(9600);
   terimaSerial.begin(9600);
   Blynk.begin(auth, ssid, pass);
   Serial.println("<Arduino is ready>");
}

void loop() {
   recvWithStartEndMarkers();
   showNewData();
   dataSet();
```

```
    dataNilai();
    dataBlynk();
    Blynk.run();
//    Serial.println("DATA");
//    Serial.println(dataSensor[0]);
//    Serial.println(dataSensor[1]);
}

void recvWithStartEndMarkers() {
    static boolean recvInProgress = false;
    static byte ndx = 0;
    char startMarker = '<';
    char endMarker = '>';
    char rc;

    while (terimaSerial.available() > 0 && newData == false) {
        rc = terimaSerial.read();

        if (recvInProgress == true) {
            if (rc != endMarker) {
                receivedChars[ndx] = rc;
                ndx++;
                if (ndx >= numChars) {
                    ndx = numChars - 1;
                }
            }
            else {
                receivedChars[ndx] = '\0'; // terminate the string
                recvInProgress = false;
                ndx = 0;
                newData = true;
            }
        }

        else if (rc == startMarker) {
            recvInProgress = true;
        }
    }
}
```

```
void showNewData() {
    if (newData == true) {
       //Serial.println("This just in ... ");
       Serial.println(receivedChars);
        newData = false;
    }
}

void dataSet(){
  recvWithStartEndMarkers();
  int data_ke = 0;
  int index = 0;
  int i;
  for(i = 0; i <= numChars ; i++){
    //i++;
    if(receivedChars[i] != line){
    dataSensor[data_ke][index] = receivedChars[i];
    index++;
    }
    else
    {
    data_ke++;
    index = 0;
    }
  }
}

void dataBlynk(){
  dataSet();
  dataNilai();
  Blynk.virtualWrite(V1, dataSensor[0]);
  Blynk.virtualWrite(V2, dataSensor[1]);
  Blynk.virtualWrite(V4, index_1, dataSensor[2], dataSensor[3],
dataMap);
}

void dataNilai(){
  char dataSenMap[50] = {"\0"};
```

```
//  dtostrf(dataSensor[0], 5, 2, data_1);
//  dtostrf(dataSensor[1], 5, 2, data_2);
  strcat(dataSenMap, "CH4 =");
  strcat(dataSenMap, dataSensor[0]);
  strcat(dataSenMap, "   ");
  strcat(dataSenMap, "CO2 =");
  strcat(dataSenMap, dataSensor[1]);
  strcat(dataSenMap, "   ");
  strcpy(dataMap, dataSenMap);
//  Serial.println(dataMap);
}
```

LAMPIRAN B
Hasil Pengujian

1. Pengambilan pertama pada jarak 600 meter

Pengambilan Data Pertama		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
8	28,35	4886,88
16	28,14	4749,69
24	28,35	4871,38
32	28,63	4886,88
40	27,97	4871,38
48	28,57	4448,95
56	28,74	4435,27
64	28,71	4301,5
72	28,71	4185,65
80	28,6	4223,8
88	28,63	4185,65
96	28,71	4185,65
104	28,77	4211,03
112	28,82	4185,65
120	28,6	4185,65
128	28,57	4185,65
136	28,68	4185,65
144	28,68	4185,65
152	28,46	4185,65
160	26,63	4185,65
168	24,5	4185,65
176	28,41	4185,65

Pengambilan Data Pertama		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
184	28,46	4185,65
192	28,46	4185,65
200	28,41	4185,65
208	28,3	4185,65
216	28,19	4185,65
224	28,35	4185,65
232	27,92	4185,65
240	26,43	4185,65
248	25,37	4185,65
256	24,86	4185,65
264	24,23	4185,65
272	25,04	4185,65
280	24,95	4185,65
288	25,04	4185,65
296	24,81	4185,65
304	25,23	4185,65
312	25,51	4185,65
320	25,13	4185,65
328	25,65	4185,65
336	25,18	4185,65
344	24,95	4185,65
352	25,6	4185,65
360	27,92	4185,65
368	27,87	4185,65
376	28,03	4185,65
384	27,87	4185,65

Pengambilan Data Pertama		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
392	28,08	4185,65
400	28,46	4185,65
408	27,66	4185,65
416	28,24	4185,65
424	28,14	4185,65
432	28,57	4185,65
440	28,03	4185,65
448	28,08	4185,65
456	28,35	4185,65
464	27,13	4185,65
472	27,81	4185,65
480	27,76	4185,65
488	27,55	4185,65
496	27,6	4185,65
504	27,24	4185,65
512	25,13	4185,65
520	27,34	4173,04
528	27,39	4185,65
536	27,34	4185,65
544	26,33	4160,48
552	26,13	4173,04
560	25,79	4185,65

2. Pengambilan kedua pada jarak 1 Km.

Pengambilan Data Kedua		
Waktu (Detik)	**Sensor**	
	MQ02 (PPM)	**MQ135 (PPM)**
8	14,28	1203,47
16	14,1	1225,38
24	14,25	1218,02
32	13,91	1179,74
40	13,85	1218,02
48	14,21	1227,85
56	14,17	1218,02
64	13,95	1230,32
72	14	1227,85
80	14,09	1225,38
88	13,98	1184,43
96	14,03	1220,46
104	14,08	1201,07
112	13,95	1230,32
120	13,98	1213,14
128	14,01	1230,32
136	14,86	1230,32
144	13,94	1198,67
152	13,93	1196,28
160	13,98	1227,85
168	14,13	1222,92
176	14,54	1225,38
184	14,64	1191,53
192	14,33	1225,38

Pengambilan Data Kedua		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
200	14,15	1163,49
208	14,06	1182,08
216	14,59	1182,08
224	14,51	1165,79
232	14,32	1158,9
240	14,19	1165,79
248	14,44	1158,9
256	14,64	1175,06
264	14,89	1165,79
272	14,7	1175,06
280	14,86	1158,9
288	14,83	1170,42
296	14,96	1161,2
304	14,8	1172,74
312	14,93	1165,79
320	14,66	1156,62
328	14,9	1120,87
336	14,92	1114,34
344	14,64	1125,26
352	14,9	1129,67
360	14,62	1165,79
368	14,64	1152,07
376	14,47	1123,06
384	14,09	1127,46
392	14,39	1116,51
400	13,8	1116,51

Pengambilan Data Kedua		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
408	13,87	1116,51
416	13,98	1114,34
424	14,06	1138,56
432	14,22	1163,49
440	14,18	1161,2
448	14,01	1154,34
456	14,63	1158,9
464	14,19	1154,34
472	13,97	1168,1
480	14,06	1152,07
488	13,71	1154,34
496	13,77	1114,34
504	14,01	1127,46
512	13,34	1120,87
520	13,6	1134,1
528	14	1149,8
536	14,61	1127,46
544	14,72	1116,51
552	14,34	1165,79
560	14,61	1123,06
568	14,73	1127,46
576	14,61	1127,46
584	13,72	1165,79
592	12,48	1118,69
600	13,59	1163,49
608	12,81	1114,34

Pengambilan Data Kedua		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
616	12,8	1127,46
624	13,85	1125,26
632	13,82	1118,69
640	13,69	1140,8
648	13,34	1138,56
656	12,92	1131,88
664	13,89	1116,51
672	13,68	1120,87
680	13,87	1125,26
688	13,71	1145,29
696	13,78	1147,54

3. Pengambilan ketiga pada jarak 1,6 Km.

Pengambilan Data Ketiga		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
8	14,89	1237,78
16	14,47	1227,85
24	14,96	1230,32
32	14,73	1235,29
40	14,84	1230,32
48	14,61	1230,32
56	14,96	1230,32
64	14,7	1225,38
72	14,96	1203,47

Pengambilan Data Ketiga		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
80	14,63	1186,79
88	14,86	1156,62
96	14,57	1158,9
104	14,86	1163,49
112	14,66	1172,74
120	14,86	1227,85
128	14,66	1186,79
136	14,47	1170,42
144	14,45	1189,15
152	14,67	1179,74
160	14,53	1186,79
168	14,48	1186,79
176	14,18	1170,42
184	14,72	1154,34
192	14,41	1170,42
200	14,29	1165,79
208	14,18	1158,9
216	14,1	1147,54
224	13,98	1165,79
232	13,95	1149,8
240		
248	14,41	1213,14
256	14,18	1205,88
264	14,37	1165,79
272	14,14	1177,4
280	14,31	1163,49

Pengambilan Data Ketiga		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
288	14,63	1182,08
296	14,86	1163,49
304	14,63	1196,28
312	14,85	1230,32
320	14,63	1191,53
328	14,8	1165,79
336	14,47	1170,42
344	14,77	1165,79
352	14,54	1175,06
360	14,76	1163,49
368	14,63	1179,74
376	14,86	1186,79
384	14,63	1168,1
392	14,93	1175,06
400	14,66	1177,4
408	14,83	1152,07
416	14,63	1177,4
424	14,86	1227,85
432	14,63	1222,92
440	14,86	1138,56
448	14,61	1116,51
456	14,93	1116,51
464	14,66	1127,46
472	14,86	1125,26
480	14,63	1138,56
488	14,86	1143,04

Pengambilan Data Ketiga		
Waktu (Detik)	Sensor	
	MQ02 (PPM)	MQ135 (PPM)
496	14,7	1138,56
504	14,86	1143,04
512	14,6	1131,88
520	14,93	1143,04
528	14,66	1170,42
536	15	1215,58
544	14,7	1198,67
552	14,86	1131,88
560	14,66	1131,88
568	14,93	1147,54
576	14,63	1158,9
584	14,93	1196,28
592	14,63	1189,15
600	14,9	1177,4
608	14,57	1230,32
616	14,84	1230,32
624	14,63	1230,32
632	14,9	1230,32
640	14,66	1230,32
648	14,93	1222,92
656	14,7	1230,32
664	14,93	1227,85
672	14,66	1230,32
680	14,83	1182,08
688	14,61	1213,14
696	14,84	1205,88

Pengambilan Data Ketiga		
Waktu (Detik)	**Sensor**	
	MQ02 (PPM)	**MQ135 (PPM)**
704	14,63	1230,32
712	14,9	1156,62
720	14,64	1161,2
728	14,83	1154,34

LAMPIRAN C
Dokumentasi Pengujian Volume

1. Pengujian sensor gas MQ-02 (CH_4)

(a) Konsentrasi 1 mL

(b) Konsentrasi 2 mL

(c) Konsentrasi 3 mL

(d) Konsentrasi 4 mL

(e) Konsentrasi 5 mL

(f) Konsentrasi 6 mL

(g) Konsentrasi 7 mL

(h) Konsentrasi 8 mL

(i) Konsentrasi 9 mL

(j) Konsentrasi 10 mL

2. Pengujian sensor gas MQ-135 (CO_2)

(a) Konsentrasi 1 mL

(b) Konsentrasi 2 mL

(c) Konsentrasi 3 mL

(d) Konsentrasi 4 mL

(e) Konsentrasi 5 mL

(f) Konsentrasi 6 mL

(g) Konsentrasi 7 mL

(h) Konsentrasi 8 mL

(i) Konsentrasi 9 mL

(j) Konsentrasi 10 mL

LAMPIRAN D
Dokumentasi

BIODATA PENULIS

Fauzi Muhammad Ikhsan, biasa dipanggil Onzin lahir di Balikpapan pada tanggal 7 Nopember 1997 merupakan anak kedua dari dua bersaudara. Penulis pernah bersekolah di SD Nasional KPS, SMP Nasional KPS, SMA Negeri 1 Balikpapan. Penulis melanjutkan studi S1 di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya. Selama masa perkuliahan, penulis memilih untuk masuk ke Bidang Studi Elektronika dikarenakan tertarik dalam bidang tersebut. Penulis juga menjadi asisten lab di Bidang Studi Elektronika. Selama masa perkuliahan penulis aktif dalam organisasi pecinta alam yaitu Kalpataru. Penulis mempunyai hobi lari lintas alam, climbing dan rappeling, serta mendaki gunung. Banyak pelajaran kehidupan yang bisa kita ambil dari alam, dan pergilah mencari pengalaman karena pelajaran terbaik didapatkan dari pengalaman.